ZNALOSTI 2005

Sborník příspěvků 4. ročníku konference
Proceedings of the 4th annual conference

Fakulta elektrotechniky a informatiky,
VŠB – Technická univerzita Ostrava

ISBN 80–248–0755–6

ZNALOSTI 2005

http://webocrat.fei.tuke.sk/znalosti2005

4. ročník konference, Stará Lesná, 9. – 11. února 2005
Sborník příspěvků

4th Annual Conference, Stará Lesná,
9th – 11th February 2005
Proceedings

Pořádající instituce (organized by)
Technická univerzita v Košiciach, Katedra kybernetiky a umelej inteligencie
Slovenská akadémia vied, Bratislava, Ústav informatiky
Slovenská spoločnosť pre umelú inteligenciu
Univerzita Pavla Jozefa Šafárika v Košiciach, PF, Ústav informatiky
ČVUT Praha, FEL, Gerstnerova laboratoř
Masarykova univerzita v Brně, Fakulta informatiky
Univerzita Karlova v Praze, MFF, Katedra softwarového inženýrství
VŠB – Technická univerzita Ostrava, FEI, Katedra informatiky
VŠE Praha, FIS, Katedra informačního a znalostního inženýrství

ZNALOSTI 2005

Technická redakce:
Jiří Dvorský, jiri.dvorsky@vsb.cz
Michal Kolovrat, michal.kolovrat@vsb.cz
Michal Krátký, michal.kratky@vsb.cz

Fakulta elektrotechniky a informatiky,
VŠB–Technická univerzita Ostrava

Počet stran: 344
Náklad: 140 ks
Vydání: 1.
Rok vydání: 2005

Pro sazbu a sestavení sborníku byl použit systém LaTeX.
Tisk a knihařské práce provedlo Vydavatelství Univerzity Palackého v Olomouci.

Vydala Fakulta elektrotechniky a informatiky, VŠB–Technická univerzita Ostrava.

Předmluva

Je mi ctí, že mohu uvést sborník konference ZNALOSTI, čtvrté toho jména, která se letos poprvé koná na Slovensku. Odborný program se tentokrát kromě vlastních příspěvků skládá ze dvou tutoriálů a šesti zvaných přednášek, na nichž se podílejí jak slovenští a čeští tak i zahraniční autoři, a panelové diskuse.

Zaslané příspěvky se letos týkají všech hlavních proudů této oblasti informační vědy, tj. dobývání znalostí z databází, inteligentního zpřístupňování textových informací, znalostního inženýrství a znalostního managementu. Ze 79 nabídnutých příspěvků bylo do tohoto sborníku vybráno 13 dlouhých a 21 zkrácených příspěvků.
Tématy tohoto ročníku jsou pak především:

- dolování v datech a v textu,
- induktivní logické programování,
- konceptuální svazy,
- logické programování a nemonotónní odvozování,
- metody shlukové analýzy,
- robotický fotbal,
- sémantický web a ontologie,
- strojové učení a přirozený jazyk,
- vyhledávání informací,
- znalosti v podnikové sféře,
- znalostní systémy.

Na tomto místě chci poděkovat všem, kteří se na přípravě tohoto sborníku a celé konference podíleli. Především jsou to autoři zaslaných příspěvků, tutoriálů a zvaných přednášek, neboť bez jejich velkého úsilí by se tato konference nemohla konat. Členům řídícího a programového výboru patří dík nejen za odpovědné hodnocení příspěvků, ale též za iniciativu a spolupráci během celého půlročního období přípravy této konference.

Zvláštní dík zaslouží organizační výbor vedený Janem Paraličem z Technické univerzity Košice za zajištění organizace naší konference, Michal Krátký s Jiřím Dvorským a Michalem Kolovratem z VŠB–Technické univerzity Ostrava za nezáviděníhodnou práci spojenou s přípravou sborníků a Jan Blaťák za pomoc během recenzního řízení.

Luboš Popelínský
předseda programového výboru
ZNALOSTI 2005

Preface

I am pleased to present the proceedings of the 4th conference ZNALOSTI (Knowledge) held – first time in Slovakia – in Stará Lesná, The High Tatras. The scientific program consists of two tutorials and six invited talks presented by Slovak, Czech as well as foreign experts, the panel discussion, contributed talks and poster sessions.

The contributed papers covers theoretical as well as practical aspects of knowledge discovery, information retrieval, knowledge engineering and knowledge management. This volume contains 13 long papers and 21 short ones that has been selected from 79 submitted manuscripts.

The main topics of this year conference are:

- data mining and text mining,
- inductive logic programming,
- formal concept analysis,
- logic programming and non-monotonic reasoning,
- cluster analysis,
- robotic football,
- ontologies and semantic web,
- natural language learning,
- information retrieval,
- knowledge management,
- knowledge-based systems.

I would like to thank to everybody who participates in preparation of this volume, first of all to the authors of tutorials, invited talks and contributed talks and posters, program committee members and other reviewers. Without their enormous effort, this conference would hardly hold.

My special thanks go to Jan Paralič from Technical University Košice and all members of the organizing committee for organizing our conference, to Michal Krátký, Jiří Dvorský and Michal Kolovrat from VŠB–Technical University of Ostrava for their enormous effort when collecting final versions of papers and finalizing the proceeding, and to Jan Blaťák, Masaryk University in Brno, for his help during the review process.

Luboš Popelínský
Program Chair
ZNALOSTI 2005

Organizace (Organization)

Řídící výbor série konferencí Znalosti (Steering Committee)

Předseda (Chair):
Jan Rauch (Vysoká škola ekonomická v Praze)

Členové (Members):
Ján Paralič (Technická univerzita v Košiciach)
Jaroslav Pokorný (Univerzita Karlova v Praze)
Lubomír Popelínský (Masarykova univerzita v Brně)
Václav Snášel (VŠB–Technická univerzita Ostrava)
Vojtěch Svátek (Vysoká škola ekonomická v Praze)
Olga Štěpánková (České vysoké učení technické v Praze)

Programový výbor (Program Committee)

Předseda (Chair):
Lubomír Popelínský (Masarykova univerzita v Brně)

Členové (Members):
Mária Bieliková (Slovenská technická univerzita v Bratislave)
Petr Berka (Vysoká škola ekonomická v Praze)
Ivan Bruha (McMaster University, Hamilton, Kanada)
Jiří Dvorský (VŠB–Technická univerzita Ostrava)
Jan Hajič (Univerzita Karlova v Praze)
Bohumil Horák (VŠB–Technická univerzita Ostrava)
Petr Hujňák (Per Partes Consulting, Praha)
Dušan Húsek (Ústav informatiky, AV ČR, Praha)
Jiří Ivánek (Vysoká škola ekonomická v Praze)
Karel Ježek (Západočeská univerzita v Plzni)
Petr Jirků (Univerzita Karlova v Praze)
Radim Jiroušek (Ústav teorie informace a automatizace, AV ČR, Praha)
Zdeněk Kouba (České vysoké učení technické v Praze)
Miroslav Kubát (University of Miami, Coral Gables, Florida)
Marián Mach (Technická univerzita v Košiciach)
Vladimír Mařík (České vysoké učení technické v Praze)
Peter Mikulecký (Univerzita Hradec Králové)
Pavol Návrat (Slovenská technická univerzita v Bratislave)
Richard Papík (Univerzita Karlova v Praze)
Ján Paralič (Technická univerzita v Košiciach)
Michal Pěchouček (České vysoké učení technické v Praze)
Jaroslav Pokorný (Univerzita Karlova v Praze)
Jan Rauch (Vysoká škola ekonomická v Praze)

Vilém Sklenák (Vysoká škola ekonomická v Praze)
Václav Snášel (VŠB–Technická univerzita Ostrava)
Jan Šmíd (Morgan State University, Baltimore, USA)
Vojtěch Svátek (Vysoká škola ekonomická v Praze)
Jana Šarmanová (VŠB–Technická univerzita Ostrava)
Jan Šefránek (Univerzita Komenského, Bratislava)
Olga Štěpánková (České vysoké učení technické v Praze)
Peter Vojtáš (Univerzita Pavla Jozefa Šafárika v Košiciach)
Zdeněk Zdráhal (Open University, Milton Keynes, Velká Británie)
Jaroslav Zendulka (Vysoké učení technické v Brně)
Jana Zvárová (Ústav informatiky, AV ČR, Praha)
Filip Železný (České vysoké učení technické v Praze)
Jan Žižka (Masarykova univerzita v Brně)

Externí recenzenti (External Reviewers)

Jan Blaťák (Masarykova univerzita v Brně)
Martin Hynar (VŠB–Technická univerzita Ostrava)
Jiří Kléma (České vysoké učení technické v Praze)
Martin Labský (Vysoká škola ekonomická v Praze)
Kristína Machová (Technická univerzita v Košiciach)
Kamil Matoušek (České vysoké učení technické v Praze)
Petr Štěpánek (Vysoké učení technické v Brně)

Organizační výbor (Organizing Committee)

Předseda (Chair):
Ján Paralič (Technická univerzita v Košiciach)

Členové (Members):
Petr Bednár (Technická univerzita v Košiciach)
Peter Butka (Technická univerzita v Košiciach)
Ladislav Hluchý (Slovenská akadémia vied, Bratislava)
Martin Sarnovský (Technická univerzita v Košiciach)
Peter Vojtáš (Univerzita Pavla Jozefa Šafárika v Košiciach)

Místo konání (Conference Location):

Hotel Academia, Stará Lesná, Vysoké Tatry
9.–11. února 2005

`http://webocrat.fei.tuke.sk/znalosti2005`
`http://www.cs.vsb.cz/znalosti`

Obsah (Contents)

Tutoriály (Tutorials)

Zvané přednášky (Invited Papers)

Vybrané příspěvky (Contributed Papers)

Dlouhé příspěvky (Long Papers)

Kratké příspěvky (Short Papers)

Data mining

Tomáš Kočka

Adastra s.r.o.
Benešovská 10, 101 00 Praha 10
Czech Republic

TKocka@seznam.cz

Abstract. We will review the applications of data mining from three different perspectives. The first perspective are the business applications themselves across different verticals. We will explain for each of the applications what is its core benefit and how its Return on Investment can be estimated. The applications will range from banking, insurance to telco providers and will cover marketing automation, credit scoring, insurance segmentation, fraud detection and customer value management in general. The second perspective will be statistical and data mining methods. We will present an overview of classification, regression and prediction methods, clustering methods and data description and hypothesis testing methods. We will go to the level of different algorithms like logistic regression, decision trees and rules, support vector machines, neural networks, k-means, expectation maximization and other algorithms. The third perspective is technological, architectural and methodological. We will show how analytical infrastructure fits into standard ERP infrastructure and where is the right place for data mining. We will show different data mining tools and describe their strong and weak points. We will finish by describing typical steps undertaken in a data mining project and highlighting its most critical phases.

EU Research in Information and Communication Technologies and Its Socio-Economic Context

Tomáš Sabol

Faculty of Economics, Technical University of Kosice
Letna 9, 042 00 Kosice, Slovakia

Tomas.Sabol@tuke.sk

Abstract. The paper discusses priority areas of European research in the area of ICT. First, recent trends and challenges are briefly outlined. Then policy priorities identified by the EC for the period 2007-2013, and specifically the three pillars of ICT policy and related implications are presented. Finally, ICT applications targeting achievement of the objectives of European ICT research are briefly listed and the key role of ICTs in economy and society is reiterated.

Keywords: ICT, EU, research, policy, ICT applications.

1 Trends and Challenges

Priorities of applied research should be driven by global social and technology trends and challenges. Actually, we say that success of the applied research is measured by the rate of success by which research outcomes contribute to finding a proper answer to these trends and mastering the main societal challenges. For illustration, some of the recent global trends:

- Globalization – resulting in growing economic integration and new forms of cooperation, but also to increased competition and thus competitiveness is becoming the key concept;
- Ageing of population in Europe - by 2010 the share of population aged 65 and over will have increased by 20% compared with 1995, by 2025 the 65+ age group will comprise 85 million people (of the EU15) and represent 22% of the population (15.4% in 1995). This trend of ageing of population has significant social as well as economic impact (more people needing healthcare, new healthcare models needed, declining rate of the population involved in the economic processes);
- Emergence of knowledge economy - economy in which human resources (knowledge workers) and knowledge are the most important assets of companies.
- Globally growing concern for security issues;
- Growing regional disparities in some countries – see e.g. the new EU members states;
- Accelerating technological development, etc.

Technological development in the area of information and communication technologies (ICT) can be characterized by the following phenomena [1]:

- miniaturisation;

- convergence of computing, communications and media technologies;
- systems able to learn and evolve;
- cross-over between ICT and other science and technology fields.

It is expected that the next wave of technologies will make systems "smaller, cheaper, and smarter" and "always best connected". But before all this comes true an enabling research environment has to be created what requires a comprehensive approach.

2 EU Research Context

The European Commission in its proposal for the EU's financial framework for the period 2007-2013 has identified three policy priorities[2]:

1. "The Internal Market must be completed so that it can play its full part in achieving the broader objective of sustainable development, mobilising economic, social, and environmental policies to that end;
2. The political concept of European citizenship hinges on the completion of an area of freedom, justice, security and access to basic public goods;
3. Europe should project a coherent role as a global partner, inspired by its core values in assuming regional responsibilities, promoting sustainable development, and contributing to civilian and strategic security."

Reaching out for these goals, the EU faces major challenges (see those mentioned above). First of all, higher economic growth is a basic precondition for any economic or social advance. World-wide competition has become more intense. To keep innovating and stay competitive, firms choose different innovation models and organisational forms, including off-shoring and outsourcing ("old" EU15 countries are trying to find a proper answer to this trend, while some new and candidate EU member states can benefit from it – still questionable whether on a temporary or sustainable basis). The race to knowledge, innovation and science and technology is global and has direct impact on the economic performance, growth and prosperity.

In this context, taking into account the global trends listed above, Europe must:

- realise higher economic growth through improving competitiveness and boosting productivity;
- adjust to the changing economic realities brought about by the globalisation of markets and the ever-faster pace of technological change;
- modernise public services – e.g. by means of e-government (the EU public sector spends 45% of the state revenues!);
- come to terms with "greying" European population and its implications;
- pay high attention to security issues (high on the political agenda and in citizens' concerns).

[2] "Building our common Future – Policy challenges and Budgetary means of the Enlarged Union 2007-2013", COM (2004) 101 of 10.02.2004

3 ICT – the Path to Sustainable Growth?

ICT is seen as a key tool to address the three EU policy priorities. The EU ICT policy is built on these three interlinked and mutually reinforcing pillars, where:

- Support for ICT research at EU level helps mobilise the industrial and research communities around high-risk long-term goals, it facilitates the aggregation of public and private research efforts on a European scale;
- Policy initiatives, notably eEurope, promote the wider deployment and adoption of ICT-based products and services in businesses, administrations and public sector services;
- The development of a regulatory framework aims to encourage competition, to improve the functioning of the internal market and to guarantee basic user interests that would not be guaranteed by market forces. The regulatory actions should be simple, aimed at deregulation, technology neutral and sufficiently flexible to deal with fast changing markets.

Most successful are usually countries where the research and innovation effort has gone hand-in-hand with deployment and regulatory initiatives.

It is necessary to emphasize (also for Slovak researchers!) that any investment in ICT research is expected to bring measurable benefits (e.g. measured by the return on investments). According to recent studies, more than half of the productivity (GDP per capita) gains in EU15 economies are attributed to ICT3. In the EU15, of around 1.4% productivity growth between 1995 and 2000, 0.7% was due to ICT. Other evidence suggests that Europe's productivity gap with the US is to a large extent explained by its weaker investment in ICT. The gains stem both from the production of innovative high value ICT-based goods and services as well as from improvements in business processes.

4 ICT Research Areas within EU

The EU's research policy pursues three related and complementary goals4:

- To accelerate the implementation of the European Research Area to create an efficient internal market for research and technological development and a space for a better coordination of national and regional research activities and policies.
- To raise the European effort on research to 3% of EU GDP by 2010 with two-thirds coming from the private sector.
- To support and strengthen excellence in research, development and innovation throughout Europe by providing direct financial support at European level as a complement to national programmes.

---

[3] "The Policy Agenda for Growth" and "The Sources of Economic Growth in OECD Countries", OECD, 2003

[4] "Science and technology, the key to Europe's future – Guidelines for future European Union policy to support research", COM(2004)353 of 16.06.2004

The objectives of European ICT research should be threefold [1]:

1. to strengthen the competitiveness of European industry: by building on strengths and fostering the ability to master ICT for innovation and growth;
2. to reinforce the competitive position of the European ICT sector; and
3. to support EU policies, by mobilising ICT to meet public and societal demands.

ICT applications in the following areas can lead to the fulfilling of these objectives:

- e-Government – ICTs are a key enabler for the modernisation of public services and administrations, they can improve access to and interactivity of existing services, increase their efficiency, deliver completely new customised citizen- and business-friendly services;
- e-Democracy - EU needs active citizens with more direct involvement in public life and the democratic processes;
- e-Business – a response to a call for 24/7 economy, personalised services; enabling 3D-digital mock-ups of complex products, modelling and simulation of product behaviour, smart wireless tags for products, adaptable software for virtual enterprises;
- e-Health - the ageing population results in higher levels of dependency and a rise in demand for health and social care; new generation of “smart” medical sensors and implants enabling advances in clinical diagnosis and treatment; new ICT tools enabling health practitioners to collaborate; remote monitoring allowing care to be delivered at the point of need;
- Knowledge technologies - access to “knowledge anywhere anytime”, new ways of interaction, co-operation, learning; enhancing the capabilities of human beings, individually and collectively, to create and share knowledge;
- Intelligent environments - ICT applications and services that are people-centred, easy to use, enabling users to access and benefit from ICT in a more convenient way (see “ambient intelligence” in FP5);
- Cognitive systems - systems that can contextualise, learn and adapt, have capabilities for perception, communication, reasoning and decision making; can naturally interact with the real world and us, being (self-) adaptive to changing situations and contexts as well as to our own preferences, expertise and needs;
- Security and trust - Europe is exposed to threats and risks from many directions: natural disasters, terrorist threats, cyber-terrorism, spam. ICT allows improved planning and control across the whole risk management chain, protection and security for knowledge, values and digital assets; makes the current infrastructures (e.g. in banking and finance, healthcare, energy, transportation) less vulnerable to failures (whether accidental or malicious);
- Micro-/nano-electronics applications; novel, neuro- and brain-like processing architectures and design of radically new computing and communication components;
- Mobile technologies;
- Networked, embedded and wireless systems providing the broadest bandwidth; ICT systems for integrating and operating a large number of different heterogeneous elements; mastering the complexity of such systems, to provide

them with learning and evolving capabilities and with self-organising and other self-like properties;
- Research and knowledge infrastructures - world class high-speed research networks and computing and knowledge grids;
- e-Inclusion - some parts of the European society feel (and potentially are) marginalized and excluded from the use Information Society services, etc.

5 Conclusions

ICTs have a unique role in economy and society. They are the key to improving productivity; mastering innovation; and essential to modernising public services. If innovation is the engine of the knowledge economy, then ICT are its fuel. The importance of research in ICT is reflected also in research budgets worldwide. ICT represents more than 30% of the total R&D budget in all major OECD countries. Furthermore, the intensity of the research effort in ICT is directly correlated with productivity growth. Within the EU, countries that invest the highest levels in ICT research (Ireland, Finland, Sweden), also have the highest productivity growth rates5. Overall, however, Europe still invests much less in ICT research than its main competitors. For instance, investment in ICT research in the EU is around one third that of the US and is 30% lower than Japan6. The picture is similar for both private and public investment.

However, more money and more research alone is not enough. There are several other key success factors – improved research efficiency and effectiveness; partnering and networking actions that bring together main players around key research and technology developments; measures for intellectual property protection and for standards setting; public financing instruments (direct measures, fiscal incentives, guarantee schemes, support of risk capital).

Reference

1. Improving Competitiveness through Co-operation. Strengthening European Research in Information and Communication Technologies. European Commission. Brussels, 2004.

[5] "ICT and Economic Growth: Evidence from OECD Countries, Industries and Firms", OECD, 2003
[6] "Investment in ICT Research, Comparative Study", IDATE, 2002

The Internet and Corpus Linguistics

James Thomas

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno
Czech Republic

thomas@fi.muni.cz

Abstract. There is a wealth of material available on the Internet for linguists predominantly English-based, though not at all exclusively. Much of it is corpus informed and is of particular interest to people working in a range of linguistics disciplines including lexicography, grammar, stylistics and pedagogy. These resources include research tools and pedagogical applications which are corpus linguistics informed. Some publicly available resources are used also by non-linguists when making data informed decisions, an example of which is academics writing papers in a language that is not their mother tongue. My talk focuses mainly on some of the practical applications of these programs for the general user and in a pedagogical context. Other related exploitations of linguistic data will also be discussed.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 7–7, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

A Scalable Distributed Ontology Repository

Marian Babik and Ladislav Hluchy

Institute of Informatics, Dept. of Parallel and Distributed Computing,
Bratislava, Slovakia

Marian.Babik@saske.sk, hluchy.ui@savba.sk

Abstract. In this paper we present a scalable distributed ontology storage system (OntStore), which unlike other ontology repositories is capable of storing and querying the ontologies and the resource descriptions in a distributed manner. It is based on the distributed hash table (DHT) approach emerging in the P2P arena and well known W3C standardized languages. The system decomposes the ontologies into the corresponding triples (subject, predicate, object) and uses the DHT system called Pastry to hash and store the elements. There are no super-peers in the system and both exact-matched and range queries can be routed. The system ensures, that queries do find the resources if the resources are available in the network. It also guaranties, that the number of routing hops for inserting a resource and resolving most queries is logarithmic to the number of nodes in the network.

1 Introduction

Sharing semantically rich and heterogeneous data is one of the today's major challenges. Publishing and authoring web pages is quite easy, also sharing through the peer-to-peer (P2P) systems makes dissemination of the files a simple job. However there is no similar technology for sharing information with different schemas and providing efficient search and inference capabilities in a distributed manner. Yet, benefits of the semantic data sharing is enormous. Considering scientific data, until recently researchers analyzed and collected data in isolation. With emerging grid technologies and globally scalable systems, integrating and aggregating data from multiple sources and providing efficient search mechanism is necessary. Example of such global scale scientific projects include LHC [1], Human Brain Project, Institute for Systems Biology [2] etc.

At global scale semantic web aims at allowing universal sharing of the semantically rich data by establishing standards for exchanging the machine-understandable information. These standars define both syntactic representation and semantic content of the information. The languages stack, that we use in our work consists of Resource Description Framework (RDF) [3], which provides data model specification; and ontologies modelling languages RDF-Schema [3] and web ontology language (OWL) [4], which enable the definition and sharing of domain vocabularies. Although there are numerous works, that describe distributed file storage systems and file-sharing P2P systems, distributed storage

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 8–17, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

of more expressive infrastructures are not so common. Such infrastructures need to extend the representation of the data and the query functionalities of the file-oriented systems.

This paper describes design and implementation of the schema based infrastructure for the scalable distribution of ontologies based on the P2P distribution model. The underlying the P2P mechanism is based on the distributed hash tables (DHT) [5]. Unlike the hash table, which is a data structure that efficiently maps key/value pairs,the DHT manages distribution of the data among a changing set of servers and allows clients to contact participating server in the network to find resources. The principle of the DHT is same as for the hash tables, to map the resources based on their unique ID key onto the identifier space. We rely on the W3C metadata languages to describe the distributed resources.

2 Design

OntStore is a system for scalable distributed storage of the ontologies represented in the OWL. Considering the fact, that centralized ontology repositories have limitations both in their failure tolerance and their scalability, search for the possible distributed alternatives is indeed inevitable.

OWL is a language for defining and instantiating ontologies. Since OWL depends on constructs defined by RDF, RDFS and XML Schema datatypes (XSD), it is essential to first introduce the languages in order to get a better understanding of the underlying mechanisms.

RDF is a foundation for processing meta data, it provides interoperability between applications that exchange machine-understandable information [3]. The RDF data model describes interrelationships among resources in terms of the named properties and values. The base of the formal model for RDF is a set called rdf:Statement, each element of which is a triple of the form (subject, predicate, object). The subject identifies the resource being described by the modeled statement (e.g. http://www.w3.org/Home/MarianBabik). The predicate identifies the original property in the modeled statement (e.g. http://description.org/ schema/ Email) and the object identifies the property value in the modeled statement (e.g. Marian.Babik@yahoo.com). Object can also be a resource, in this case the mentioned email can be substituted by a reference to an organizational e-mail directory. RDF is based on XML [3] and every resource can be described by a slightly modified Uniform Resource Identifiers (URI) [3] (e.g. http://purl.org /metadata/dublincore#Creator). Objects can also be described by a rdf:Literal, which is lexical form of a unicode string.

The declaration of these properties (attributes) and their corresponding semantics are defined in the context of the RDF as an e RDF schema (RDFS)[3]. A schema defines not only the properties of the resource (e.g., title, author, subject etc.), but may also define the kinds of resources being described (books, Web pages, people etc.). In schemas, new resources can be defined as specialization of old ones, thus allowing to infer implicit triples. Schemas also constrain the context in which defined resources may be used (i.e. validity). Based on first-order

logic those two notions can be seen as one, since they all can be expressed by rules allowing to infer new facts (i.e. new triples)[3]. In the following the 3-nary logical predicate (subject, predicate, object) will be used to represent a believed triple.

XSD represents a standardized definitions for common data types and constraints for the context in which these datatypes can be used [4].

OntStore is a distributed storage for OWL/RDF/RDFS, which provides self-organizing and content addressable network based on the DHT systems. DHT supports location of resources based on their unique key. Our distributed network is based on the DHT system called Pastry [11]. Pastry functions as an overlay on top of the internet protocol (IP), using distributed, fault-tolerant data structure to explicitly track the location of all the resources. This structure is based on the hashed-suffix routing structure presented by Plaxton, Rajaraman and Richa [7]. Pastry is based on the assumption, that nodes and messages can be identified with unique identifiers, represented as strings of digits (e.g. SHA-1 hashing). Identifiers are uniformly distributed in the namespace. Given a message and a key, Pastry reliably routes the message to the node with nodeId that is numerically closest to the key, among all live nodes. In each routing step, the current node normally forwards the message to a node whose nodeId shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the current nodeId. If no such node is found in the routing table, the message is forwarded to a node whose nodeId shares a prefix with key as long as the current node, but numerically closer to the key than the current nodeId. (i.e. 5324 routes to 0629 by `5324 -> ***9 -> **29 -> *629 -> 0629`, * represents wildcards) Fig.1 show the sample routing procedure of the Pastry. Pastry can route to any node in less than $log_{2^b}(N)$, where b is configuration value (usually 4) and N is number of nodes. For extensive description of the design and complexity of the Pastry see [11].

Unlike the other DHT systems Pastry guaranties that the nearest copy of the resource is located. Pastry can be enhanced by exploiting knowledge of the underlying network characteristics [8]. It can also support resilient routing [9] and was successfully implemented as a base overlay system in the global storage system called PAST [12]. The current Pastry distribution has support for parallel insertions, resiliency against massive failures and a distributed test framework.

As described previously the principle construct for the RDF/RDFS is the triple represented as rdf:statement, which consists of three elements, rdf:subject, rdf:predicate and rdf:object. RDF is modeling statements as nodes and arcs in a graph (i.e. RDF graph) [3]. OWL semantics and abstract syntax document [4] describes mapping of the OWL into the RDF graphs. The RDF graph thus can be used as the common component structure to index both languages.

Given a believed triple (subject, predicate, object) (abbrev. (s, p, o)), according to previous description of the Pastry, we can distribute the OWL by computing a hash key for every given element of the triple and storing the triple at three locations corresponding to the hash key of the subject, predicate and object. This way on any given node there will be three stacks of triples sorted by

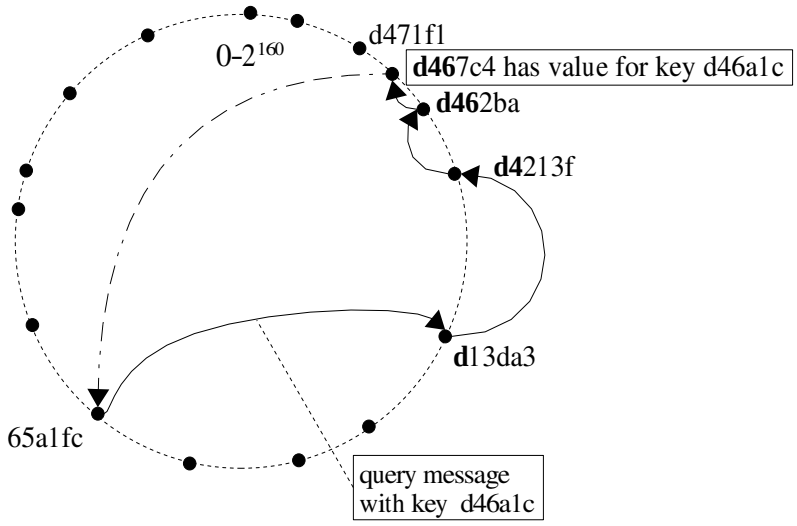


Fig. 1. Routing a message from node 65a1fc with key d46a1c. Figure depicts live nodes in Pastry's circular namespace with base 160.

the subject, predicate and object. For instance, a search for the unknown subject and object, given the predicate (s?, p, o?), will hash the known predicate and locate the peers, which will search their local base and return every triple, which contains the known predicate. Similar approach can be taken to search for any given subject, object or predicate. We can identify seven native query patterns as shown in Table 1.

Table 1. Simple query patterns

No.	Query pattern	Description
Q1	(s?, p, o)	for given object of predicate find all relevant subjects
Q2	(s, p?, o)	find all predicates for given subject and object
Q3	(s, p, o?)	for given subject of predicate find all relevant objects
Q4	(s? ,p?, o)	for given object find all relevant subject/predicate matching triples
Q5	(s?, p, o?)	for given predicate find all subject/object matching triples
Q6	(s, p?, o?)	for given subject find all predicate/object matching triples
Q7	(s, p, o)	return the corresponing triple

There are several issues, which should be noted when indexing the OWL the described way. Decomposition of the OWL into the triples creates a set of simple RDF graphs, which can have anonymous subject or object called anonymous node. This anonymous or also called blank node should be created locally to the mapping and should be unique [4]. Although the unique identification depends on the implementation, it could conflict with the notion of hash based sharing.

Therefore it is necessary to extend the unique ID the way, that it'll be unique not just in the local ontology description, but also in the global context. This can be achieved by creating the prefix with a unique ontology namespace. We'll discuss the implementation issues of the blank nodes in the evaluation section. Another issue is the query resolving, since any native query can return blank node as a subject or an object, it is necessary to track this blank node and return all the relevant triples. This will assure that the system will not return incomplete answers, but will negatively influence the performance by a constant factor reflecting the necessary lookups.

The query language, which defines possible queries will initially enhance the atomic queries described previously by conjunctive, disjuntive and range queries. Conjunctive and disjunctive queries are queries joining the atomic triples by the logical AND or OR respectively. Any conjunctive or disjunctive query will be answered by issuing one query for each atomic pattern and computing intersection or union of the result. Range queries are more difficult, since one needs to find the way to hash the numerical values. This can be solved by using a locality sensitive hashing [19]. LSH can hash the values based on some locality function, which can be based on the distribution of the numerical values. An example of such LSH function is $\mathrm{H}(x) = \frac{(x-x_{min})(2^m-1)}{(x_{max}-x_{min})}$ [18], where m is the base of the hash function (160bit), x_{min}, x_{max} are minimal and maximal values of x.

3 Evaluation and implementation

We have implemented the testing framework for the distributed ontology repository based on the FreePastry [11] and Jena [13]. FreePastry is freely available implementation of the Pastry network providing direct as well as distributed driver for the simulation of the distributed network. Direct driver is based on the Euclidean network and idealized node life. It emulates a network of nodes that are randomly placed in a plane, where proximity is based on euclidean distance in the plane. Distributed driver has two implementations, remote method invocation (RMI) and Wire. RMI is well known remote procedure call library, which is part of the standard $Java^{TM}$ SDK distribution. Wire is an implementation of the event-driven architecture based on sockets, and uses the non-blocking NIO support in java. Jena is a java framework for building semantic web applications. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine.

We have implemented the triple loader with Jena toolkit and used the triple output to publish and query the direct driver of the FreePastry. We have studied number of routing hops to resolve native queries and publish well known food and wine ontology. Food and wine ontology is composed of 2709 triples (8127 requests for storage). Further we have studied the distribution of the triples per node and its possible influence on performance.

The number of routing hops that it takes to route message in the DHT network is important metric of the performance of the system. Fig. 2 shows the number of hops for resolving the native queries Q4-Q6. In the simulation we

have submitted over 1000 queries using randomly choosen node per query. We have simulated queries to up to 1600 nodes. Fig. 2 clearly shows, that routing hops are bounded by the number of nodes in the network. The results shown, closely resembles the results of the pastry evaluation tests [11].

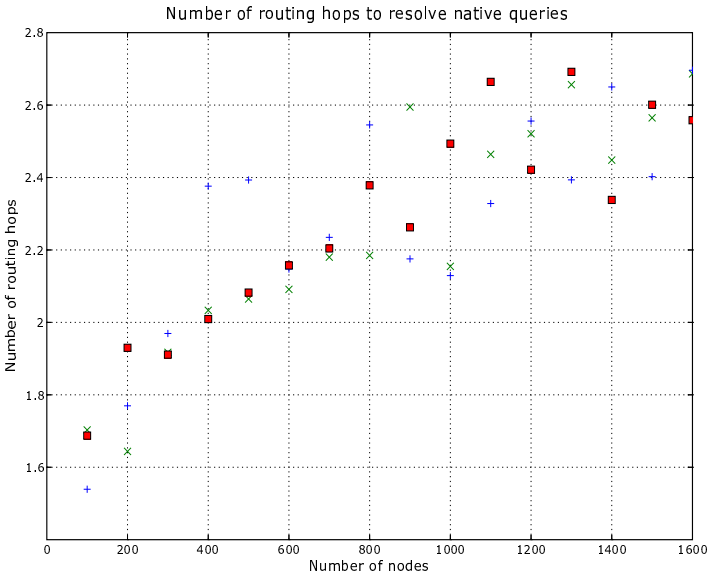


Fig. 2. Number of routing hops needed to answer native queries Q4-Q6. Number of routing hops for queries Q1-Q3 is analogical. Square marks indicate values for Q4, plus marks Q5 and cross marks Q6.

The distribution of the triples over the network is important in understanding the possibility of overloading a node, which will have to store frequently used elements of the triple. It is obvious that popular constructs of the languages used, will have signicant contribution. It's also obvious that the distribution will relate to the hash function used. In the evaluation we have used SHA-1 hash function provided by FreePastry implementation. We have tried to hash URIs by using their original, reversed and relative form. Original form differs from relative in using absolute instead of relative namespaces when hashing URI. Reversed form is a simple reversed string representation of URI. In Fig.3 is the distribution of the subject, predicate and object URIs and literals over the network when hashing the reversed form. Fig.4 shows frequency count of the triples in the ontology and Tab.2 summarizes the most popular constructs.

From both figures it's clear that the most frequent elements are blank node IDs, rdf#type and owl:Restriction. Blank node IDs are created in the Jena by

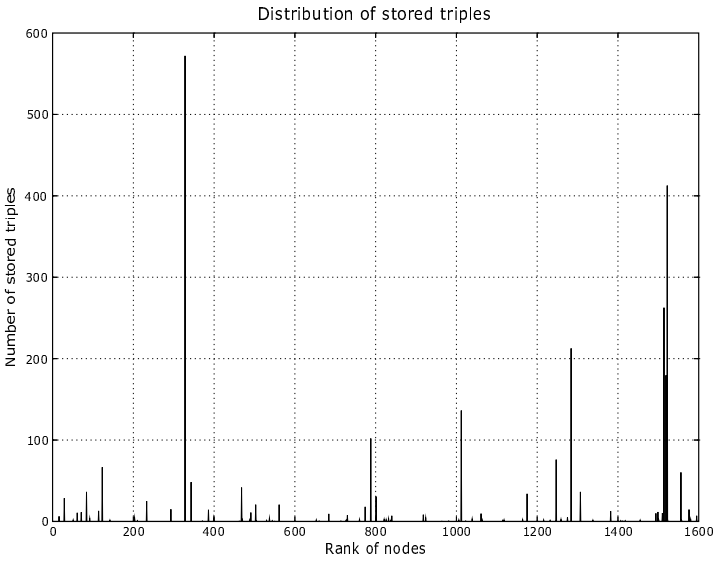


Fig. 3. Distribution of stored triples over the network. Peaks corresponds to the popular URIs and Literals as shown in Table 2.

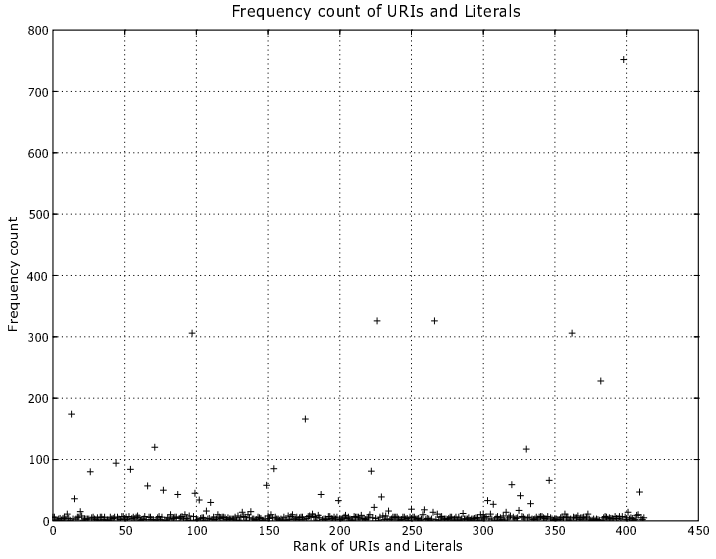


Fig. 4. Distribution of frequency count of the URIs and Literals, except anonymous nodes, in the wine and food ontology.

calling the RMI class UID, which returns identifier that is unique over time with respect to the host it is generated on. Since we have only two ontologies and the simulation is running on one computer the unique ontology identifier doesn't influence the distribution. Having multiple ontologies and several computers can increase the dispersion of the IDs, but even this way for the large ontologies it would not be sufficient. It's obvious that the algorithm needs to assign broader space for unique random ID.

Table 2. Frequency count of the most popular URIs, including count for anonymous nodes.

Freq.count	URI
2319	Anonymous nodes
752	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
326	http://www.w3.org/2002/07/owl#Restriction
306	http://www.w3.org/1999/02/22-rdf-syntax-ns#first
228	http://www.w3.org/2000/01/rdf-schema#subClassOf
174	http://www.w3.org/2002/07/owl#hasValue

Another issue is the frequency of the language constructs. It can be partly solved by simply not indexing them after some threshold [18]. This implies, that queries of type: return all the possible rdf:types in the system, will not be possible. However, since such queries are very general and of rare use this limitations seems reasonable. Another way to cope with the popular terms, would be to find way to uniformly distribute the terms around all the nodes. Such scheme would however need to exploit the neighboring nodes of overloaded node in a way, that would reflect the distribution of terms in use.

4 Related work

There are many centralized RDF repository implementations, such as RDFDB [21], RDFStore [22] and Jena [13]. They usually use files or relational databases as the back-end and some form of the SQL-like query languages. Their limitations come from the fact, that they are centralized, thus having single point of failure as well as limited throughput issues.

Search for the better network topology and efficient routing mechanism inspired by the simple P2P system have become the foundation of the DHT based P2P overlay network, such as CAN [14], Tapestry [6], Chord [15] etc. Although they have different implementation of routing algorithms and overlay network, they tend to use common interface for accessing objects by keys. They provide very good scalability and failure resilience. However they don't support multi-attribute and range queries.e

Edutella [16] provides RDF based meta-data infrastructure for P2P applications, but unlike our system it is based on the super-peer topology and uses

semantic similarity to route the queries in the network. Similar to Edutella in terms of schema-based approach is the Piazza [17] system. Piazza is a peer data management system, which allows the definition of the semantic mappings between pair of peers. Unlike our system it is based on the centralized architecture.

There are proposals for semantic file systems such as HAC [20], which have been designed to support content-based access to file objects in addition to name-based access. There are also P2P file systems such as CFS [23], PAST [12], OceanStore [10], which are layered on top of the DHT. Semantic access mechanism in P2P file systems are proposed by [19]. They are build on top of the DHT and provide semantic indexing and locating, based on the locality sensitive hashing [19].

In Semantic Overlay Network (SON) nodes with semantic similar content are "clustered" together [24]. Queries are routed to the appropriate SONs, increasing the chances that matching files will be found quickly and reducing the search load on nodes that have relevant content. Unlike our system they store files instead of resources and do not provide any sort of the semantic schema based description of the content. RDFPeers [18] is most closely related to our study, since it is a distributed repository of RDF, which extends Chord to efficiently answer multi-attribute and range queries. Unlike our system they aren't considering storage and retrieval of OWL and they use different P2P system.

5 Conclusion

We have described system for scalable distributed ontology repository, which routing and storage costs are logarithmic to the number of nodes in the network. We have performed initial testing and evaluation and discussed the influence of the distribution on the performance. We have evaluated and shown the logarithmic routing complexity of the system. We would like to perform further experiments in querying performance and simulate and optimize the range and conjuctive queries as well as broaden the possibilities of the current query language. We would also like to study possible reasoning implementations, that would infer new facts in a distributed manner.

References

1. The Large Hadron Collider: http://lhc-new-homepage.web.cern.ch/lhc-new-home
2. Institute for Systems Biology: http://www.systemsbiology.org/
3. RDF core WG: RDF Specification, http://www.w3.org/RDF/
4. OWL Web Ontology Language: http://www.w3.org/2001/sw/WebOnt/
5. S. Ratnasamy et al.: Routing Algorithms for DHTs: Some Open Questions. In Proceedings. of 1st International Workshop on Peer-to-Peer Systems, March 2002.
6. B.Y.Zhao, J.D.Kubiatowicz and A.D.Joseph: Tapestry: An infrastructure for fault-resilient wide-area location Technical Report UCB/CSD-01-1141,April 2001
7. C.G. Plaxton, R. Rajaraman and A.W.Richa: Accessing nearby copies of replicated objects in a distributed environment Theory of Computing Systems,241-280,1999

8. B.Y. Zhao et.al.: Brocade: landmark routing on overlay networks 1st Int'l Workshop on P2P Systems (IPTPS),2002
9. B. Y. Zhao et.al.: Exploiting Routing Redundancy via Structured Peer-to-Peer Overlays 11th IEEE Int'l Conference on Network Protocols,2003
10. J.Kubiatowicz et. al: OceanStore: An Architecture for Global-Scale Persistent Storage Proceedings of Ninth ASPLOS 2000
11. Mahajan, M. Castro and A. Rowstron: Controlling the Cost of Reliability in Peer-to-peer Overlays IPTPS'03, Berkeley, CA, February 2003
12. P. Druschel and A. Rowstron: PAST: A large-scale, persistent peer-to-peer storage utility HotOS VIII, Schoss Elmau, Germany, May 2001
13. B. McBride, Jena: Implementing the RDF Model and Syntax specification, In 2nd Int'l Semantic Web Workshop,2001
14. S.Ratnasamy et.al: A scalable content-addressable network In Proc. ACM MOBICOM 2000,2001
15. I. Stoica et.al: Chord: A scalable peer-to-peer lookup service for Internet applications Proceedings of the SIGCOMM 2001,August 2001
16. W.Nejdl et.al.: EDUTELLA: A P2P networking infrastructure based on RDF In 11th World Wide Web Conference,2002
17. I. Tatarinov et.al.: The Piazza peer data management project SIGMOD Record 32(3),2003
18. Min Cai and Martin Frank: RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network Proceedings of the 13th international conference on World Wide Web, 2004, 1-58113-844-X
19. P.Indyk, R. Motwani: Approximate nearest neighbors: towards removing curse of dimensionality In Proc. of 13th annual ACM Symp.,604-613,1998
20. A. Gupta, U. Manber: Integrating content-based access mechanisms with hierarchical file systems In Proc. 3th USENIX (OSDI'99),265-278,1999
21. R.V. Guha: rdfDB: An RDF database http://guha.com/rdfdb
22. RDFStore http://rdfstore.sourceforge.net
23. F. Dabek et.al.: Wide-area cooperative storage with CFS In Proc. ACM SOSP'01,Oct. 2001
24. A.Crespo, H.Garcia-Molina: Semantic Overlay Networks for P2P Systems Technical Report, Stanford University,2003

The ECML/PKDD Discovery Challenge on the Atherosclerosis Risk Factors Data

Petr Berka[1], Marie Tomečková[2], and Jan Rauch[1]

[1]University of Economics, W. Churchill Sq. 4, Praha 3
berka@vse.cz, rauch@vse.cz

[2]Institute of Computer Science, Academy of Sciences,
Pod Vodarenskou vezi 2, Praha 8
tomeckova@euromise.cz

Abstract. It becomes a good habit to organize a data mining cup, a competition or a challenge at machine learning or data mining conferences. The main idea of the Discovery Challenge organized at the European Conferences on Principles and Practice of Knowledge Discovery in Databases since 1999 was to encourage a collaborative research effort rather than a competition between data miners. Different data sets have been used for the challenges during the six years. The paper summarizes our experience gained when organizing and evaluating the challenge on the atherosclerosis risk factor data.

Keywords: atherosclerosis risk, data mining, discovery challenge

1 Introduction

It becomes a good habit to organize a data mining cup, a competition or a challenge at machine learning or data mining conferences. Such events serve several purposes: they can be used for comparison of various approaches and algorithms, they give the participants a possibility to access and analyze real-world data, and they can result in a knowledge interesting for the domain experts who provided the data.

Cups and competitions are usually organized around a clearly specified classification problem. The participants are provided with pre-classified training data and a set of examples to be classified. The goal is to build a model that will perform well on the evaluation data. The models are then ranked according to their performance and the winners (sometimes also the losers) are announced. Thus the first purpose is stressed. Let us mention here e.g. the COIL2000 competition, the EUNITE 2001 or 2002 competition and, of course, the KDD cups held since 1997. Challenges have a less competitive nature. The aim here is to prepare conditions of a real/realistic data mining problem (classification or description) and to find a solution. The results are then discussed with the domain experts. This kind of events is organized e.g. at the European or Pacific-Asian KDD conferences.

The main idea of the Discovery Challenge organized at the European Conferences on Principles and Practice of Knowledge Discovery in Databases since 1999 was to

encourage a collaborative research effort, a broad and unified view of knowledge and methods of discovery, and emphasis on business problems and solutions to those problems. During the six Discovery Challenges organized so far, different data sets have been used. One data set was taken from a financial domain (data about accounts of clients of a bank), the other data sets were taken from various areas of medicine (data about patients with collagen diseases, data about patients with atherosclerosis, and data about patients with hepatitis). Although the data came from very different domains, they shared some common features. The participants were faced with multirelational problem with a mixture of static data (characteristics of clients or patients) and dynamic data (transactions or laboratory tests and examinations).

Our aim was to follow as closely as possible a real KDD process, nevertheless the challenge conditions differed from conditions of a real KDD projects in two main points. The time for analysis was rather short (about two or three months), and the participants have only indirect (if any) access to domain experts.

The rest of the paper summarizes our experience gained when organizing and evaluating the ECML/PKDD Discovery Challenge on atherosclerosis risk factor data.

Table 1. Summary of Discovery Challenge submissions.

year	location	data (#. papers)	sum
1999	Prague	Financial (7) Thrombosis (3)	10
2000	Lyon	Financial (3) Thrombosis (2)	5
2001	Freiburg	Thrombosis (5)	5
2002	Helsinki	Atherosclerosis (5) Hepatitis (5)	10
2003	Cavtat	Atherosclerosis (9) Hepatitis (3)	12
2004	Pisa	Atherosclerosis (11) Hepatitis (5) Genes (2)	18

2 The Atherosclerosis Domain

Atherosclerosis s a total complicated disease of the vessels in all organisms. It is a dynamic process that it begins in childhood and adolescence and continues for the whole life. The experts' opinions on the origin and progress of the disease are developing. Interaction and influence of genetic predisposition and exterior environment as well as of so-called risk factors is considered. On the other hand there are some so-called protective factors.

Among the non-affectable risk factors, crucial are sex, age and family history. The affectable risk factors are factors of life style (e.g. physical activity, smoking, reaction on stress), blood pressure, metabolic factors (level of lipids and glucose), and many more (coagulopathies, infections, inflammation, factors changing the function of endothelium, social and psychological factors).

In the early seventies of the twentieth century, a project of extensive epidemiological study of atherosclerosis primary prevention was developed under the name National Preventive Multifactor Study of Hard Attacks and Strokes in the former Czechoslovakia. The aims of the study were:

1. Identify atherosclerosis risk factors prevalence in a population generally considered to be the most endangered by possible atherosclerosis complications, i.e. middle aged men.
2. Follow the development of these risk factors and their impact on the examined men health, especially with respect to atherosclerotic cardiovascular diseases.
3. Study the impact of complex risk factors intervention on their development and cardiovascular morbidity and mortality.
4. 10–12 years into the study, compare risk factors profile and health of the selected men, who originally did not show any atherosclerosis risk factors with a group of men showing risk factors from the beginning of the study.

Following risk factors were defined at the beginning of the study: arterial hypertension (BP $\geq$ 160/95 mm Hg), cholesterol (level $\geq$ 260mg%) triglycerides (level $\geq$ 200mg%), smoking ($\geq$ 15 cig./day), overweight (Brocka index > 115 %), positive family case history. Later, further laboratory examinations were included: blood sugar level, HDL cholesterol, LDL cholesterol and uric Acid.

The study included data about 1419 men born between 1926–1937 and living in Prague 2. The men were divided according to presence of risk factors (RF), overall health conditions and ECG result into following three groups: normal (a group of men showing no RF defined above), risk (group of men with at least one RF defined above – the prevalence of risk factors for this group is shown in Table 2) and pathological (group of men with a manifested cardio-vascular disease). Long-term observation of patients was based on following the men from normal group and risk group (randomly divided into intervened risk group – RGI and control risk group – RGC). The men from the pathological group were excluded from further observation.

Intervention was the key problem of the study. We tried to optimize and modify influencable RF. Intervention was based on non-pharmacological influence. Pharmacological intervention may be mostly used only in the last years.

1. *non-pharmacological intervention:* interviews on lifestyle, i.e. diet, physical activity, suitability or necessity to stop smoking and reduce weight. The interviews were repeated during each stay and except for general instructions, they focused also around specific RF of a given man.
2. *pharmacological intervention:* treatment of arterial hypertension and hyperlipoproteinemy – was very limited in the initial stages of the study. Pharmacological therapy was recommended with respect to the overall risk of a given man and his possible other diseases.

The regular visits of a doctor themselves could represent an intervention, provided the patient new the reason of the visit, parameters to be followed and desirable parameter values.

3 The STULONG Data and Problem Description

STULONG is the data set concerning the twenty years lasting longitudinal study of the risk factors of the atherosclerosis in the population of 1 417 middle aged men. For the Discovery Challenges, four data files have been used:

1. The file *ENTRY* contains values of 64 attributes obtained from entry examinations; these attributes are either codes or results of measurements of different variables or results of transformations of the rest of the 244 attributes actually surveyed for each patient.
2. Risk factors and clinical demonstration of atherosclerosis have been followed during the control examination for the duration of 20 years. The file *CONTROL* contains results of observation of 66 attributes recorded during these control examinations (10572 records).
3. Additional information about health status of 403 men was collected by the postal questionnaire. Resulting values of 62 attributes are stored in the file *LETTER*.
4. There are 5 attributes concerning death of 389 patients. Values of these attributes are stored in the file *DEATH*.

STULONG data were analyzed using some statistical methods: descriptive statistics, logistic regression and survival analysis. The domain experts were curious about applying data mining methods to this data. Therefore they asked some questions concerning some uncovered relations hidden in the data. The listed analytic questions (possible tasks), which have not been subjected to study yet, can be divided into four groups:

- analytic questions related to the entry examination (what are the relations between social factors, or physical activity, or alcohol consumption and the risk factors),
- analytic questions related to the long-term observation (are there any differences between men of the two risk subgroups RGI, RGC, who came down with the observed cardiovascular diseases in the course of 20 years and those who stayed healthy),
- analytic questions concerning postal questionnaire,
- analytic questions concerning entry examination, long-term observation and death.

Further use of the STULONG data is possible under condition of following explicit quotation: "*The study (STULONG) was realized at the 2nd Department of Medicine, 1st Faculty of Medicine of Charles University and Charles University Hospital, U nemocnice 2, Prague 2 (head. Prof. MUDr M. Aschermann, DrSc), under the supervision of Prof. MUDr. F. Boudík, DrSc, with collaboration of MUDr. M. Tomečková, CSc and Doc. MUDr. J. Bultas, CSc. The data were transferred to the electronic form by the European Centre of Medical Informatics, Statistics and Epidemiology of Charles University and Academy of Sciences (head. Prof. RNDr. J. Zvárová, DrSc). The data resource is on the web pages http://euromise.vse.cz/ challenge2003. At present time the data analysis is supported by the grant of the Ministry of Education CR Nr LN00B107*."

Table 2. Prevalence of risk factors in the risk group.

Risk factor	n	%
hypercholesterolemia	290	34.2
hypertension	287	34.0
smoking	543	63.3
obesity	196	23.0
positive family history	216	25.3

4 Preprocessing and Modeling

The analytic questions given above predetermined both the modeling and preprocessing steps of the analysis. When working with the *ENTRY* table only, the preprocessing was rather simple: to solve problems with numerical attributes and with missing values. When including the *CONTROL* table into the experiments, the preprocessing becomes more complex, as the *CONTROL* table contains results of laboratory tests taken from one patient over time. One possibility how to process these data was just looking if some value occurs in the list of examinations of one patient [1, 7, 14, 22], the other possibility was to process the subsequent results of one particular test as time series and to work with trends [7, 19, 20, 21].

4.1 Descriptive tasks

The descriptive tasks – associations or segmentation (subgroup discovery), are used if the main purpose of the data mining is to find some relation between attributes or examples. As the analytic questions suggest to focus on mining for descriptive models, different forms of association rules were the results of most analyses. Beside "classical" association rules [15], rules and exceptions [13], hypotheses in the sense of GUHA method [8, 11, 18], fuzzy rules [5], sequential rules [7], episode rules [20] or first order rules [6, 26] have been used as well. The way of generating the association rules ranged from a systematic exhaustive way done by rule specialization to random search using genetic algorithms.

The mining for association rules was used to solve the questions concerning relations between characteristics of the patients taken from different data tables. Most simple was the analysis of table *ENTRY* only; this analysis could answer questions concerning relations between different characteristics collected at the entry examination of patients, e.g (from [15]):

`beer(up to 1 liter)` $\Rightarrow$ `vine(daily): conf(0.98).`

More complicated was the analysis of the tables *ENTRY* and *DEATH* together; here, the *DEATH* table was usually used to define a new group of patients to be described using the entry examination, e.g. (from [8]):

`education(university) & height[176-180]` $\Rightarrow$ `death_cause(tumor): conf(0.62)`

Most complicated was the analysis of the tables *ENTRY* and *CONTROL*. Due to the temporal character of the second table, the resulting associations can express statements like "IF the patient regularly consumed alcohol when he entered the study AND his physical activity after job decreased for several control examinations, THEN his cholesterol rate will increase at a control examination taken X months after that period " [7], "IF BMI is quickly decreasing THEN diastolic pressure is decreasing" [19], or " IF the patient has no hypercholesterolemia AND he sometimes follows his diet, THEN the patient will have no hypercholesterolemia within next 40 months" [20].

A step towards better understanding of the resulting associations using automated transformation into natural language sentences was shown in [24]. So e.g. an association rule in the form

`Physical activity after a job(great) &`
`Physical activity in a job (mainly sits)` $\Rightarrow$ `BMI(normal)`

can be automatically transformed into sentences like „X patients confirm this dependence: if a patient has great activity after job and a sedentary job, then s/he has a normal value of BMI" or „a combination of great activity after job and a sedentary job implies a normal value of BMI. This fact is confirmed by X patients".

Interesting concept of emerging patterns has been used in [10, 21]. Emerging patterns are patterns whose frequency increases significantly form one data set to another. This basic idea was used to differentiate between healthy patients and patients with atherosclerosis (the classes were defined using data from the *DEATH* or *CONTROL* tables).

Some papers describe results of clustering based on various patients' characteristics. The clusters can correspond to the original groups (e.g. [12]) or can be defined using the data. Subgroup discovery based on the idea of identification of groups of patients showing significantly different rates of cardiovascular disease (CVD) in comparison with the overall CVD rate is shown in [21], the presence of CVD in this analysis was derived from the table *CONTROL*. Similar definition of clusters has been used also in [25], the aim here was to analyze social characteristics of healthy patients and patients with atherosclerosis.

4.2 Classification tasks

Classification (and regression) tasks are used if the main purpose of the data mining is to build a model that can be used for decision or decision support. The classification tasks performed on STULONG data deal either with classifying patients into the three predefined groups normal, risk or pathological or with classifying them into classes derived form the data. Different approaches have been used also for this type of analyses. [17] used rough set approach to build a set of decision rules to classify patients into classes defined using the tables *CONTROL* and *DEATH* (no disease during follow-up, heart disease during follow-up or as the death cause, other disease during follow-up or as the death cause). In [26], the application of ILP to learn classification rules from the *CONTROL* table is presented; the goal was to predict whether a person in the risk group comes down with a cardiovascular disease or not. Papers [21, 19] (subsequent results of one research group) describe a model that

predicts whether a patient will suffer from CVD in some future based on the values of control examinations taken before.

The classification problem can be eventually turned into a regression problem. A real-valued risk estimate for an individual is proposed in [16] and [1] (the subsequent results of one research group). In the first paper, the global risk estimate is defined as the sum on linear combination of variables related to the personal case history (a static part of the cardiovascular risk) and an exponential function variables related to the personal case history (taking into account cumulative effects of risk factors); the parameters of the function were tuned manually. In the second paper presented next year, genetic algorithm has been used to find optimal parameters for the risk function.

Another example of using genetic algorithm to evolve a discrimination function can be found in [27]. The resulting discrimination function was able to model 8426 (71%) records correctly; anyway the formula was too complicated to be evaluated and interpreted by the domain experts (see Fig. 1).

```
(+ (+ (+ (+ (- (- (- alcohol vzdelani) (- (* (* (+ moc
chlst) (+ kysmoc (+ (+ (* (- dusnost pivo12) (- alcohol
kysmoc)) (- syst1 (- hypll HTD))) (* ldl glykemie))))
(+ (* -3.33355 (* glykemie HT)) (+ (+ (+ (+ imtrv (* -
3.33355 (* glykemie HT))) (+ (* glykemie HT) (+ (+ (-
hypll HTD) (* ldl glykemie)) (* ldl glykemie)))) (-
alcohol vzdelani)) (+ (* ldl glykemie) glykemie)))) (+
(+ (- ICT vinomn) (+ (+ (- (* -3.33355 byvkurak) HT) (*
-3.33355 (* glykemie HT))) hypll)) (* (- vyska HTD) (+
dusnost alcohol))))) HT) (* -3.33355 (* glykemie HT)))
dobakour) (+ (* (+ imtrv (* -3.33355 (* glykemie HT)))
byvkurak) syst2)) (+ (+ (+ vzdelani (+ (* vinomn
byvkurak) smoking)) (* (- dusnost pivo12) (- dusnost
pivo12))) (+ (+ (+ glykemie (* glykemie HT)) (- hypll
HTD)) (* ldl glykemie))))
```

Fig. 1. Discrimination function, taken from [27].

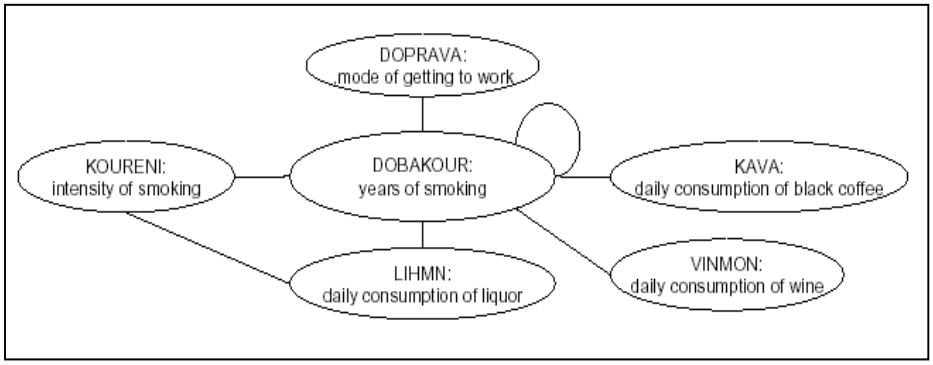


Fig. 2. Network model of unhealthy behaviors of healthy group, taken from [22].

An approach based on co-occurrence matrix and principal component analysis (PCS) is described in [22]. Co-occurrence matrix describes (for one patient) the strength of relations between pairs (rows and columns) of characteristics. The set of these matrices is the used to create (using PCA) so called feature vectors. The feature vector can then be presented in a network model. Fig. 2 shows a network model that describes unhealthy behaviors that healthy patients shared in common, but did not develop any cardiovascular diseases at the end of the study.

5 Evaluation and Deployment

5.1 The data mining expert point of view

It is worth mention, that during the three subsequent Discovery Challenges, the analyses became more elaborated and more complex. In the first year, all papers focus on the analysis of the *ENTRY* table only, and use mainly association rules to find relations between different characteristics of the patients. In the next challenges, data miners used a broader range of data mining methods including rules, exceptional cases, clusters, classification, emerging patterns or linear algebra. Several contributions focussed on mining temporal data (e.g., sequential or episode rules, trend analysis, temporal abstraction, clustering time-series). Some contributions combined several methods and new approaches were proposed. Let us mention e.g. the transformation of association rules into expressions in natural language [24], emerging patterns [10], trend analysis [19], or eigen co-occurrence matrix algorithm [22]. So the Discovery Challenge contributed not only to a specific medical domain but also to data mining and machine learning.

5.2 The domain expert point of view

The data providers gained from the challenge deeper insight into the data. The experts preferred the results of description tasks in favor of the results of classification tasks. Even if the found associations were often no surprise, they were better accepted than the less understandable classification models. The need of understandability of the resulting models can be demonstrated on results shown in Fig. 1 and Fig. 2. Although in both cases rather complicated methods (genetic algorithms, linear algebra) have been used, the graphical representation of the latter was well acknowledged.

Some interesting, sometimes counter-intuitive results obtained in description tasks have been further analyzed and interpreted. As an example let us mention here the positive impact of drinking beer on the health status of the patient (not observed for wine) or the positive influence of number of visits during the follow-up examinations. The former result correlates with the known positive impact of alcohol consumption (Czech men are rather beer drinkers than wine drinkers), the latter result can be interpreted by the fact that during the visits the doctor educates the patient about

healthy life style and also checks the impact of pharmacological intervention on his health status. An interesting useful result was the correlation of Body Mass Index with the skin folds – a very good discrimination of the three basic groups of men (see Fig. 3), or the relation between the level of education and the smoking habits.

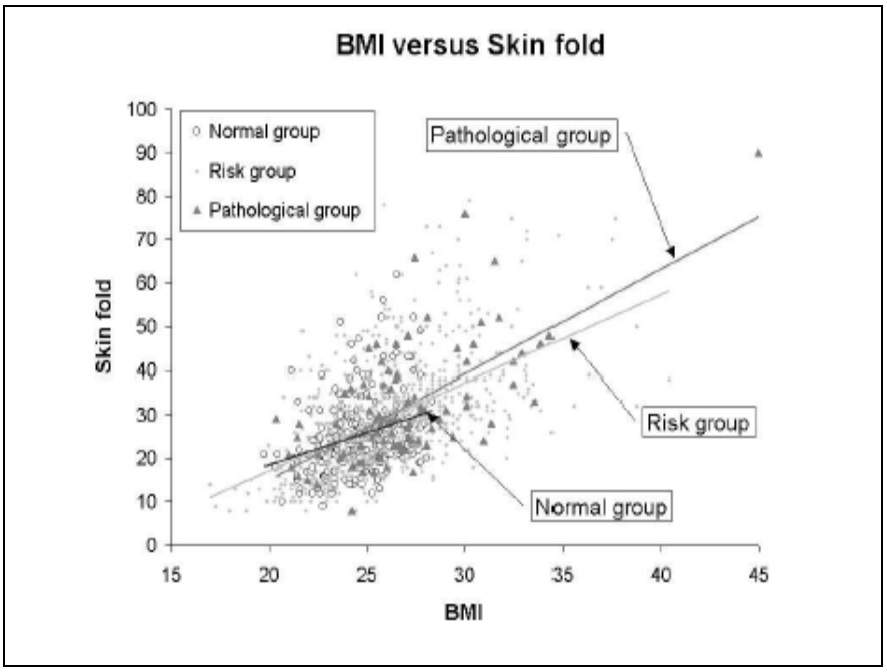


Fig. 3. Correlation between skin fold and BMI for different groups, taken from [26].

6 Conclusions

The reusability of successful data mining solutions can help in new data mining projects. This fact has been recognized by the machine learning and data mining communities. Discovery challenges can provide a workbench for finding such prototype solutions of realistic problems. The interest in the challenge, especially in the year 2004 was likely boosted by a French national research project, that tackles the atherosclerosis data – the working group on Knowledge Discovery Scenarios and Discovery Challenges.

The challenge participants had the opportunity to analyze large real-world data and to test and present their approach. They gained a hands-on experience with realistic data mining projects. Such experience can motivate further research – this was especially the case of groups, that participated in more than one challenge. All papers

presented at the challenge workshops are listed in the references and are available on-line at the Discovery Challenge home-page `http://lisp.vse.cz/challenge`.

The general lessons learned from the series of Discovery Challenge workshops: (1) cooperate with the domain experts, (2) use knowledge-intensive preprocessing, (3) make the results understandable, and (4) show some, even preliminary results soon to the experts, can help in real data mining and knowledge discovery projects.

7 Acknowledgments

The data preparation and data analyses of the STULONG data were supported by the grant of the Ministry of Education CR Nr. LN00B107 (2000-2004). The work presented in the paper is further supported by the grants IGA 17/04 (University of Economics) and GACR 201/05/0325 .

References

1. Azé,J. - Lucas,N. - Sebag,M.: A New Medical Test for Atherosclerosis Detection GeNo. In [3].
2. Berka P., Cremilleux B. (eds.) Discovery Challenge Workshop Notes. ECML/PKDD 2004 15th European Conference on Machine Learning and 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, 2004.
3. Berka P. (ed.) Discovery Challenge Workshop Notes. ECML/PKDD 2003 14th European Conference on Machine Learning and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, 2003.
4. Berka P. (ed.) Discovery Challenge Workshop Notes. ECML/PKDD 2002 13th European Conference on Machine Learning and 6th European Conference on Principles and Practice of Knowledge Discovery in Databases, Helsinki University, 2002.
5. Berzal F., Cubero J.C., Sanchez D., Serrano J.M., Vila M.A. Finding Fuzzy Approximate Dependencies within STULONG Data. In [3].
6. Blatak J. Mining first-order frequent patterns in the STULONG database. In [2].
7. Brisson L., Pasquier N., Collard M., Hébert C. HASAR: mining sequential association rules for atherosclerosis factor analysis. In [2].
8. Burian J., Rauch J. Analysis of Death Causes in the STULONG Data Set. In [3].
9. Couturier O., Delalin H., Fu H., Kouamou G. E., Mephu-Nguifo E. A three-step approach for STULONG database analysis: characterization of patients' groups. In [2].
10. Cremilleux B., Soulet A., Rioult F. Mining the Strongest Emerging Patterns Characterizing Patients Affected by Diseases Due to Atherosclerosis. In [3].
11. Dolejší P., Lín V., Rauch J., Šebek M. System of KDD Tasks and Results within the STULONG Project. In [4].
12. Durand N., Cleuziou G., Soulet A. Discovery of overlapping clusters to detect atherosclerosis risk factors. In [2].
13. Goncalves E.C., Plastino A. Mining strong associations and exceptions in the STULONG data set. In [2].
14. Lachiche N., Fuchs S., Gançarski P. and Derivaux S. Discriminative power of related controls in the STULONG project. In [2].

15. Lin F. Longitudial Study of Atherosclerosis Risk Factors: Relation Factors to BMI and Finding in Entry Examination. In [4].
16. Lucas N., Azé J., Sebag M. Atherosclerosis Risk Identification and Visual Analysis. In [4].
17. Hoa N.S., Son N.H. Analysis of STULONG Data by Rough Set Exploration System (RSES). In [3].
18. Karban T. SDS-Rules and Classification on PKDD2003 Discovery Challenge. In [3].
19. Kléma J., Nováková L., Karel F.,Štěpánková O. Trend analysis in Stulong data. In [2]
20. Meger N., Leschi C., Lucas N., Rigotti C. Mining episode rules in STULONG dataset. In [2].
21. Nováková L., Kléma J., Jakob M., Rawles S., Štěpánková O. Trend Analysis and Risk Identification. In [3].
22. Oka M., Koiso T., Meng E., Kato K. Extracting features of patients using the Eigen co-occurrence matrix algorithm. In [2].
23. Salleb A., Turmeaux T., Vrain C., Nortet C. Mining quantitative association rules in a atherosclerosis dataset. In [2].
24. Strossa P., Rauch J. Association Rules in STULONG and Natural Language. In [4].
25. Soulet A., Hébert C. Using emerging patterns from clusters to characterize social subgroups of patients affected by atherosclerosis. In [2].
26. Van Assche A., Verbaeten S., Krzywania D., Struyf J., Blockeel H. Attribute-Value and First Order Data Mining within the STULONG Project. In [3].
27. Werner J.C., Kalganova T. Risk Evaluation using Evolvable Discriminate Function. In [3].
28. Wettschereck D. Educational Data Preprocessing. In [4].

Mining Relevant Text Documents Using Ranking-Based *k*-NN Algorithms Trained by Only Positive Examples

Jiří Hroza and Jan Žižka

Faculty of Informatics, Department of Information Technologies
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
{xhroza1,zizka}@informatics.muni.cz

Abstract. The problem of mining relevant information from large numbers of unstructured text documents is often handled with various machine learning algorithms trained using both *positive* and *negative* examples that were prepared by an expert in a given specific domain. However, when just *positive* examples are available, the task requires algorithms adapted to the different situation. A modified *k-nearest neighbors* algorithm, trained using only positive examples, can classify by way of *ranking* unlabeled instances depending on their similarity to training examples. This procedure provides a significant part of unlabeled positive instances with high precision. The main objective is to find a method for mining relevant documents from large volumes (hundreds or thousands) of similar medical text files. Experiments and comparisons with various real data obtained from several Internet resources and represented as a *bag of words* provided—under specific conditions—quite acceptable results from the *precision-recall* point of view.

1 Introduction

Mining only relevant information or knowledge from extensive electronic resources belongs among the most desired aims of users working with text documents in various domains. However, enormous volumes of data in database systems, digital libraries, various Internet resources, etc., constitute often a serious problem for human users which—in addition—are usually able to utilize just a very small part of suggested documents. Another difficulty follows from the fact that most of the text documents are unstructured ones in natural languages: proceedings, journals, newspapers, contributions in newsgroups, web pages, and so like.

A typical example is the Internet searching when a user submits a query containing several keywords, possibly joined together with logical connectives, and a browsing system frequently returns hundreds, thousands, or even more answers—either directly text documents or links to them. In such a case, the user is naturally interested in a very limited number of documents that should be as relevant as possible from his or her point of view, even if there is a lot of

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 29–40, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

more or less *relevant* documents because the user usually cannot spend most of the time just reading.

An effort to mine relevant, generally unstructured text documents, was inspired by the demand of physicians that regularly take advantage of special documents downloaded from resources like on-line medical databases provided by the National Library of Medicine, or many fulltext databases accessible to academics and professional healthcare providers (Biological Abstracts, Zoological Record—BIOSIS, digital library of ACM, EIFL Direct, Springer-Verlag, Web of Science, MEDLINE, etc.). When the physicians describe their problems with gathering relevant information from many electronic documents, they very often can provide only positive examples (as they do not store negative ones), which makes the solution more difficult.

2 A Machine Learning Approach

One promising way of mining requested information is *machine learning*. Recently, there are various successful machine learning algorithms applied to the widely required task dealing with separating out text documents. A very good contemporary overview is in [9]. Commonly, machine learning algorithms require training by examples of all considered categories for a given domain. In the most simple case, there are only two classes: *positive (interesting, relevant)* and *negative (uninteresting, irrelevant)* text documents. Algorithms can learn what is relevant and what is irrelevant. Thus, after the first searching step based on more-or-less rough approach, in the second step they are able—with certain errors—to provide desired information.

However, another specific obstacle sometimes emerges: to train a machine learning algorithm, users can often provide just positive examples of documents that are relevant for them, or very unbalanced sets where the number of positive examples highly outweighs the negative ones, which has a negative impact on the training process, and the functionality of the algorithms is not reliable [6]. The reason is that users naturally hold examples that are useful for their work—a typical example is a collection of medical text documents held by physicians-specialists. Thus, there are attempts to modify some algorithms to employ only one (positive) training class, see e.g. [5] about using Support Vector Machines or [7] dealing with Bayes probabilities. In the following sections, this paper describes an alternative solution which is based on the k-nearest neighbors algorithm.

3 *k*-NN for Learning from Positive Examples

The k-nearest neighbors algorithm (k-NN, see e.g. [1]) is an often used method also for the text categorization. Its principle for a multi-class problem is very simple. Unlike many other machine learning algorithms, it principally requires just a trivial training phase: storing a suitable representation of labeled training examples into a database. When a new, unlabeled example occurs, k-NN computes the distance to all known (labeled) patterns. Using the k nearest labeled

patterns, this algorithm chooses the most frequent class for the new example, according to the previous learning. Even if k-NN is a suboptimal procedure, increasing the number of training examples generally leads to lower errors (theoretically approaching the double error rate of the Bayes method [1]). On the other hand, the classification process can take a longer time, especially when k is also high.

The k-NN algorithm cannot be applied directly to the one-class problem. However, there is a possibility to employ a process of *ranking:* if the task is not to directly $classify$ but to $rank$ unlabeled documents according to their relevance, it can be done in the following way. When the distances of unknown examples from the k nearest positive patterns are computed, the resulting value (see Eq. (2)) can be used for sorting the unknown tested examples—nearer unlabeled instances take more front positions from the 'being positive' point of view.

Documents are represented as a *bag of words*, where individual different words create respective dimensions. Therefore, occurrences of words correspond to coordinates of the multidimensional vectors. Using the cosine measure (see Eq. (1) and more details in [1]), which is the inner (scalar) product of vectors and reflects the similarity of vectors representing individual text documents, the positive documents can accumulate in front of the negative ones. In other words—based on the word contents, the used method assigns priorities to documents that are more similar to the positive examples. When a user investigates such documents according to their $rank$, he or she has much higher chance to sooner obtain a proportionate number of truly relevant documents. The learning and ranking algorithms are straightforward (see Table 1 and 2).

Table 1. The learning algorithm

1. After an appropriate preprocessing of data, represent all relevant training documents labeled as *positive* using a *bag of words*— n labeled documents are now multidimensional vectors $\mathbf{v_i}$, $i = 1, \ldots, n$ with coordinates given by individual words.
2. Store the entire set of training documents-vectors into the database.

4 Training and Testing Data

The main aim was mining real unstructured text documents which would include interesting ones from a specific user's point of view. In Table 3, 4, and 5, their basic statistic data are shown. The columns *Word number* quote the average number of words per document. Three different sets of data were used:

Table 2. The ranking algorithm

1. Represent m new unlabeled documents (which can be either relevant or irrelevant to a certain degree) using a *bag of words* from the training phase.
2. For each new unlabeled vector $\mathbf{u_i}$, compute its cosine similarity measure $s(\mathbf{u_i}, \mathbf{v_j})$, i.e., its relevancy degree, to all labeled training vectors (neighbors) $\mathbf{v_j},\ i = 1, \ldots, n,\ j = 1, \ldots, m$:

$$s(\mathbf{u_i}, \mathbf{v_j}) = \frac{{\mathbf{u_i}}^T \mathbf{v_j}}{\|\mathbf{u_i}\|\ \|\mathbf{v_j}\|}, \tag{1}$$

   where ${\mathbf{u_i}}^T$ is transpose of $\mathbf{u_i}$. Two identical vectors have the similarity $s = 1$; without any common coordinate (word), the similarity $s = 0$.
3. For each $\mathbf{u_i}$, select its k nearest neighbors $\mathbf{v_j},\ j = 1, \ldots, k$, where a higher similarity s means a closer distance. Using the k highest similarities s_{kNN}, compute the resulting $\mathbf{u_i}$'s value $w(\mathbf{u_i})$ used for setting up its ranking position:

$$w(\mathbf{u_i}) = \sum_{j=1}^{k} s_{kNN}(\mathbf{u_i}, \mathbf{v_j}) \tag{2}$$

4. According to the w's obtained in the previous step, create the rank of all investigated documents: higher w's mean higher positions in the rank.
5. Within the acquired rank, label the first r vectors as *positive* ones and provide them as *relevant* documents.

– The standard Reuters-21578 dataset [2] that was slightly modified—for the experiments—by removing all one-sentence documents (so-called brief ones, having only the <TITLE> tag and an artificial text-body "Blah blah blah."). Thus, the final number of employed documents slightly decreased from 21,578 to 19,779. In addition to the numbers of documents in Table 3, there were also 10,167 documents from the remaining topics—this was used as the negative instances, too.
– Commonly employed 20 topically more or less different newsgroups [3] from Internet conferences. Each of the newsgroups had 990 examples and represented one topic.
– Special medical data GLALL [4] that comprise selected documents obtained from the Internet resources. These documents were separated by an expert physician into two subgroups: positive examples and—in this case, for testing the ranking algorithm—negative ones from the same special medical domain. The GLALL positive and negative medical documents were generally very

similar from a human point of view, so mining relevant documents was not easy also for humans.

The baseline for Reuters topics was between 0.01 and 0.19, for 20 Newsgroups 0.05, and for GLALL 0.58. The mining procedure tried to imitate typical situations described by physicians. In the experiments with the Reuters dataset, ten topics having the highest number of documents were used as the relevant classes. From these ten topics, each of them was successively chosen as a positive class and the remaining nine topics together with the rest of all data acted as negative instances for testing. Similarly, from the 20 newsgroups data, each of the topics was in rotation used as a relevant class and the 19 remaining topics as an irrelevant class. The GLALL documents consisted of a positive and negative class, so using this data was straightforward.

Table 3. Reuters basic statistics

Document ID	Topic label	Document number	Word number
1	earn	3799	93
2	acq	2213	143
3	money-fx	689	232
4	grain	580	207
5	crude	573	245
6	trade	517	283
7	interest	426	211
8	ship	303	181
9	wheat	288	201
10	corn	224	243

5 Data Preprocessing

In all the experiments, simple preprocessing based on *stemming* was used. To obtain a stem of each word, the commonly used Porter's stemming algorithm [8] of the suffix stripping was applied. This step is commonly used and one of its advantages is decreasing the number of many dimensions. After this stage, a dictionary could be created as a bag of words from the original training examples. Removing the most common words (stop words) from the bag of words was tested as well. In one part of tests, the stop-list containing the first 100, 200, and 300 English stop words was used (e.g., the first five stop words were *the, of, and, a, to, ...*). Then, using the dictionary, all the appropriate documents were encoded into vectors of numbers. Every position in a vector corresponded to a certain word in the dictionary. Two types of encoding were employed. The first straightforward method used the binary tag '1' in a position of the vector if

Table 4. 20 Newsgroups basic statistics

Doc. ID	Topic label	Doc. num.	Word num.	Doc. ID	Topic label	Doc. num.	Word num.
1	alt.atheism	990	323	11	rec.sport.hockey	990	280
2	comp.graphics	990	257	12	sci.crypt	990	319
3	comp.os.ms-windows.misc	990	484	13	sci.electronics	990	193
4	comp.sys.ibm.pc.hardware	990	190	14	sci.med	990	288
5	comp.sys.mac.hardware	990	173	15	sci.space	990	281
6	comp.windows.x	990	304	16	soc.religion.christian	990	383
7	misc.forsale	990	133	17	talk.politics.guns	990	330
8	rec.autos	990	208	18	talk.politics.mideast	990	490
9	rec.motorcycles	990	186	19	talk.politics.misc	990	402
10	rec.sport.baseball	990	222	20	talk.religion.misc	990	327

Table 5. GLALL basic statistics.

Document ID	Topic label	Document number	Word number
1	interesting	406	128
2	uninteresting	295	107

the corresponding word was present in the document; otherwise, '0'. The second method used a word-frequency in a document to give more information to the k-NN algorithm. Essentially, these frequencies played a role of word weights.

6 Description and Results of Experiments

In the experiments, there were employed two algorithms. The first one was k-NN trained only by positive examples as explained in Section 3. The second one was the standard naïve Bayes classifier, see [6], which was used for the comparison with results obtained by k-NN. This classifier was trained using both positive and negative examples. Each algorithm used the 10-fold cross-validation, where the training phase employed one fold, while the testing phase the remainder. The reason was that the user usually has a few of the positive examples and he or she needs to mine additional interesting documents from a large number of unlabeled instances provided by, for example, a WWW search engine.

Evaluation of the algorithms' success was done in the following way. After the ranking and subsequent sorting of the test documents, there should be relevant documents at the first positions, and irrelevant ones at the last positions. In the graphs, the effectiveness of algorithms is presented using the *precision* dependence on *recall* [10]:

$$precision = \frac{number\ of\ identified\ positive\ items}{total\ number\ of\ assigned\ items} \tag{3}$$

$$recall = \frac{number\ of\ identified\ positive\ items}{total\ number\ of\ positive\ items} \tag{4}$$

For example, in the graph in Fig. 1 if one is interested in documents represented using word-frequency without removing stop-words (*freq,0*), he or she can see that the recall value 0.5 predicts the precision around 0.8 for positive instances. The other method used for comparing results is so-called F_1-*measures*, which combine precisions and recalls according to the definition in [10]:

$$F_1(R, P) = \frac{2RP}{R + P}, \tag{5}$$

where R stands for the *recall* and P stands for the *precision*. In the experiments described in this paper, the $F1$ measure was computed using the best pair of the recall value and its corresponding precision.

6.1 Ranking the Reuters Dataset

The following graphs Fig. 1 and 2 are samples of the ranking procedure applied to two selected topics from the Reuters dataset. Obviously, the 5-NN algorithm rearranges the unlabeled instances so the more relevant ones get higher priorities, which helps the user to obtain them.

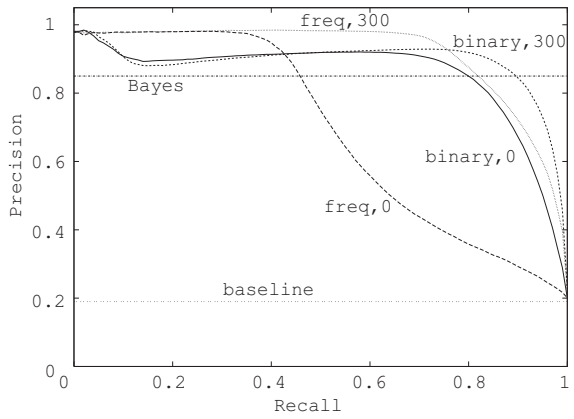


Fig. 1. The Reuters ID No. 1 for 5-NN. The dependence precision-recall for different word representations (frequency, binary) and stop-word number (0 and 300). For the comparison of the results, the baseline and Bayes are provided

The average values in Table 6 show the precision for the first 10 and 50 documents with the highest priority determined by the user. The precision depends on k nearest neighbors, on the representation of words (b stands for *binary*, f stands for $frequency$), and on the number of stop-words. The numbers of documents were selected 10 and 50 because users usually prefer not to read too many documents, i.e., not more than, for example, ten; certainly not more than

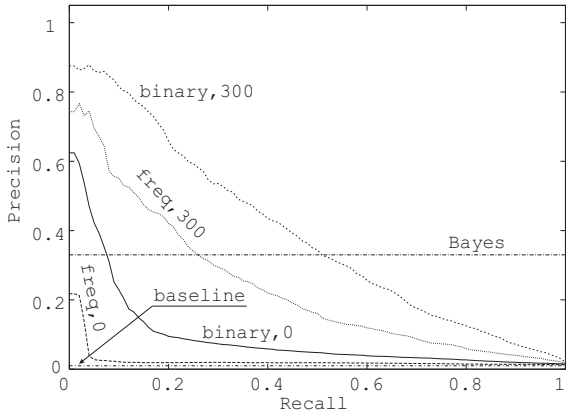


Fig. 2. The Reuters ID No. 10 for 5-NN. The dependence precision-recall for different word representations (frequency, binary) and stop-word number (0 and 300). For the comparison of the results, the baseline and Bayes are provided

fifty. It is possible to notice the differences between 10 and 50. In the experiments, there were used also 20, 30, and 40 documents, however, the precision decreased approximately linearly. Obviously, the frequency representation together with using at least 100 stop-words provided better results. However, it is difficult to say generally whether 100, 200, or 300 stop-words should be employed. Regarding the number of neighbors, the best results were obtained for 5-NN, the frequency encoding and a nonempty list of stop-words. Table 7 compares the $F1$ measure between the naïve Bayes and the best k-NN over all topics in the Reuters data. According to the expectation, the naïve Bayes learning from both positive and negative data achieved in this case better results than k-NN learning only from positive examples. On average, the $F1$ measure for Bayes was 0.55 while for k-NN it was 0.44.

6.2 Ranking the Newsgroups

The following two tables demonstrate results obtained using other dataset. For the 20 Newsgroups, the results depicted in Table 10 are very similar to the Reuters data. Moreover, in the case illustrated by Table 9, Bayes provided even much better results (0.90) on average than k-NN (0.36) that was inferior.

6.3 Ranking the Medical Dataset GLALL

Unlike the previous data, the medical dataset GLALL gave rather different outcomes. The number of nearest neighbors played the main role, and the best results were obtained for 1-NN. Compared with the results for other datasets, 5-NN provided the worst results. In addition, the dependence on stop-words as well as on encoding was quite negligible. Surprisingly, in the case of GLALL, k-NN's $F1$ measure was slightly better: 0.73 in comparison with 0.68 for Bayes. This result is in Table 11.

7 Conclusions

The experiments with real data from three various domains revealed that users can successfully employ even a relatively simple algorithm as k-NN trained only by *positive* examples. If a user is interested only in a small part of relevant, however as significant as possible data, this algorithm is capable to mine the requested text documents with very high precision. The best achieved precision values, where the requested number of documents was 10, were 88% for the standard Reuters dataset, 87% for the 20 newsgroups, and 93% for the special medical dataset GLALL. In addition, GLALL had significantly lower decrease of precision (from 0.93 to 0.68) than the other datasets: 0.88 to 0.33 for Reuters, and 0.87 to 0.11 for 20 Newsgroups; the reason is probably in the higher GLALL's baseline.

The k-NN's outcomes were compared with the naïve Bayes classifier, which expectedly gave mostly better results, except for GLALL. On the other hand, Bayes requests both positive and negative examples for its training. The remaining problem is that it is not easy to predict which algorithm and its parameters can provide the best results because the results were rather different. Testing with more real data could bring more accurate answers, and this is the aim of the future work.

8 Acknowledgments

This research was partly supported by the Grant *Human-Computer Interaction, Dialog Systems and Assistive Technologies, MSM*-143300003.

References

1. Duda, R. O., Hart, P. E., and Stork, D. G. (2001): Pattern Classification. Second Edition, John Wiley & Sons, 2001
2. `http://www.research.att.com/~lewis`
3. `http://www.ai.mit.edu/~jrennie/20Newsgroups/`
4. `http://www.fi.muni.cz/~xhroza1/datasets/glall/`
5. Manevitz, L. R. and Yousef, M. (2001): One-Class SVMs for Document Classification. In: Journal of Machine Learning Research 2 (2001), December 2001, pp. 139-154
6. Mitchell, T. M. (1997): Machine Learning. McGraw Hill, 1997
7. Muggleton, S. (1996): Learning from Positive Data. In: S. Muggleton (Ed.): Proceedings of the 6th International Workshop on Inductive Logic Programming, Springer-Verlag, LNAI-1314, 1996, pp. 358-376
8. Porter, M. F. (1980): An Algorithm For Suffix Stripping. Program 14 (3), 1980, pp. 130-137
9. Sebastiani, F. (2002): Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, No. 1, March 2002, pp. 1-47
10. Van Rijsbergen, C. J. (1979): Information Retrieval, 2nd Edition. Department of Computer Science, University of Glasgow, 1979

Table 6. The precision values for the Reuters dataset based on different settings for the neighbor number k, frequency (f) or binary (b) encoding, and number of stop-words

For the first 10 documents				For the first 50 documents			
Precision value	Neighbor number k	Encoding f or b	Stop-word number	Precision value	Neighbor number k	Encoding f or b	Stop-word number
0.88	5	f	200	0.77	5	f	100
0.88	5	f	300	0.76	5	f	300
0.87	5	f	100	0.76	5	f	200
0.86	1	f	100	0.75	3	f	100
0.86	1	f	200	0.74	3	f	200
0.85	1	f	300	0.74	3	f	300
0.85	3	f	100	0.70	1	f	300
0.85	1	f	0	0.69	1	f	100
0.84	3	f	200	0.69	1	f	200
0.84	3	f	300	0.58	1	f	0
0.80	1	b	300	0.44	1	b	300
0.80	1	b	200	0.44	1	b	0
0.79	1	b	100	0.43	1	b	200
0.78	1	b	0	0.42	1	b	100
0.58	3	f	0	0.40	3	f	0
0.47	3	b	0	0.33	5	f	0
0.46	5	f	0	0.32	3	b	0
0.42	3	b	100	0.28	3	b	300
0.41	3	b	300	0.28	5	b	0
0.41	3	b	200	0.27	3	b	200
0.40	5	b	0	0.27	3	b	100
0.34	5	b	300	0.23	5	b	300
0.34	5	b	200	0.22	5	b	200
0.33	5	b	100	0.21	5	b	100

Table 7. The comparison of the F1-measure for Reuters

Document ID	F1-measure for Bayes	F1-measure for k-NN
1	0.85	0.83
2	0.63	0.36
3	0.63	0.44
4	0.56	0.34
5	0.65	0.46
6	0.48	0.35
7	0.53	0.55
8	0.47	0.39
9	0.42	0.40
10	0.33	0.29

Table 8. The precision values for the 20 Newsgroups dataset based on different settings for the neighbor number k, frequency (f) or binary (b) encoding, and number of stop-words

For the first 10 documents				For the first 50 documents			
Precision value	Neighbor number k	Encoding f or b	Stop-word number	Precision value	Neighbor number k	Encoding f or b	Stop-word number
0.87	5	f	100	0.76	5	f	200
0.84	5	f	200	0.75	5	f	300
0.83	5	f	300	0.75	5	f	100
0.81	3	f	100	0.71	3	f	100
0.76	3	f	200	0.70	3	f	200
0.73	3	f	300	0.68	3	f	300
0.59	1	f	100	0.58	1	f	100
0.54	5	f	0	0.51	1	f	200
0.53	3	f	0	0.49	1	f	300
0.51	1	f	0	0.34	5	f	0
0.50	1	f	200	0.34	3	f	0
0.47	1	f	300	0.33	1	f	0
0.33	1	b	300	0.13	1	b	300
0.32	1	b	200	0.13	1	b	200
0.31	1	b	100	0.12	3	b	300
0.28	3	b	100	0.12	3	b	200
0.25	3	b	200	0.11	5	b	300
0.23	3	b	300	0.11	5	b	200
0.21	1	b	0	0.09	1	b	100
0.18	5	b	100	0.09	3	b	100
0.17	3	b	0	0.09	5	b	100
0.16	5	b	200	0.07	1	b	0
0.14	5	b	300	0.06	3	b	0
0.11	5	b	0	0.06	5	b	0

Table 9. The comparison of the F1-measure for 20 Newsgroups

Document ID	F1-measure for Bayes	F1-measure for k-NN	Document ID	F1-measure for Bayes	F1-measure for k-NN
1	0.91	0.33	11	0.97	0.52
2	0.91	0.41	12	0.94	0.41
3	0.97	0.94	13	0.70	0.14
4	0.86	0.28	14	0.94	0.28
5	0.82	0.21	15	0.93	0.26
6	0.95	0.37	16	0.94	0.44
7	0.85	0.27	17	0.91	0.31
8	0.86	0.25	18	0.97	0.51
9	0.86	0.24	19	0.91	0.28
10	0.93	0.33	20	0.87	0.28

Table 10. The precision values for the GLALL dataset based on different settings for the neighbor number k, frequency (f) or binary (b) encoding, and number of stop-words

For the first 10 documents				For the first 50 documents			
Precision value	Neighbor number k	Encoding f or b	Stop-word number	Precision value	Neighbor number k	Encoding f or b	Stop-word number
0.93	1	b	300	0.79	1	f	0
0.92	1	b	200	0.77	1	f	100
0.92	1	b	100	0.75	1	f	300
0.92	1	b	0	0.75	1	f	200
0.90	1	f	300	0.73	3	f	0
0.90	1	f	200	0.73	1	b	200
0.90	1	f	100	0.73	1	b	100
0.90	1	f	0	0.73	1	b	0
0.84	3	f	0	0.72	5	f	0
0.83	3	b	0	0.72	3	f	100
0.81	5	f	0	0.72	1	b	300
0.81	3	b	100	0.71	3	f	200
0.80	3	b	300	0.70	5	f	100
0.80	3	b	200	0.70	3	f	300
0.79	5	f	300	0.68	5	f	200
0.79	3	f	200	0.67	5	f	300
0.78	5	f	200	0.64	3	b	0
0.78	5	f	100	0.63	3	b	200
0.78	3	f	300	0.62	3	b	100
0.78	3	f	100	0.61	5	b	0
0.75	5	b	0	0.61	3	b	300
0.72	5	b	100	0.59	5	b	300
0.69	5	b	300	0.58	5	b	200
0.68	5	b	200	0.58	5	b	100

Table 11. The comparison of the F1-measure for GLALL

Document ID	F1-measure for Bayes	F1-measure for k-NN
1	0.68	0.73

Pseudo-dělení binárních matic a jeho aplikace

Aleš Keprt, Václav Snášel

Katedra informatiky, Fakulta elektrotechniky a informatiky
VŠB Technická Univerzita Ostrava
17. listopadu 15, 708 33, Ostrava-Poruba, Česká republika
Ales@Keprt.cz, Vaclav.Snasel@vsb.cz

Abstrakt. Úspěšnost každého klíčového algoritmu závisí také na mnoha dalších podpůrných algoritmech. Ukázalo se, že problém redukce dimenze binárního prostoru pomocí faktorového rozkladu binárních matic (např. redukce dokument-term incidenční matice) je hodně závislý na možnosti provádět pseudo-dělení obecných binárních matic. Příspěvek představuje řešení tohoto problému.

Klíčová slova: binární matice, pseudo-dělení, binární faktorová analýza, složitost

1 Úvod

Úspěšnost každého klíčového algoritmu závisí také na mnoha dalších podpůrných algoritmech. Ukázalo se, že problém redukce dimenze binárního prostoru pomocí faktorového rozkladu binárních matic (např. redukce dokument-term incidenční matice) je hodně závislý na možnosti provádět pseudo-dělení obecných binárních matic.

Binární prostory jsou obdobou vektorových prostorů, jejich studium a různé druhy analýz binárních dat (binárních analýz) často nemohou používat známé algoritmy lineární algebry, neboť binární data vyžadují pro dosažení smysluplných výsledků analýzy specifický přístup. Jedním z příkladů je binární faktorová analýza (BFA), což je analýza snažící se o vyjádření skrytých vztahů mezi zkoumanými objekty/veličinami tím způsobem, že tyto rozdělí do libovolně se překrývajících shluků vždy obsahujících podobné nebo podobně se chovající objekty. (Více o BFA viz [1], [2], [3], [4].)

BFA je čistě binární metodou používající výhradně booleovských operací (které je možno vyjádřit určitou kombinací booleovského součtu, součinu a negace). Není tedy lineární, a proto pro její výpočet nelze použít běžné faktorizační metody založené na lineární algebře.

BFA se také odlišuje od jiných binárních shlukovacích metod tím, že umožňuje libovolné překrývání shluků, čímž umožňuje dosáhnout mnohem lepších výsledků než jiné binární nebo dokonce lineární metody.

Výpočet BFA stojí na provádění velkého množství operací na binárních číslech, binárních vektorech a binárních maticích. Jednou z problematických operací, které se dříve BFA algoritmy snažily vyhnout, je binární maticové pseudo-dělení, kdy se snažíme najít jeden z činitelů při znalosti druhého činitele a výsledného součinu. Právě na řešení tohoto problému je zaměřen tento příspěvek.

2 Pseudo-dělení binárních matic

Jak bude předvedeno v další kapitole, pseudo-dělení binárních matic je u mnoha algoritmů pro řešení BFA důležitou, někdy i klíčovou operací, neboť nám především umožňuje stanovit kvalitu dílčího či kandidátního řešení rozkladu binární matice na booleovský součin dvou binárních matic.

2.1 Použitá notace

V textu se budeme pohybovat v oblasti binární algebry. Všechny matice jsou tedy binární, tj. obsahují jen nuly a jedničky. Operace na maticích, vektorech i skalárech jsou binární (booleovské). Operace „násobení“ $\odot$ a „součet“ $\oplus$ tedy chápeme booleovsky, na počítači jim odpovídají `AND` a `OR`. Při omezení na pouze binární hodnoty je pak u těchto operací jediný rozdíl oproti klasické aritmetice: $1 \oplus 1 = 1$. Viz tabulka 1.

Tabulka 1. Binární operátory.

$\odot$	0	1
0	0	0
1	0	1

$\oplus$	0	1
0	0	1
1	1	1

Pro zpřehlednění vzorců a algoritmů používáme zjednodušenou notaci všude tam, kde nemůže dojít k záměně významu. Například malým a s dvěma indexy a_{ij} automaticky chápeme prvek na pozici řádku i a sloupci j matice označené velkým písmenem A. malým a_i s jedním indexem značíme řádkový vektor matice A. Všechny indexy jsou číslovány od jedničky. Binární operace kombinujeme i s klasickou aritmetikou všude tam, kde je to vhodné pro lepší srozumitelnost.

V situacích, kde by k záměně významu dojít mohlo, dáváme na notaci velký pozor. Například rozlišujeme klasickou sumu $\sum$, která všechny argumenty sečte klasickým součtem $+$, a binární sumu $\bigvee$, kde argumenty sčítáme binárně operátorem $\oplus$.

2.2 Definice problému pseudo-dělení

Je dána matice X typu $n \times p$ a matice A typu $m \times p$. Hledáme matici F typu $n \times m$ tak, aby platila přibližná rovnost

$$\mathsf{X}_{[n\times p]} \approx \mathsf{F}_{[n\times m]} \odot \mathsf{A}_{[m\times p]}$$

Přitom hledáme takovou matici F, aby odchylka daná hodnotou funkce discrepancy d, udávající počet rozdílných bitů mezi součinem $\mathsf{F} \odot \mathsf{A}$ a datovou maticí X, byla co nejmenší.

$$\hat{\mathsf{X}} = \mathsf{F} \odot \mathsf{A}$$

$$d = \sum_{i=1}^{n} \sum_{j=1}^{p} |\hat{x}_{ij} - x_{ij}|$$

Jelikož maticový součin $\mathsf{F} \odot \mathsf{A}$ není komutativní operací, je vhodné poznamenat, že hledáme levou matici součinu F, zatímco pravou matici A známe.

2.3 Naivní řešení

Nejjednodušším algoritmem je postupné zkoušení všech možností. Práce zabývající se binární faktorovou analýzou kdysi skutečně byly omezeny na toto slepé hledání matice F, časová výpočetní složitost je však řádu 2^N. To omezuje praktickou použitelnost tohoto algoritmu na velmi malé matice.

2.4 Výpočet po řádcích

Výpočet po řádcích byl poprvé představen v práci [3]. Začneme rozepsáním vzorce pro násobení matic.

$$x_{ij} = \bigvee_{k=1}^{m} (f_{ik} \odot a_{kj}) \tag{1}$$

Vyjádříme-li matice pomocí řádkových vektorů $\mathsf{X} = [x_1, \ldots, x_n]$ a $\mathsf{A} = [a_1, \ldots, a_m]$, můžeme vzorec 1 přepsat takto

$$x_i = \bigvee_{k=1}^{m} (f_{ik} \odot a_k) \tag{2}$$

Podobně můžeme odchylku d vyjádřit jako sumu řádkových odchylek

$$d = \sum_{i=1}^{n} d_i$$

Z vzorce 2 je vidět, že pro výpočet i-tého řádku matice X potřebujeme znát jen i-tý řádek matice F. To samozřejmě platí i opačně: Matici F můžeme hledat po řádcích s využitím vždy jen jednoho odpovídajícího řádku matice X. Postupně tedy do každého řádku dosazujeme všechny hodnoty (bitově zakódované do čísel 0 až $2^m - 1$) a počítáme odchylku d_i pro daný řádek i.

2.5 Výpočet řádku zatřiďováním

Dále se zaměříme na výpočet jednotlivého řádkového vektoru f_i, zde popsaný postup tedy opakujeme pro všechna $i \in [1, n]$. Klíčem úspěchu je vhodná interpretace vzorce 2. Každý řádek x_i musíme chápat jako binární vektorový součet nějaké podmnožiny řádků z A. To, zda konkrétní řádek a_k je v této podmnožině obsažen, je určeno hodnotou bitu f_{ik}. Naším cílem je co nejrychleji najít tuto podmnožinu, čímž vypočteme hodnotu celého řádkového vektoru f_i.

Nejprve pro každý řádek a_k spočítáme počet odchýlených bitů, tj. těch, pro které při obsažení a_k v hledané podmnožině platí jeden z těchto dvou vztahů:
Pozitivní odchylka: $a_{kj} = 0$ a $x_{ij} = 1$
Negativní odchylka: $a_{kj} = 1$ a $x_{ij} = 0$
Pozitivní a negativní odchylky počítáme vždy jako sumu všech odchylek stejného typu pro řádkové vektory x_i a a_k.

Negativní odchylka je velmi nepříjemná; způsobuje, že jedničkový bit, který se takto generuje tam, kde být nemá, již nelze nijak odstranit. Binární součet jedničky s libovolným počtem libovolných dalších hodnot je vždy roven jedné.

Naopak pozitivní odchylka nepůsobí problémy, neboť chybějící jedničku můžeme do bitu x_{ij} dosadit přidáním dalšího řádku z A.

Velmi výhodné je využít paralelních booleovských výpočetních instrukcí, které obsahují úplně všechny mikroprocesory a jsou podporovány i ve vyšších programovacích jazycích. Jedná se o klasické operace `and`, `or`, `not` a `xor` nad celočíselnými datovými typy (integer). Máme-li matice v paměti uloženy po řádcích tak, že po sobě jdoucí bity na řádku matice odpovídají po sobě jdoucím bitům v paměti počítače, můžeme výpočet výrazně urychlit. (Pořadí uložení maticových řádků v paměti počítače však rozhodující není.) Například běžné procesory typu x86 umožňují snadno počítat se 32 bity. Náš algoritmus je proto záměrně navržen tak, aby všechny jeho části pracovaly s řádkovými vektory o délce 32 bitů, přičemž využití všech těchto 32 bitů není povinné (pro vektory kratší než 32 bitů zůstávají ostatní paměťové bity vynulované). Kód v pseudo-jazyku používá klasické počítačové značení řádkových vektorů (`X[i]` je vektor x_i, `A[k]` je vektor a_k, atp.) a operací mezi nimi.

```
Pro všechny řádky A[k] {
  match = A[k] AND X[i]
  if(match==0) {
    //řádky nemají žádné odpovídající jedničkové bity
    //tento řádek rovnou zahodíme (není potřeba nikde pamatovat)
  } else {
    //match nyní obsahuje odpovídající si jedničkové bity
    //dále zjistíme, jakou odchylku vytvářejí ostatní bity
    goodbits = počet_nenulových_bitů(match)
    negative_d = počet_nenulových_bitů(A[k] AND NOT X[i])

    if(negative_d==0) {
      //negativní odchylka je nulová --> tento řádek určitě použijeme
      přidej A[k] mezi vybrané řádky
      goodsum = goodsum OR A[k]
    } else if(goodbits > negative_d) {
      //řádek generuje jistou negativní odchylku,
      //ta je však menší než počet odpovídajících si jedniček
      //proto si řádek dáme do seznamu kandidátů
```

```
      přidej A[k] mezi kandidátní řádky
    } else {
      //řádek generuje víc negativních odchylek, než shodných bitů
      //proto jej můžeme rovnou zahodit
    }
  }
}
```

Důležitou součástí tohoto kódu je funkce `počet_nenulových_bitů()`. Tato operace není nativně podporována mikroprocesorem (což je s podivem), takže její implementaci nejlépe zajistíme vyhledávací tabulkou – předpočítáme si tabulku (pole), obsahující na každé pozici i hodnotu odpovídající počtu jedniček v bitově zakódovaném čísle i. Tabulka může mít například $2^8 = 256$ hodnot, funkční hodnotu 32bitového vektoru pak spočítáme jako součet funkčních hodnot čtyř osmibitových vektorů. Toto řešení je velmi rychlé a paměťově nenáročné.

Výše uvedený algoritmus tedy roztřídí řádky a_k do tří skupin:
Vybrané: Ty určitě budou součástí hledané podmnožiny.
Vyřazené: Ty součástí podmnožiny určitě nebudou.
Kandidátní: Ty se zúčastní dalšího výpočtu.

Nyní tedy už máme určitou množinu řádků A, které negenerují žádnou negativní odchylku, takže jejich obsažení v hledané podmnožině je určitě bezpečné. Superpozici těchto řádků tedy odečteme od x_i a dále budeme pokračovat jen se zbývajícími bity, označme si je `rest`.

```
rest = X[i] - goodsum
if(rest==0) hotovo!
```

Je-li `rest=0`, výpočet je u konce, neboť jsme našli přesné vyjádření x_i pomocí superpozice podmnožiny řádků z A. Vektor f_i je pak roven superpozici vybraných řádků z A.

$$f_i = \bigvee(a_j : a_j \text{ je mezi vybranými řádky})$$

Což můžeme zapsat také v pseudo-jazyku

```
if(rest==0) {
  F[i] = 0
  pro všechny vybrané_A[j] {
    F[i] = F[i] OR vybrané_A[j]
  }
  return F[i]; //hotovo!
}
```

Je-li `rest` nenulové, pokračujeme dál. Nejprve omezíme všechny kandidátní řádky na bity neobsažené v `goodsum`.

```
pro všechny kandidátní_A[j] {
  kandidátní_A[j] = kandidátní_A[j] AND rest
}
```

Mezi nimi pak hledáme takovou podmnožinu, jejíž odchylka d od vektoru `rest` je minimální. To už opravdu musíme provést zkoušením všech kombinací, počítáme zde již obyčejnou d jako počet rozdílných bitů mezi dvěma vektory.

```
d[k] = počet_nenulových_bitů(A[k] XOR rest)
```

Hlavní vliv na rychlost této poslední fáze výpočtu má počet kandidátních řádků. Naštěstí se při experimentech ukázalo, že počet kandidátních řádků je obvykle velmi nízký, drtivá většina řádků z A totiž přímo padne mezi vybrané či vyřazené řádky.

3 Analýza složitosti

Pokusme se nyní formálně vyjádřit časovou složitost algoritmu představeného v předchozí kapitole. Obyčejné slepé hledání matice F má řádovou složitost $O(2^{nm} \cdot nmp) \approx O(2^{nm})$, neboť musíme vyzkoušet všechny bitové kombinace $O(2^{nm})$ a u každé pak provést maticový součin s výpočtem odchylky $O(nmp)$.

Při hledání matice F po řádcích již dosáhneme značného zrychlení – takto upravený algoritmus má totiž řádovou složitost jen $O(n \cdot 2^m \cdot p) \approx O(2^m)$, neboť pro každý řádek datové matice $O(n)$ opakujeme prohledání všech podmnožin $O(2^m)$ s porovnáním vůči datové matici $O(p)$.

Rychlost našeho výsledného algoritmu je však ještě vyšší a závisí především na počtu řádků z A, které vyjdou z první části algoritmu jako kandidátní a zúčastní se tak jeho druhé části. Zrychlení je přitom o to větší, čím méně je těchto kandidátních řádků. Poměr zrychlení vůči původnímu algoritmu je však nelineární, neboť se jedná o zmenšování hodnoty mocniny N ve vzorci 2^N. Protože tento vztah nelze vyjádřit obecně, rozhodli jsme se analyzovat chování algoritmu na reálných výpočtech. Změřili jsme celkové počty vybraných, vyřazených a kandidátních řádků. Naměřené hodnoty a jejich poměr ukazuje tabulka 2.

Během experimentů v reálném prostředí byly zpracovány více než dvě miliardy maticových řádků. Naší snahou bylo zahrnout rovnocenné množství výpočtů na třech algoritmech (genetický algoritmus GABFA, konceptuální svazy, blind search BFA), podrobnosti o těchto algoritmech využívajících pseudo-dělení binárních matic jsou v další kapitole. Především se ukázalo, že je obrovský rozdíl mezi počty provedených pseudo-dělení. Drtivou většinu výpočtů připadlo na algoritmus blind search, naopak pod algoritmus konceptuálních svazů spadá jen jedno promile z celkového počtu výpočtů.

Podíl kandidátních řádků na celkovém počtu je 7.55%. Jelikož však nejdůležitější algoritmus, kde se pseudo-dělení používá, je genetický algoritmus GABFA, berme raději za směrodatnou hodnotu podílu 11.79%. Hodnota udává, že namísto původní složitosti řádu $O(2^m)$ má nový algoritmus v průměrném případě

Tabulka 2. Výsledky experimentů. Tabulka uvádí podíly vybraných, vyřazených a kandidátních řádků A na celkovém počtu více než dvou miliard výpočtů.

	řádky matice A			
	vybrané	vyřazené	kandidátní	celkem
Genetický algoritmus GABFA:				
(provedeno vícero měření)	2 573 961	1 228 738	3 022 301	
	1 560 609	833 598	1 550 793	
	540 595	348 354	1 166 051	
	10 452	173 928	50 620	
	7 834	175 419	51 747	
	12 990	192 808	59 202	
	1 914 331	34 034 944	2 309 925	
	1 604 040	29 477 802	1 767 758	
	8 224 812	66 465 591	9 978 397	84 668 800
	9.71%	78.50%	**11.79%**	
Konceptuální svazy:				
	331068	1918156	45726	2 294 950
	14.43%	83.58%	**1.99%**	
Blind search BFA:				
	10 177 140	1 839 111 783	147 242 916	1 996 531 839
	0.51%	92.12%	**7.37%**	
celkem:				
	18 733 020	1 907 495 530	157 267 039	2 083 495 589
	0.90%	**91.55%**	**7.55%**	

složitost řádu $O(2^{0.1179m})$. V nejhorším případě je složitost stále $O(2^m)$, ale reálně je mezi původním a novým algoritmem propastný rozdíl[1], a to tím větší, čím větší je počet faktorů. Například při výpočtu 35 faktorů se mění výpočetní časy z jednotek dnů na jednotky sekund. (Porovnáváme zde již částečně optimalizovaný výpočet po řádcích s naším algoritmem.) Dvě miliardy řádků, které jsou zahrnuty do zde uvedených experimentů, byly naším algoritmem zpracovány během několika desítek sekund.

Neméně důležitý je také fakt, že veškeré výpočty provádíme pomocí paralelních booleovských instrukcí, což na běžných počítačích s procesory typu x86 znamená výpočet 32 hodnot v jednom průchodu algoritmem. Z toho důvodu jsou také nejdůležitější části algoritmů uváděny v pseudo-kódu, což je mnohem jednodušší, než používání klasické algebraické notace.

4 Aplikace algoritmu

V této kapitole představíme několik algoritmů, kde se pseudo-dělení binárních matic uplatní. Všechny navíc mají jednu společnou vlastnost (neboli „společný

[1] Srovnejte například s fenoménem algoritmu Quick sort.

faktor“): Jsou používány k redukci dimenze binárního prostoru, extrakci příznaků a vyhledávání shluků či faktorů v binárních datech.

4.1 Blind search BFA

Jak již bylo zmíněno, algoritmem nejvíce závislým na pseudo-dělení binárních matic, je Blind search BFA (viz [3], [4]). Tento algoritmus postupně prochází úplně všechny kombinace bitových hodnot pro matici A a pro každou takovou kandidátní matici počítá pseudo-podíl X/A. Kandidátních matic je opravdu hodně, proto se nelze divit obrovskému počtu opakování pseudo-dělení, často i mnohem větších řádů než miliardy, na které jsme narazili během výše uvedených experimentů (viz tabulka 2).

Jako konkrétní příklad zde uveďme datovou sadu p3 (matice o rozměru 100×100), kterou jsme dlouho považovali za problematickou, neboť výpočet BFA na ní trval 7 a půl dne. Po přechodu na nový algoritmus pseudo-dělení je výpočet hotov na stejném počítači za pouhých 8 sekund.

4.2 BFA pomocí konceptuálních svazů

Dalším algoritmem používajícím pseudo-dělení je výpočet BFA pomocí převodu na problém stavění konceptuálních svazů (viz [4], [5]). Tento algoritmus se snaží pro výpočet BFA využít řadu znalostí z formální konceptuální analýzy (FCA). Ty jsou v posledních letech středem výzkumu mnoha vědců, takže v této oblasti můžeme najít mnoho zajímavých algoritmů. (BFA je narozdíl od FCA nehierarchickou analýzou.) Jak ukázaly experimenty, algoritmus provádějící převod výsledků FCA na výsledky BFA vystačí s poměrně malým množstvím pseudo-dělení (pouhé jedno promile počtu, který byl nutný pro výpočet stejného problému metodou Blind search). Také je zajímavé, že pouze 1.99% řádků spadá do kandidátních, takže skutečná složitost našeho algoritmu pseudo-dělení se zde mění z exponenciální na de facto lineární.

4.3 GABFA

Genetický algoritmus GABFA je dalším možným způsobem, jak řešit problém BFA. Narozdíl od výše uvedených algoritmů, průběh tohoto algoritmu závisí na náhodných číslech, takže podíly kandidátních řádků na celkovém počtu, stejně jako samotný celkový počet výpočtů se v jednotlivých případech mění. I při opakovaném zkoušení GABFA na identických datech nám vycházely pokaždé velmi rozdílné hodnoty. Zatímco samotný algoritmus GABFA nakonec vždy dospěl ke stejnému (a správnému) řešení, výpočty pseudo-dělení, které k tomu potřeboval, se v jednotlivých případech velmi lišily. Proto jsme experimenty několikrát zopakovali a výsledná hodnota 11.79% kandidátních řádků je jejich váženým průměrem.

4.4 Lživé atraktory neuronové sítě

Výzkum, který byl v posledních 10 letech prováděn na ruské akademii věd vedl k závěru, že lidský mozek provádí mimo jiné operace odpovídající binární Hopfieldově síti (viz [6]). Jedním z výsledků tohoto výzkumu je pak modifikovaná Hopfieldova neuronová síť, která umí provádět výpočet BFA. Výhodou tohoto algoritmu je především fakt, že tato síť dokáže zpracovat mnohem větší datové sady než předešlé algoritmy. Při správném naučení sítě odpovídají hledané faktory atraktorům sítě. (Atraktor je stav, do něhož síť konverguje po počáteční náhodné inicializaci.)

Výpočet této sítě může fungovat i bez pseudo-dělení binárních matic. Nepříjemným problémem jsou však lživé atraktory (spurious attractors), které se v Hopfieldových sítích přirozeně objevují. Jsou to stavy, do kterých síť v některých případech konverguje, aniž by tento stav odpovídal faktoru. Zde se pak může uplatnit pseudo-dělení binárních matic. Pomocí pseudo-dělení totiž velmi snadno spočítáme matici F, kterou jinak neuronová síť vypočítat nedokáže. Lživým atraktorům (chybným řádkům matice A) pak odpovídají nulové sloupce matice F. Nutno podotknout, že není známa žádná jiná automatická metoda detekce lživých atraktorů v tomto typu neuronové sítě.

5 Shrnutí

Představen byl nový algoritmus pro pseudo-dělení binárních matic, který nabízí masivní využití v binární faktorové analýze. Analýza složitosti ukázala, že v nejhorším případě má nový algoritmus časovou složitost řádu $O(2^m)$, stejně jako algoritmus původní. Průměrná výpočetní složitost je však přibližně $O(2^{0.1179m})$, což je o mnoho lepší. Praktické experimenty ukázaly, že nový algoritmus redukuje mocninu ve výrazu složitosti zhruba $9\times$, čímž dosahuje nesrovnatelně lepších výsledků. Například při výpočtu 35 faktorů se mění výpočetní časy z jednotek dnů na jednotky sekund. Přínos algoritmu pro praxi byl doložen několika ukázkami jeho aplikace.

Reference

1. *BMDP (Bio-Medical Data Processing).* A statistical software package. SPSS. `http://www.spss.com/`
2. Húsek, D., Frolov, A.A., Řezanková, H., Snášel, V., Keprt, A.: O jednom neuronovém přístupu k redukci dimenze. In *Znalosti 2004*, Brno, **2004**, ISBN 80-248-0456-5.
3. Keprt, A.: *Paralelní řešení nelineární booleovské faktorizace.* VŠB Technical University, Ostrava (unpublished paper), **2003**, 14 pp.
4. Keprt, A.: Using Blind Search and Formal Concepts for Binary Factor Analysis. In *Dateso 2004 – Proceedings of 4th annual workshop.* Ed. Václav Snášel, Jaroslav Pokorný, Karel Richta, VŠB – Technická Univerzita Ostrava, Czech Republic; CEUR WS – Deutsche Bibliothek, Aachen, Germany; **2004**, pp. 120–131, ISBN 80-248-0457-3 (VŠB TUO), ISSN 1613-0073 (CEUR).

5. Keprt, A., Snášel, V.: Binary Factor Analysis with Help of Formal Concepts. In *CLA 2004 – Concept Lattices and their Applications.* Eds. Václav Snášel, Radim Bělohlávek, VŠB – Technical University of Ostrava, Czech Republic, **2004** pp. 90–101, ISBN 80-248-0597-9.
6. Sirota, A.M., Frolov, A.A., Húsek, D.: Nonlinear Factorization in Sparsely Encoded Hopfield-like Neural Networks. In *ESANN 1999 Proceedings – European Symposium on Artificial Neural Networks.* D-Factor Public., Bruges, Belgium, **1999**, pp. 387–392, ISBN 2-600049-9-X.

Annotation:

Binary Matrix Pseudo-Division and its Applications

The level of success of each important algorithm depends also on its several support algorithms. Dimension reduction of a binary space using binary factor set (e.g. reduction of a document-term matrix) is very dependent on the ability to pseudo-divide generic binary matrices. This paper presents the algorithm for this kind of pseudo-division.

The presented algorithm seems to be very good. Its time complexity is $O(2^m)$ in the worst case, which is the same as the old simple algorithm has. This is the worst case, but the average time complexity of our algorithm is approx. $O(2^{0.1179m})$, and that's significantly better. We also presented the benefits of our algorithm on practical examples.

Individuálne znalosti a ich prenos v tíme mobilných robotov

Peter Kostelník, Adrián Tóth

Katedra kybernetiky a umelej inteligencie, FEI, Technická Univerzita, Košice, Letná 9, 040 01, Košice

{kostelni,totha}@neuron.tuke.sk

Abstrakt. Príspevok sa zaoberá individuálnym získavaním znalostí vo forme symbolov v komunite mobilných robotických agentov a negotiatívnym prenosom týchto znalostí. Symboly reprezentujúce prostredie vytvárajú jednoduchý komunikačný jazyk, ktorý sa u jednotlivých agentov môže líšiť. Uvádza sa stručný prehľad reprezentácie jednoduchých úloh a udalostí vo forme symbolov a možnosti ich spájania do modelov komplexnejších úloh a udalostí. Popisuje sa mechanizmus prenosu individuálnych znalostí v skupine agentov, kde každý z agentov má k dispozícii len časť znalostí potrebných pre splnenie definovanej úlohy. Na záver sa uvádza výsledok základného experimentu zameraného na sledovanie vplyvu kolektívneho prenosu znalostí na shopnosti jednotlivých agentov splniť definovanú úlohu.

Kľúčové slová: kognitívny agent, multi-robotické systémy, evolučná lingvistika, prenos znalostí

1 Úvod

Jeden z najznámejších a veľmi zložitých problémov v oblasti umelej inteligencie, ale aj kognitívnych vied vo všeobecnosti, je odpoveď na otázku: „Je pre umelý systém možné získať význam zdanlivo bezvýznamných symbolov len na základe interakcie s reálnym prostredím? Ak áno, ako?“. Tento problém je najčastejšie referovaný, ako tzv. *problém ukotvenia symbolu* (symbol grounding problem). Pre agentov, ktorý v nejakej forme „uvažujú“ o svojom prostredí alebo používajú koncepty popisujúce dané prostredie, je zvyčajne problém ukotvenia symbolu potrebné riešiť.

Práca, ktorej časť je prezentovaná v tomto príspevku, sa zaoberá možným riešením problému formovania použiteľnej symbolickej reprezentácie prostredníctvom interakcií s prostredím a možnosťami použitia takto získanej reprezentácie. Znalosti robotických agentov použitých v tejto práci sú získané v procese učenia a sú reprezentované symbolmi, pričom tieto symboly môžu priamo predstavovať pojmy jednoduchého komunikačného jazyka vytvoreného medzi agentom a jeho inštruktorom. Na základe inšpirácie z teórie semiotiky sa predpokladajú tri základné časti, z ktorých je každý symbol (koncept) zložený: (1) *referent* – reálne objekty v prostredí (kontext), na ktoré symbol referuje, (2) *význam*, ktorý symbol nesie a (3) *forma* – označenie, reprezentatívne meno symbolu.

Vzhľadom na možnosť, že rôzni agenti mohli nadobudnúť znalosti v procese konzultacií s rôznymi inštruktormi, dochádza k situácii, kedy sú symboly z rovnakých

domén reprezentované rôznymi formami, teda, každý z agentov môže mať vyvinutý jazyk používajúci odlišné pojmy pre rovnaké udalosti v danej doméne. Pri skupinovej kooperácii vyžadujúcej komunikáciu takýchto agentov potom dochádza k situácii, kedy si agenti nie sú schopní porozumieť, pretože každý z nich používa odlišný jazyk. V takomto prípade vzniká potreba stabilizácie spoločného slovníka takejto skupiny agentov. Možným riešením tohto problému je použitie tzv. jazykových hier, ktoré umožňujú skupinovú stabilizáciu slovníka spoločného pre celú skupinu agentov, pričom spoločný slovník, *lexikon*, vzniká v procese skupinovej negociácie informácií o jednotlivých pojmoch jazykov agentov (napr. [3]). Ďalším problémom, ktorý vzniká pri potrebe kooperatívnej tímovej akcie je, že znalosti v komunite agentov sú distribuované, to znamená, že žiaden z agentov nemá kompletné informácie potrebné na splnenie požadovanej úlohy. V tomto prípade vzniká potreba prenosu individuálnych znalostí o čiastkových modeloch úloh, pomocou ktorých je možné dosiahnuť riešenie požadovanej cieľovej úlohy.
V nasledujúcich častiach textu bude stručne popísaná základná použitá riadiaca architektúra využívajúca „ukotvené“ symboly a samotné riešenie problému prenosu individuálnych znalostí o čiaskových úlohách v komunite agentov operujúcich v spoločnom prostredí.

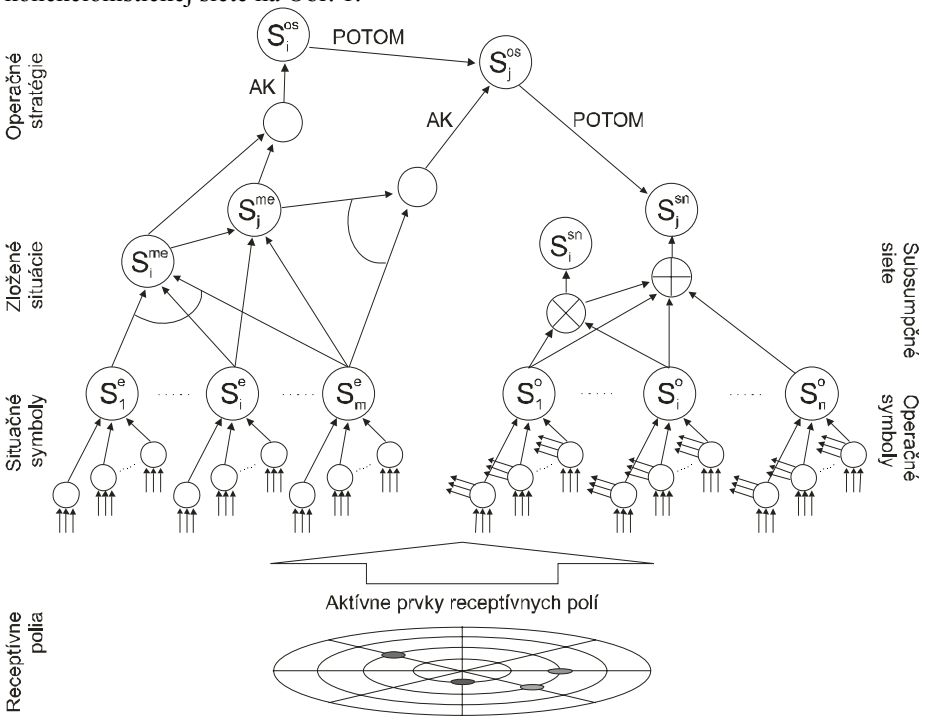
2 Popis riadiacej architektúry

Riadiaca architektúra použitá v tejto práci je tvorená hierarchiou piatich znalostných úrovní:

- *Úroveň operačných a situačných symbolov*: Jednoduché modely situácií a riešení triviálnych úloh. Jednotlivé symboly sú získavané v procese učenia pomocou demonštrácie ľudským inštruktorom. Význam symbolov je ukotvený v reálnom prostredí prostredníctvom percepčných informácií.
- *Úroveň meta-symbolov*: Komplexné koncepty vytvorené definíciou hierarchií už existujúcich operačných symbolov (subsumpčné siete) a situačných symbolov (komplexné situácie reprezentované AND/OR grafmi).
- *Úroveň operačných stratégií*: Interpret pravidlového systému obsahujúci množinu komplexných stratégií reprezentovaných produkčnými pravidlami, kde predpokladová časť je tvorená logickou kombináciou situačných symbolov (resp. situačných meta-symbolov) a dôsledková časť obsahuje operačný symbol, subsumpčnú sieť, resp. ďalšiu operačnú stratégiu. Operačné stratégie môžu byť vo forme pravidiel explicitne definované, ale môžu byť získané aj v procese učenia konzultovaného s inštruktorom.
- *Úroveň multi-agentovej kooperácie*: Umožňuje použitie uvedených znalostných úrovní pri tímovej práci, ale boli navrhnuté aj mechanizmy pre rôzne typy prenosu znalostí medzi agentami.

Tento príspevok sa zaoberá práve možnosťami prenosu znalostí v komunite agentov, na úrovni jednoduchých symbolov, preto v nasledujúcich častiach textu bude uvedený stručný popis reprezentácie symbolov a možnosti prenosu takto získaných znalostí.

Prehľad navrhovanej architektúry je schematicky ilustrovaný formou konekcionistickej siete na Obr. 1.



Obr. 1. Schematický prehľad riadiacej architektúry.

3 Všeobecný model symbolu

Úroveň operačných a situačných symbolov je najnižšou úrovňou navrhovanej architektúry. Táto úroveň v prípade operačných symbolov umožňuje učenie sa riešiť jednoduché úlohy prostredníctvom priamych senzo-motorických prepojení, v prípade situačných symbolov je možné učiť sa rozpoznávať jednoduché situácie/udalosti, ktoré môžu vzniknúť počas interakcie robota s okolím. V oboch prípadoch sú jednotlivé modely úloh, resp. situácií/udalostí realizované, ako asociačné siete. V prípade senzo-motorických, resp. operačných symbolov sú senzorické informácie priamo prepájané na zodpovedajúce motorické akcie. Architektúra operačných a situačných symbolov je identická, až na to, že situačné symboly neobsahujú prepojenie na motoriku.

Učenie sa oboch typov symbolov je realizované pomocou učiacich príkladov generovaných človekom-inštruktorom v procese demonštrácie. Počas učenia, v každom kroku demonštrácie inštruktor použije nejakú motorickú akciu, demonštrácia je teda realizovaná, ako teleoperované riadenie robota. Jeden učiaci príklad potom obsahuje dve zložky:

- aktuálny senzorický vstup reprezentovaný aktuálnou percepčnou kategóriou, kde percepčná kategória predstavuje prípad aktivácie senzorických receptívnych polí,
- aktuálnu motorickú reakciu.

Keďže je tento model úlohy/situácie chápaný, ako symbol, podľa definície symbolu by mal byť zložený z troch vzájomne prepojených súčastí: (1) *formy*, (2) *referenčného objektu* a (3) *významu*. Nový symbol vzniká vždy na požiadavku človeka-inštruktora a je prioritne definovaný unikátnym názvom úlohy/situácie, ktorý reprezentuje formu daného symbolu. Význam a referencia sú reprezentované, ako množina naučených asociačných pravidiel zodpovedajúcich modelu učenej úlohy/situácie.

Jednotlivé asociačné pravidlá sú základnými stavebnými prvkami symbolov. Pre oba typy symbolov je základom asociačného pravidla percepčná kategória. Každá percepčná kategória je prakticky modelom jednej konkrétnej situácie, ktorá sa vyskytla počas interakcie robota s prostredím, je vytvorená kombináciou prvkov receptívnych polí a zodpovedá referenčnej časti symbolu. Množina percepčných kategórií vytvára významovú časť symbolu. Symbol je teda vytvorený pomocou asociácií prvkov receptívnych polí na zodpovedajúce situácie reprezentované percepčnými kategóriami. V prípade operačných symbolov sa navyše na jednotlivé percepčné kategórie napájajú motorické akcie použité v daných situáciách inštruktorom.

V tejto práci (podobne, ako napr. v [8]) nie je symbol, podľa definície, úplný. Chýba tu totiž priame napojenie na referenčný objekt. Referenčné objekty, resp. v tomto prípade, všeobecnejšie, referenčné situácie sú reprezentované len vo forme prvkov receptívnych polí vytvárajúcich percepčné kategórie. Táto väzba nie je síce priama, ale postačuje pre identifikáciu referenčného objektu. Aj napriek tomuto ohraničeniu predpokladáme, že symbol reprezentujúci model úlohy je možné považovať za ukotvený.

Učiaci proces je možné úplne zjednodušene popísať tak, že pri učení sa situačných symbolov sa zapamätávajú isté charakteristické situácie prezentované v procese demonštrácie, v prípade operačných symbolov sa na zapamätané situácie navyše napájajú akcie použité inštruktorom v týchto situáciách. Učiaci proces si, samozrejme, nevyžaduje uchovanie všetkých prezentovaných situácií. Percepčné kategórie (situácie) sú učiacim algoritmom zhlukované a produktom zhlukovania sú len prototypy percepčných kategórií. Podľa [2] je možné definovať tri základné fázy procesu ukotvenia symbolu:

- *Ikonizácia*: proces prevodu senzormi získaných analógových údajov do tzv. ikonických štruktúr. V tejto práci je prevod na ikony riešený na úrovni percepcie, ikonické štruktúry zodpovedajú receptívnym poliam, ktorých aktivácia vytvára percepčné kategórie.
- *Diskriminácia*: schopnosť porovnať dva senzorické vstupy určiť mieru podobnosti, resp. rozdielnosti týchto vstupov. V tejto práci je diskriminácia kľúčovou fázou „ukotvenia“ symbolov a je realizovaná, ako mechanizmus zhlukovania percepčných kategórií.
- *Identifikácia*: schopnosť rozpoznať, či konkrétny senzorický vstup zodpovedá symbolu. V tejto práci fáza identifikácie využíva diskriminačný mechanizmus pre určenie podobnosti aktuálnej situácie s možnými situáciami zodpovedajúcimi symbolu. Výstupom identifikácie je symbol (resp. všeobecne, množina

symbolov) obsahujúci kategóriu, ktorá najviac zodpovedá aktuálnemu senzorickému vstupu.

V procese ukotvenia symbolu navrhovanom v tejto práci je prakticky nemožné oddeliť diskriminačný mechanizmus od mechanizmu identifikácie. Fázu identifikácie je možné interpretovať v kontexte navrhovanej architektúry dvoma spôsobmi:

- V procese učenia, ktorý je založený na kategorizácii percepčných informácií je potrebné identifikovať pomocou diskriminačného mechanizmu najbližší prototyp symbolu k aktuálnemu senzorickému vstupu. Pokiaľ nie je aktuálna senzorická informácia dostatočne podobná so žiadnym prototypom (identifikácia bola neúspešná), vytvorí sa nový prototyp. Pokiaľ bola identifikácia úspešná, identifikovaný prototyp sa aktualizuje pomocou aktuálnej senzorickej informácie.
- V procese samostatného používania symbolu robotom je identifikácia použitá:
  - V prípade aktuálne používaného operačného symbolu na určenie najpravdepodobnejšej reakcie napojenej na prototyp najpodobnejší s aktuálnym senzorickým vstupom.
  - V prípade situačných symbolov je potrebné identifikovať všetky situácie, ktoré zodpovedajú aktuálnej senzorickej informácii. To znamená, pre všetky situačné symboly sa identifikujú najpodobnejšie prototypy. Symboly tých prototypov, ktoré spĺňajú nejaké vopred definované kritérium podobnosti sa označia za aktívne. Množina aktivovaných situačných symbolov potom komplexne popisuje aktuálnu situáciu v okolí robota.

Nasledujúca časť príspevku sa bude zaoberať možnosťami prenosu individuálne získaných znalostí v tíme agentov na úrovni operačných a situačných symbolov.

4 Jazykové hry

Pojem jazykových hier bol predstavený Ludwigom Wittgensteinom (1958). Wittgenstein nazýval jazykovou hrou prakticky každé použitie jazyka. Význam jazykovej hry priamo závisí od toho, akým spôsobom je jazyk používaný. Wittgenstein navrhol niekoľko typov jazykových hier, ako napr. dávanie pokynov a ich plnenie, popis výskytov objektu alebo popis jeho rozmerov, konštrukcia objektu z jeho popisu, oznamovanie výskytu udalosti, uvažovanie o udalosti, a pod. Na základe Wittgensteinovej inšpirácie bolo vyvinutých niekoľko špeciálnych variant jazykových hier, ktoré boli implementované najčastejšie v oblasti mobilnej robotiky. Ako príklad je možné uviesť tzv. *diskriminačnú hru* [5], *imitačnú hru* [1], *pozorovaciu hru* [4], *odhadovaciu hru* [7], *identifikačnú hru* [8] a mnohé ďalšie.

V kontexte tejto práce je pojem jazykovej hry chápaný, ako systém interakcií medzi agentami, ktorí používajú nejaký jazyk za účelom komunikácie určitých informácií o externom svete, ale aj o jazyku samotnom. Jazyková hra je vymedzená vopred určenými pravidlami, ktoré definujú: čo môže byť povedané, ktorý agent môže hovoriť a kedy, čo je výsledkom reakcií agentov, aké formy nelingvistických informácií môžu byť použité (pod pojmom nelingvistická informácia sa chápe napr. zameranie pozornosti na nejaký objekt alebo situáciu v prostredí, ktorá bude

predmetom komunikácie alebo komunikácia nízkoúrovňových informácií, napr. ikonických popisov etalónov kategórií, lingvistická informácia môže byť napr. forma reprezentujúca symboly), a pod.

Ak komunita agentov operuje v spoločnom prostredí a rieši problémy v spoločnej doméne, s vysokou pravdepodobnosťou získava informácie o rovnakých udalostiach a akciách pre danú doménu, resp. prostredie. Ak je každý agent usmerňovaný iným inštruktorom, dochádza k situácii, kedy sa individuálne znalosti agentov o rovnakých modeloch riešenia úloh (operačné symboly), resp. rovnakých udalostiach (situačné symboly) líšia. Tieto znalosti sú rôzneho charakteru, každý z agentov má o určitom modeli riešenia úlohy, resp. určitej udalosti rôzne informácie v závislosti od toho, v akom kontexte dané znalosti získal. Komunikácia o znalostiach prebieha na úrovni operačných a situačných symbolov, pretože tieto symboly sú nosičom základných pojmov. Vyššie úrovne riadiacich a rozhodovacích štruktúr agentov sú priamo naviazané na operačné a situačné symboly.
Pre jednu situáciu, resp. jeden model riešenia úlohy môže mať viacero agentov vytvorený zodpovedajúci symbol, takéto symboly, keďže boli získané individuálne, sa u každého agenta budú s vysokou pravdepodobnosťou líšiť jednak svojím obsahom a jednak formou, napriek tomu, že reprezentujú prakticky tú istú vec. Táto situácia vyplýva z prirodzenej schopnosti agentov získavať informácie o prostredí a je typická.
Problém nastáva až na úrovni multi-agentovej kooperácie, pokiaľ je potrebné, aby agenti o svojom prostredí a úlohách, ktoré vedú k celkovému riešeniu, vzájomne komunikovali. Cieľom jazykovej hry navrhovanej v tomto príspevku je práve prenos znalostí o pozorovaných udalostiach a modeloch čiastkových úloh..

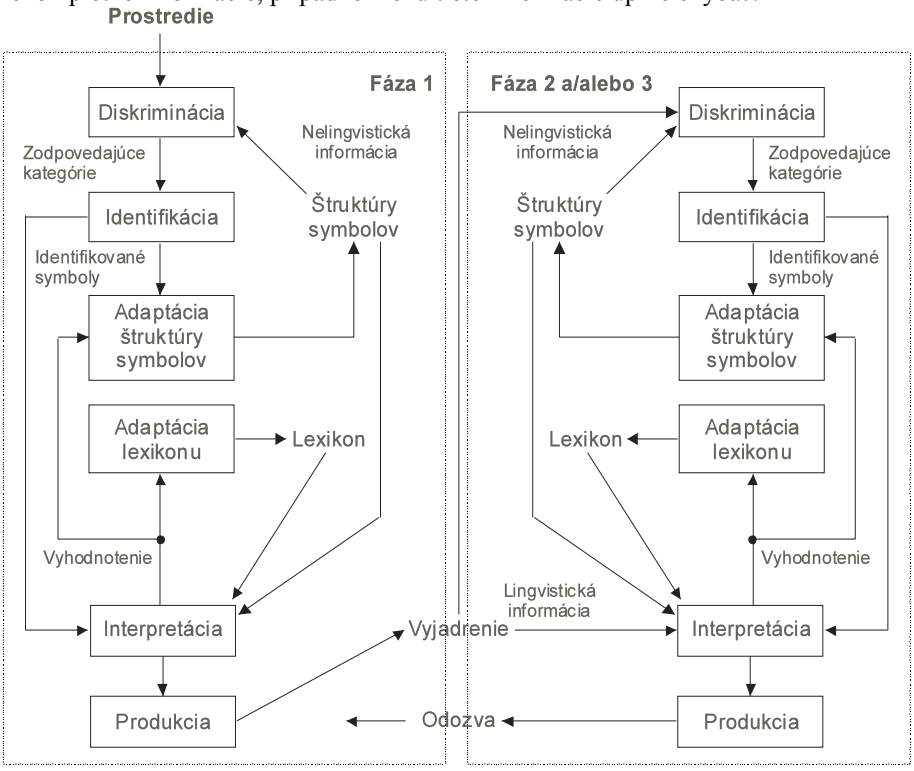
4.1 Všeobecný scenár jazykovej hry

Scenár jazykovej hry implementovaný v rámci tejto práce je zovšeobecnením a modifikáciou inšpirácie publikovanej napr. v [6]. Pre účely výskumu, ktorého je tento príspevok súčasťou, boli vyvinuté dva základné scenáre jazykových hier:

- *Koordinovaný (off-line) scenár:* Umožňuje stabilizáciu spoločného slovníka a/alebo prenos znalostí prostredníctvom moderovaného dialógu bez interakcie s reálnym prostredím. V tomto prípade plní jeden z agentov rolu koordinátora hry a určuje, ktorý agent môže komunikovať a kedy.
- *Nekoordinovaný (on-line) scenár:* Umožňuje stabilizáciu spoločného slovníka a/alebo prenos znalostí priamo počas operácie tímu v reálnom prostredí. Tento scénar nedefinuje, kedy možu jednotlivý agenti komunikovať.

Tento príspevok je zameraný na popis možností prenosu znalostí priamo počas interakcie s prostredím. Každý z agentov sa snaží splniť požadovanú cieľovú úlohu, ktorá si ale vyžaduje znalosti o čiastkových úlohách, pomocou ktorých je možné cieľovú úlohu splniť. Cieľová úloha je definovaná, ako operačná stratégia pozostávajúca z pravidiel, ktorých dôsledkovou časťou sú jednoduchšie operačné stratégie, subsumpčné siete alebo operačné symboly. Pre jednoduchosť sa predpokladá, že je známa štruktúra použitých operačných stratégií, subsumpčných sietí, znalosti o udalostiach v prostredí (situačné symboly) sú tiež kompletné.

Znalosti agentov sa líšia len na úrovni operačných symbolov, pomocou ktorých sú subsumpčné siete a stratégie vytvorené. Dané operačné symboly môžu obsahovať nekompletné informácie, prípadne môžu tieto informácie úplne chýbať.



Obr. 2. Všeobecná štruktúra procesov a dátových tokov jedného komunikačného aktu.

V prezentovanej implementácii jazykovej hry prebieha komunikácia prakticky len v prípade situácie, kedy agent potrebuje použiť jeden alebo viac alternatívnych modelov triviálnych úloh (sú reprezentované operačnými symbolmi), ale nemá znalosti o tom, ako daný model (modely) použiť.

Nekoordinovaný scenár využíva tri základné fázy komunikácie:

- *Identifikácia problému*: Pokiaľ dôjde k situácii, ktorá si vyžaduje použitie jedného alebo viacerých alternatívnych operačných symbolov a tieto symboly neobsahujú informácie o tom, ako reagovať v danej situácii, agent vytvorí model lokálneho problému a komunikuje ho ostatním agentom v tíme.
- *Generovanie riešenia*: Agenti po prijatí takejto žiadosti o pomoc sa pokúsia vytvoriť odpoveď obsahujúcu riešenie problému, táto odpoveď je opäť komunikovaná celému tímu.
- *Adaptácia znalostných štruktúr*: Po prijatí riešenia sa každý z agentov pokúsi rozšíriť vlastné znalostné štruktúry o nové informácie o riešení daného problému.

Ku komunikačným aktom dochádza nekoordinovane a celá sekvencia je spustená prvou fázou, teda identifikovaným problémom. Všetky fázy komunikácie budú podrobnejšie popísané v nasledujúcej časti textu.
Všeobecná štruktúra procesov a dátových tokov jedného komunikačného aktu je ilustrovaná na Obr. 2.

4.2 Identifikácia probému

Problém vzniká vtedy, keď agent nie je schopný reagovať na aktuálny stav prostredí žiadnou motorickou akciou, teda v situácii, kedy si aktuálne aktívna operačná stratégia alebo subsumpčná sieť vyžaduje použitie operačného symbolu, ktorého štruktúra asociačných pravidliel neobsahuje znalosti, ako reagovať na aktuálny stav v prostredí. Aktuálny stav v prostredí je reprezentovaný aktuálnou percepčnou kategóriou. Vo všeobecnosti, operačný symbol, ktorý má byť použitý nie je schopný generovať odozvu vtedy, keď jeho štruktúra neobsahuje asociačné pravidlo, ktoré s istou toleranciou mapuje aktuálnu percepčnú kategóriu na konkrétnu motorickú reakciu. Štruktúra znalostí pre úlohu reprezentovanú daným symbolom teda nie je kompletná. V tomto prípade agent vygeneruje správu formulovanú, ako žiadosť o pomoc. Táto správa obsahuje zoznam neznámych asociačných pravidiel pre všetky operačné symboly, ktoré v danej situácii boli zodpovedné za generovanie motorickej reakcie a neobsahovali potrebné asociačné pravidlá. Správa je zložená z dvoch častí:

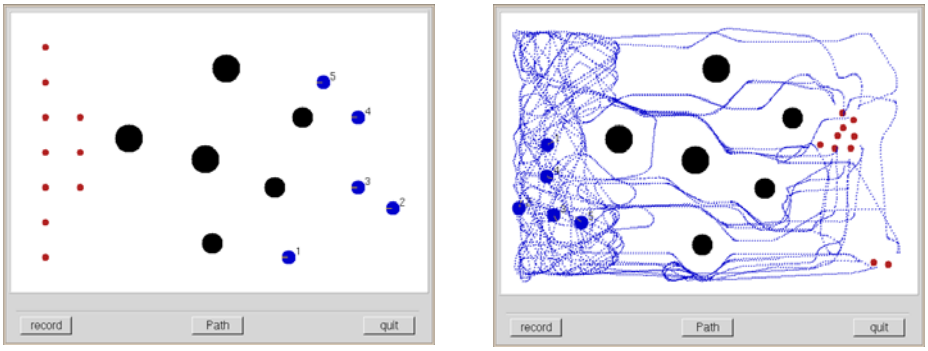
- *Nelingvistická časť informácie:* Aktuálna percepčná kategória, ktorá zodpovedá situácii v prostredí, kedy vznikla potreba použiť problémové operačné symboly.
- *Lingvistická časť informácie:* Obsahuje zoznam foriem daných operačných symbolov, ktorý agent nie je schopný použiť.

Správa o identifikovanom nedostatku v znalostných štruktúrach je komunikovaná každému agentovi v tíme.

4.3 Generovanie riešenia

Každý agent, ktorý prijal správu o identifikovanom probléme sa pokúsi vygenerovať odpoveď obsahujúcu možné riešenie. Pre každý komunikovaný symbol je potrebné zistiť, či znalostni agenta obsahujú symbol so zodpovedajúcou formou, pokiaľ áno, agent sa pokúsi nájsť asociačné pravidlo, ktoré s definovanou toleranciou mapuje komunikovanú percepčnú kategóriu na zodpovedajúcu motorickú akciu. Pokiaľ takéto pravidlo existuje, je vložené do zoznamu riešení a označené formou daného operačného symbolu. Pokiaľ takto získaná odpoveď - zoznam riešení nie je prázdny, agent bol schopný reagovať aspoň na jeden z problémových operačných symbolov a odpoveď je komunikovaná opäť všetkým agentom v tíme, ktorí v prípade potreby adaptujú svoje znalostné štruktúry pre dané symboly.
Dôvod pre rozoslanie odpovede každému agentovi je založený na predpoklade, že podobná nekompletnosť znalostí sa môže vyskytnúť aj u iných agentov, ako len u jedného, ktorý na problémovú situáciu narazil. Dôsledkom tejto stratégie potom je, že všetci agenti, ktorí v budúcnosti s istou pravdepodobnosťou na rovnakú nekompletnosť narazia si doplnia štruktúru znalostí už v prípade prvého výskytu

daného problému. Teda, pre každý takýto problém prebehne komunikácia v tíme práve raz.
Pokiaľ žiaden z agentov na správu nebol schopný reagovať nastáva situácia, kedy tím, ako celok nemá indormáciu o spôsobe riešenia lokálnej úlohy, čo môže byť príčinou neschopnosti splniť globálny cieľ.



Obr. 3. Počiatočný a záverečný stav simulácie.

4.4 Adaptácia znalostných štruktúr

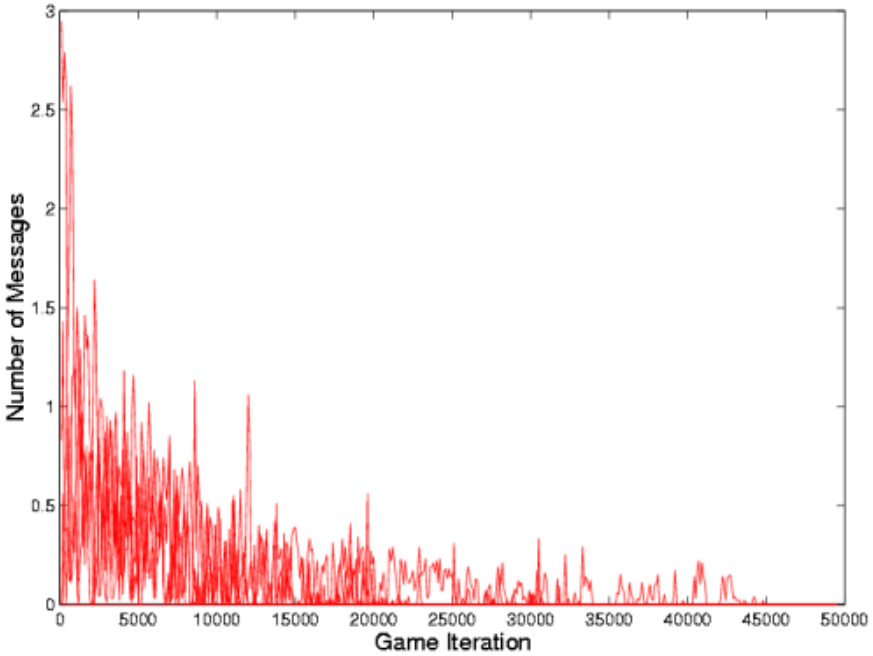
Každý agent po prijatí správy o riešení identifikovaných problémov si v prípade potreby rozšíri vlastné znalostné štruktúry o komunikované informácie. Pre každý z komunikovaných symbolov sa zistí, či znalosti s definovanou toleranciou obsahujú asociačné pravidlo pre problémovú percepčnú kategóriu. Pokiaľ takáto asociácia neexistuje, je do zodpovedajúceho symbolu doplnená. Takýmto spôsobom je možné predísť situácii, kedy by agent nebol schopný reagovať príslušným operačným symbolom na stav prostredia reprezentovaný komunikovanou percepčnou kategóriou.

Tabuľka 1. Distribúcia znalostí o deviatich čiastkových úlohách (T #i) v tíme piatich agentov (A #j).

	T #1	T #2	T #3	T #4	T #5	T #6	T #7	T #8	T #9
A #1	100%	0%	0%	0%	0%	0%	0%	0%	100%
A #2	0%	100%	0%	0%	0%	0%	0%	100%	0%
A #3	0%	0%	0%	100%	0%	100%	0%	0%	0%
A #4	0%	0%	100%	0%	0%	0%	100%	0%	0%
A #5	0%	0%	0%	0%	100%	0%	0%	0%	0%

5 Základný experiment

Pre experimentálne overenie možností prenosu znalostí v multi-robotickom tíme bola použitá jednoduchá úloha tzv. *delivery task*. Cieľom úlohy bolo presunúť objekty umiestnené kdekoľvek v ľavej časti prostredia na jedno operatívne určené miesto v pravej časti prostredia, pričom prostredie obsahovalo prekážky, ktorým sa bolo potrebné vyhýbať (príklad počiatočného a cieľového stavu riešenia úlohy je na Obr. 3.). Cieľová úloha bola definovaná, ako hlavná operačná stratégia zostavená z troch



Obr. 4. Počet komunikačných aktov v priebehu simulácie prenosu znalostí.

čiastkových operačných stratégií, štyroch čiastkových subsumpčných sietí a deviatich operačných symbolov. Úlohu bolo možné splniť len s použitím všetkých deviatich definovaných operačných symbolov. V experimentoch bola použitá skupina piatich agentov, kde každý z agentov mal kompletné znalosti o niektorých triviálnych úlohách. Žiaden agent nemal znaosti o riešení všetkých úloh, žiadni dvaja agenti nemali znalosti o rovnakej úlohe. Počiatočná distribúcia znalostí v tíme je uvedená v tabuľke 1.

Cieľom základného experimentu bolo overiť, ako prenos individuálnych znalostí pomocou jazykových hier v čase vplýva na shopnosť tímu agentov splniť cieľovú úlohu, ktorú na začiatku žiaden z agentov splniť schopný nebol.

V každom kroku simulácie bol u každého agenta sledovaný počet prijatých správ, na ktoré bol schopný odpovedať. Počet spracovaných správ pre päť agentov počas 50000 krokov simulácie je ilustrovaný grafom na Obr. 4. Podľa výsledkov experimentu, počet komunikovaných problémových situácií konverguje v čase k nule. Po dostatočnom počte simulačných krokov sa teda schopnosť jednotlivých agentom plniť definované úlohy zlepšuje. Po dostatočnom počte krokov simulácie by mal byť každý

z agentov schopný splniť kompletnú úlohu samostatne. Problémom je, že čas potrebný na prenos znalostí je v značnej miere závislý od toho, s akými situáciami boli agenti počas operácií v prostredí konfrontovaní. Vzhľadom na dynamiku prostredia nie je možné čas potrebný na kompletný prenos odhadnúť vopred.

6 Záver

Napriek tomu, že znalosti v komunite agentov sú individuálne a distribuované, je možné v takejto skupine agentov po istom čase o chýbajúcich znalostiach komunikovať a vzájomne si ich rozširovať. Vhodným riešením tohto problému sa javí byť koncepcia jazykových hier. Navyše, pomocou mechanizmov pre prenos znalostí je možné prispôsobiť komunitu agentov tak, aby pri operácii v spoločnom prostredí bolo umožnené distribuovanému zberu znalostí, ktoré sú po častiach zdieľané v celej skupine. Prenos znalostí vo všeobecnosti umožňuje za behu dopĺňať do znalostných štruktúr aj také informácie o konceptoch prostredia, s ktorými agent prakticky vôbec nemusí prísť do priameho styku.

7 Referencie

1. De Boer, B. Generating Vowels in a Population of Agents. *In C. Husbands and I. Harvey (Eds.), Proceedings of the Fourth European Conference on Artificial Life,* Cambridge MA and London, pp. 503-510, MIT Press, 1997.
2. Harnad, S. The Symbol Grounding Problem. *Physica D 42*, 1990.
3. Kostelník, P. Komunikácia o znalostiach v multi-robotickom tíme. In. J. Kelemen, V. Kvasnička (Eds), Kognícia a umelý život 4., Hradec nad Moravicí, 2004.
4. Oliphant, M., Batali, J. Learning and the Emergence of Coordinated Communication. *Center for Research on Language Newsletter 111*, 1997.
5. Steels, L. Perceptually Grounded Meaning Creation. *In M. Tokoro (Eds.), Proceedings of the International Conference on Multi-Agent Systems,* Menlo Park CA, AAAI Press, 1996.
6. Steels, L., Vogt, P. Grounding Adaptive Language Games in Robotic Agents. *In P. Husbands, I. Harvey (Eds.), Proceedings of the Fourth European Conference on Artificial Life, ECAL'97*, MIT Press, 1997.
7. Steels, L., Kaplan, F. Situated Grounded Word Semantics. *In Proceedings of IJCAI 99,* Morgan Kaufmann, 1999.
8. Vogt, P. *Lexicon Grounding on Mobile Robots.* Ph.D. Thesis, Vrije Universiteit Brussel, 2000.

Annotation:

Individual knowledge and its transfer in multi-robot team

In our work we provide the possible solution to the problem of formation of sufficient symbolic world representation. We propose the hierarchic control architecture that enables robot to create and use the symbolic model of the environment using the incremental learning. In this submission we present the experience with so-called *language games* implemented on the level of multi-robot interactions. Language games are interactions between agents using some language in order to transfer some knowledge about the external world. In this work, language games are used for negoriative transfer of knowledge about the common task in the multi-robot team.

Využití vývoje tématu při vylepšení odpovědi vektorového dotazu

Jan Martinovič, Václav Snášel

Katedra informatiky, FEI, VŠB - Technická Univerzita Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba

jan.martinovic@vsb.cz, vaclav.snasel@vsb.cz

Abstrakt. V dnešním světě existují stále větší kolekce dokumentů. Jednou z možností, jak v těchto kolekcích vyhledávat, je použití vektorového modelu. V článku popisujeme postup k vylepšení odpovědi získané vektorovým dotazem, ve které jsou jednotlivé získané dokumenty seřazené podle snižujícího se koeficientu podobnosti. Navrhli jsme metodu pro oddálení nerelevantních dokumentů od dotazu za pomoci dotazování na vývoj tématu. K získání informací o vývoji tématu využíváme prohledávání shluků vytvořených aglomerativním shlukováním.

Klíčová slova: shlukování dokumentů, vývoj témazu, přesnost, úplnost, dotaz, vektorový

1 Úvod

V dnešním světě existují velké kolekce textových dokumentů. S rozvojem internetu nabývají tyto kolekce stále větších rozměrů. Proto byly vytvářeny systémy, které usnadňují práci s těmito kolekcemi (*dokumentografické informační systémy*). Mezi rysy těchto velkých kolekcí patří jejich dynamičnost. Moderní přehledy problematiky [2] se dívají na textové kolekce jako na statické. Uvolnění těchto předpokladů lépe odpovídá dnešním požadavkům [1].

Pro vyhledávání v kolekcích dokumentů existují různé systémy využívajících booleovských, vektorových, pravděpodobnostních a dalších modelů k reprezentaci dokumentů, dotazů, pravidel a procedur umožňujících určit shodu mezi požadavkem uživatele (*dotaz*) a dokumenty. Každý z těchto modelů obsahuje řadu omezení. Tato omezení neumožňují uživateli nalézt všechny dokumenty, které očekává. Mezi očekávanými dokumenty (*relevantní dokumenty*) nalezneme i dokumenty špatné (*nerelevantní*) a některé relevantní dokumenty nejsou v seznamu vůbec obsaženy.

V práci popíšeme přístup k vylepšení vektorového dotazu pomocí shlukovacích metod a metod hledání vývoje tématu. Cílem metod hledání vývoje tématu je k dokumentu, který zadá uživatel, vyhledat seznam dokumentů tématicky souvisejícími se zadaným dokumentem. Práce navazuje na výsledky prezentované na konferenci Znalosti 2004 [3] a workshopu DATESO 2004 [4].

2 Hodnocení efektivity

Základními ukazateli vyhledávacích systémů jsou [8]:

- rychlost zpracování požadavků,
- uživatelský komfort, projevující se zejména ve formě interakce se systémem,
- způsob kladení dotazů a poskytování odpovědí,
- schopnost poskytnout informace o relevantních dokumentech.

Míra schopnosti poskytnout relevantní dokumenty se vyjadřuje pomocí těchto ukazatelů: *koeficient přesnosti* – P a *koeficient úplnosti* – R.

Nyní si představíme tabulku souvislostí, která nám usnadní orientaci v definicích přesnosti a úplnosti [9]. kde N je počet dokumentů v celé kolekci, A respek-

	relevantní	nerelavantní	
získané	$A \cap B$	$\bar{A} \cap B$	B
nezískané	$A \cap \bar{B}$	$\bar{A} \cap \bar{B}$	$\bar{B}$
	A	$\bar{A}$	N

Tabulka 1. Tabulka souvislostí

tive $\bar{A}$ představuje relevantní respektive nerelevantní dokumenty a B respektive $\bar{B}$ představuje získané respektive nezískané dokumenty po položení dotazu do systému. Pak definujeme:

$$P = \frac{|A \cap B|}{|B|}$$

$$R = \frac{|A \cap B|}{|A|}$$

Koeficient úplnosti lze chápat jako pravděpodobnost, že relevantní dokument byl vybrán, koeficient přesnosti jako pravděpodobnost, že vybraný dokument je relevantní. Ideální případ by byl, kdyby oba koeficienty byly rovny 1. Ukazuje se ale, že tohoto ideálního případu nelze v praxi dosáhnout.

K zjednodušení informací o efektivitě systému byly vytvořeny metody, které naměřenou přesnost a úplnost zobrazují do 1-dimenzionálního prostoru. Jednou z nich je Van Risjbergova F-míra [6]:

$$F_\beta = 1 - \frac{1+\beta^2}{\frac{\beta^2}{R} + \frac{1}{P}} = \frac{\left(1+\beta^2\right) RP}{\beta^2 P + R} = \frac{\left(1+\beta^2\right)|A \cap B|}{\beta^2|A| + |B|},$$

kde β indikuje poměrovou důležitost mezi přesností a úplností.

3 Vektorový model

Vektorový model [8] dokumentů pochází ze 70. let. Dokumenty a uživatelské dotazy jsou ve vektorovém modelu reprezentovány pomocí vektorů.

Pokud bylo pro indexaci n dokumentů použito celkem m různých termů $t_1 \dots t_m$, potom je každý dokument d_i reprezentován vektorem:

$$d_i = (w_{i1}, w_{i2}, \dots, w_{im}) ,$$

kde w_{ij} je váha termu t_j v dokumentu d_i. Váha s největší hodnotou odpovídá termu s největší důležitostí.
Indexový soubor vektorového modelu reprezentuje matice:

$$D = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{nm} \end{pmatrix} ,$$

kde i–tý řádek odpovídá i–tému dokumentu, a j–tý sloupec j-tému termu.
Dotaz ve vektorovém modelu můžeme reprezentovat pomocí m–místného vektoru vah:

$$q = (q_1, q_2, \dots, q_m) ,$$

kde $q_j \in \langle 0, 1 \rangle^m$.

Na základě dotazu q můžeme pro každý dokument d_i spočítat *koeficient podobnosti*. Tento koeficient si můžeme představit jako "vzdálenost" vektoru dokumentu a vektoru dotazu. Pro výpočet podobnosti jsme použili *kosinovou míru*:

$$sim(q, d_i) = \frac{\sum_{k=1}^{m} (q_k w_{ik})}{\sqrt{\sum_{k=1}^{m} (q_k)^2 \sum_{k=1}^{m} (w_{ik})^2}}$$

Další informace o vektorovém modelu jsou uvedeny v [8].

4 Shluková analýza

Úkolem shlukové analýzy je zjistit, zda mezi objekty existují skupiny objektů se stejnými nebo podobnými vlastnostmi. Tyto skupiny objektů se nazývají *shluky*. My se zabýváme shlukováním dokumentů, které se dá rozdělit do dvou kroků: vytvoření shluku a vyhledání relevantního shluku [5]. Spojováním podobných dokumentů do shluků lze dosáhnout zvýšení rychlosti vyhledávání ve vyhledávacích systémech. Důvod, proč se provádí analýza shluků, je obsažen v tzv. hypotéze o shlucích [8]:

Úzce vztažené dokumenty směřují k tomu, že jsou relevantní vůči týmž požadavkům.

Procesu, při kterém se hledá ideální rozklad množiny dokumentů do shluků, ve kterých jsou navzájem podobné dokumenty, se říká *shlukování*. Shluk je tedy tvořen množinou navzájem si podobných dokumentů.

Metody shlukovaní založené na matici podobnosti

Tyto metody pracují obvykle v čase $O(n^2)$ nebo vyšším (n je počet dokumentů). Podobnostní matici Sim_C pro kolekci C lze popsat takto:

$$Sim_C = \begin{pmatrix} sim_{11} & sim_{12} & \dots & sim_{1n} \\ sim_{21} & sim_{22} & \dots & sim_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ sim_{n1} & sim_{n2} & \dots & sim_{nn} \end{pmatrix},$$

kde i-tý řádek odpovídá i-tému dokumentu a j-tý sloupec j-tému dokumentu.

U těchto shlukovacích metod se vytváří hierarchie rozkladů zadaných dokumentů. V průběhu výpočtu se vytváří shlukovací hladiny, na kterých jsou body spojovány do shluků. Hierarchické metody se dají rozdělit do dvou skupin:

aglomerativní – Na startu těchto metod je každý dokument brán jako jeden shluk, postupně se dokumenty spojují (shlukují) dohromady. Výpočet je ukončen v momentě, kdy všechny dokumenty tvoří dohromady jediný shluk.

divizivní – Pracují přesně opačně než aglomerativní metody. Na startu těchto metod tedy tvoří všechny dokumenty jeden shluk. Shluky se postupně rozpadají až do chvíle, kdy je každý bod samostatným shlukem.

5 Vylepšení odpovědi ve vektorovém modelu

5.1 Definování vývoje tématu

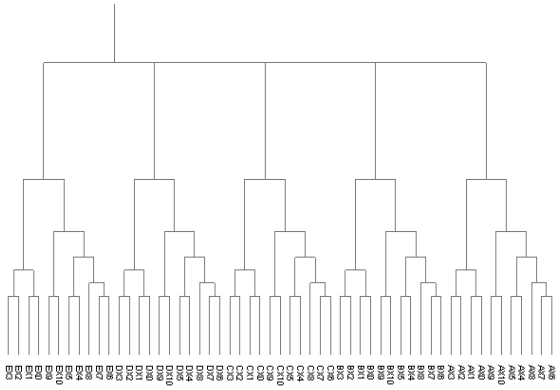
Cílem hledání vývoje tématu je k zadanému dotazu vyhledat seznam dokumentů tématicky souvisejících se zadaným dotazem. Dotazem se zde myslí, jak dotaz zadaný pomocí termů nebo také dokument, který je určen za relevantní.

Příkladem hledání vývoje tématu může být název operačního systému počítače. Zadaný dotaz bude pojednávat o nejnovější verzi operačního systému. My tento dotaz chceme rozšířit o dokumenty, které budou postupně sledovat vývoj tohoto operačního systému zpět do minulosti, k jeho prvním verzím.

Vytvoření hierarchie dokumentů pomocí shlukování nám napomáhá k zjištění vývoje tématu. Při zjišťování vývoje, procházíme hierarchii zdola-nahoru, dokud nejsou splněny vyhledávací kritéria, kterými může být například počet dokumentů vrácených uživateli, či podobnost nově do vývoje přidaného dokumentu s dotazem.

Shlukování podobných dokumentů do stejné části hierarchie shluků si ukážeme na umělé kolekci dokumentů. Umělou kolekci jsme zvoli pro, že u ní umíme přesně říct zda shlukování proběhlo podle našeho očekávání a zda lze pomocí něj sledovat vývoj dokumentů.

Umělá kolekce obsahuje 55 dokumentů. Celkový počet termů je 550. Každý dokument obsahuje 10 termů. Název dokumentů má tvar yxz.txt, kde y znamená, že dokument obsahuje slova obsahující posloupnost písmen $yyyy$, kde $y \in$ 'a'-'j'



Obr. 1. Dendogram výsledku shlukování průměrovou metodou

a končící postupně znaky 'a'-'j'. z udává, kolik slov navíc začíná znakem 'X'. Znak 'X' je přidáván nejdříve ke slovům končícím písmenem 'a', poté na 'b' až na 'j'.

Na obrázku 1 je vidět, že dokumenty, jejichž název začíná na stejné písmeno, se sloučily do společného shluku, který se dále rozpadá do podshluků podle toho, jak se mění obsah jednotlivých dokumentů.

Jako příklad si uvedeme termy několika dokumentů:

- **Dx0:** dddda ddddb ddddc ddddd dddde ddddf ddddg ddddh ddddi ddddj
- **Dx5:** Xdddda Xddddb Xdddc Xddddd Xdddde ddddf ddddg ddddh ddddi ddddj
- **Dx10:** Xdddda Xddddb Xdddc Xddddd Xdddde Xddddf Xdddg Xddddh Xddddi Xddddj
- **Bx5:** Xbbbba Xbbbbb Xbbbbc Xbbbbd Xbbbbe bbbbf bbbbg bbbbh bbbbi bbbbj

5.2 Algoritmus získání vývoje tématu

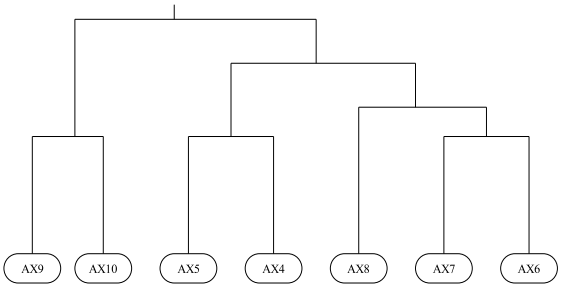
Pro získání vývoje tématu z hierarchie shluků definujeme algoritmus TOPIC, který používá počet dokumentů ve vývoji, jako omezující kritérium.

Algoritmus TOPIC:

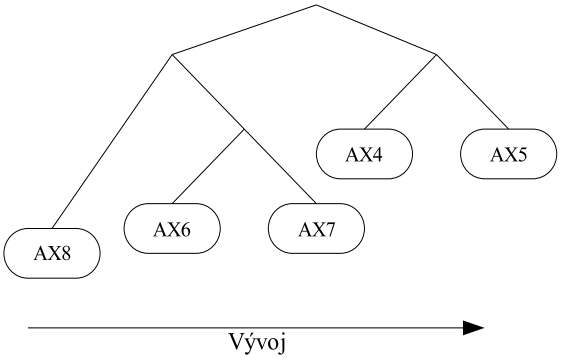
1. Určíme počet dokumentu, který chceme vrátit.
2. Nalezneme listový shluk obsahující označený relevantní dokument.
3. Postupujeme o úroveň výš v hierarchii.
4. Provedeme postupný průchod vedlejšího shluku. V průchodu nejdříve projdeme shluk vytvořený na nejbližší podhladině. Každý dokument, na který narazíme, přidáme do výsledné kolekce. Pokud počet dokumentů ve výsledné kolekci je roven požadovanému počtu dokumentů, ukončíme prohledávání.

5. Pokračujeme bodem 3.

Příklad převedeného dendogramu 2 na vývoj tématu je na obrázku 3. Vývoj je proveden pro dokument AX6 a počet požadovaných dokumentů je 5.



Obr. 2. Hierarchie dokumentů



Obr. 3. Vývoj tématu pro dokument AX8 pomocí algoritmu TOPIC

5.3 Uspořádání odpovědi ve vektorovém modelu

Odpovědí na dotaz ve vektorovém modelu je kolekce dokumentů, která je uspořádána podle koeficientu podobnosti dotazu a dokumentu. V této části si představíme metodu, která mění toto uspořádání pomocí informací o vývoji tématu zjištěných ze shluků dokumentů. Geometricky změna tohoto uspořádání znamená oddálení nerelevantních dokumentů od dotazu a přiblížení dokumentů relevantních. Pro tuto změnu uspořádání jsme navrhli dva algoritmy a pojmenovali je SORT-ONE a SORT-EACH.

Algoritmus SORT-ONE spojí kolekci získanou vektorovým dotazem s kolekcí vývoje tématu tak, že postupně přidává dokumenty z jedné a pak z druhé kolekce:

1. Provedeme vektorový dotaz a získanou kolekci dokumentů označíme C_V.
2. Vybereme dokument D_V, který nejlépe charakterizuje kolekci C_V.
3. K dokumentu D_V nalezneme pomocí algoritmu TOPIC kolekci vývoje C_T. Počet dokumentů ve vývoji bude pro jednoduchost odpovídat počtu dokumentů v celé kolekci dokumentů.
4. Označíme výslednou setříděnou kolekci C_S a určíme počet dokumentů, který má obsahovat - *count*.
5. Provedeme následující třídění:

   v_index = 0 // index prvního dokumentu v kolekci C_V
   t_index = 0 // index prvního dokumentu v kolekci C_T
   while C_S neobsahuje *count* dokumentů **do**
   while nedojde k přidání dokumentu do C_S **do**
   if C_S neobsahuje dokument $C_V[v\_index]$ **then**
   přidej dokument $C_V[v\_index]$ do C_S
   v_index = v_index + 1 // přesun na další dokument
   end
   if C_S obsahuje *count* dokumentů **then**
   ukonči třídění
   while nedojde k přidání dokumentu do C_S **do**
   if C_S neobsahuje dokument $C_T[t\_index]$ **then**
   přidej dokument $C_T[t\_index]$ do C_S
   t_index = t_index + 1 // přesun na další dokument pak
   end
   end

6. Kolekci C_S zobrazíme uživateli.

Algoritmus závisí na výběru dokumentu charakterizujícího kolekci výsledku. Tento dokument může uživatel vybrat sám z kolekce dokumentů, kterou mu nabídneme po provedení vektorového dotazu, nebo jej můžeme vybrat automaticky. Pro automatický výběr dokumentu jsme navrhli následující metodu:

– Metoda DOC-FIRST:
  Jako reprezentant se vybere dokument s největší hodnotou koeficientu podobnosti k dotazu.

Algoritmus SORT-EACH, přesune dokumenty v kolekci získané vektorovým dotazem tak, aby dokumenty ze stejného vývoje tématu byly za sebou:

1. Provedeme vektorový dotaz a získanou kolekci dokumentů označíme C_V.
2. Označíme výslednou setříděnou kolekci C_S a určíme počet dokumentů, který má obsahovat - *count*.
3. Určíme, kolik rozšiřujících dokumentů má obsahovat vývoj tématu k zadanému dokumentu. Tuto hodnotu označíme *level*.

4. Provedeme následující třídění:

foreach dokument D_V v C_V **do**
  if C_S je prázdná **then**
    vlož D_V do kolekce C_S
    goto Continue
  end
  K dokumentu D_V nalezneme pomocí algoritmu TOPIC
    kolekci vývoje C_T. Počet dokumentů ve vývoji
    bude $level + 1$ (dokument D_V).
  foreach dokument D_T v C_T mimo dokument D_V **do**
    if dokument D_T je v C_S **then**
      zařaď dokument D_V za D_T do C_S
      goto Continue
    end
  end
  if nebyl dosud dokument D_V zařazen **then**
    vlož D_V do kolekce C_S
  label: Continue
end

5. Kolekci C_S zobrazíme uživateli.

6 Testování

6.1 Postup testování

Pro testování jsme použili kolekci dokumentů nazvanou "Medlars Collection" (dostupnou na ftp://ftp.cs.cornell.edu/pub/smart). Kolekce obsahuje 1033 anglických abstraktů z oblasti medicíny (velikost 1.03 MB). Dále kolekce obsahuje sadu 30 dotazů, z nichž jsme vybrali 24, které na vektorový dotazu vracejí alespoň 100 dokumentů.

Vytvořili jsme indexy shluků dokumentů pomocí různých metod shlukování lišících se v metodě přepočtu matice podobnosti. Použité metody byly tyto: nejbližšího souseda, nejvzdálenějšího souseda, Wardova, průměrová a metoda mediánová.

Definovali jsme následující testovací metody pro úpravu vektorového vyhledávání:

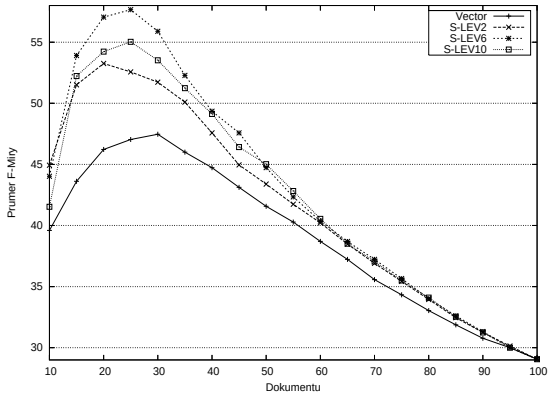
- **SORT-ONE s DOC-FIRST** (dále S-FIRST):
  Výsledek vektorového dotazu je upraven pomocí metody SORT-ONE s metodou výběru dokumentu DOC-FIRST.
- **SORT-EACH s** $level$**=2, s** $level$**=6 a s** $level$**=10** (dále S-LEV2, S-LEV6 a S-LEV10):
  Výsledek vektorového dotazu je upraven pomocí metody SORT-EACH s $level$=2, s $level$=6 nebo s $level$=10.

– **Rozšíření pomocí KLD** (dále E-KLD):
Nalezneme metodou UP-DONW-2 (viz. [3]) shluk nejvíce podobný dotazu. V tomto shluku pomocí metody KLD (viz. [3]) nalezneme 5 rozšiřujících termů a přidáme je k původnímu dotazu. Pomocí tohoto nového dotazu provedeme vektorový dotaz.

Pro vytvořené hierarchie shluků jsme provedli následující test s každou výše uvedenou metodou:

1. Metodu jsme provedli na každém z 24 dotazů.
2. Pro každý výsledek dotazu jsme spočítali hodnotu přesnosti a úplnosti a z - nich F-míru s $\beta = 1$ pro prvních 5,10, ... 100 získaných dokumentů.
3. Vypočítali jsme průměrnou hodnotu F-míru ze všech dotazů pro prvních 5,10, ... 100 získaných dokumentů.

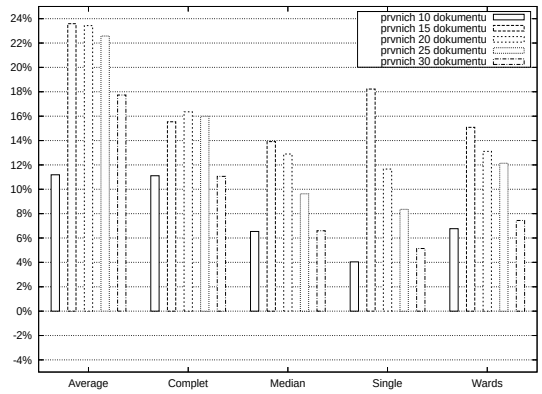
6.2 Zhodnocení výsledků testování



Obr. 4. Srovnání vektorového dotazu s metodami S-LEV2, S-LEV5 a S-LEV6 u indexu vytvořeného metodou průměrů.

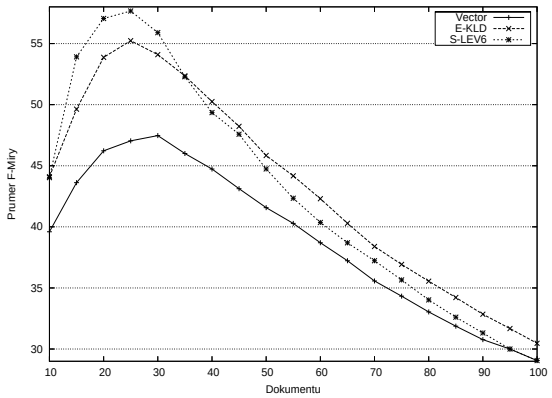
Testy jsme provedli s metodami S-LEV2, S-LEV6 a S-LEV10, které provádí přeskládání výsledku vektorového dotazu podle vývoje dokumentů zjišťovaného ve shlucích. Ukázalo se že pomocí tohoto postupu lze dosáhnout zlepšení až k 22% F-míry oproti původnímu vektorovému dotazu. Toto zlepšení se projevuje v testech provedených u prvních 10-30 dokumentů. Vzájemným srovnáním jednotlivých metod jsme zjistili, že nejlepších výsledků se dosáhlo u metody S-LEV6. Zhoršení u metody S-LEV10 lze přisoudit velikosti kolekce dokumentů, kdy při 1033 dokumentech, shluky které obsahují samostatná téma jsou malé. Příklad grafu srovnávajícího jednotlivé metody u indexu vytvořeného metodou průměrů je na obrázku 4. Procentuální vylepšení F-míry metodou S-LEV6 oproti

původnímu vektorovému dotazu u jednotlivých shlukovacích metod je uvedeno na obrázku 5. Je zde patrné, že nejlepší výsledky se dosahují opět u metody průměrů. Dále se jako vhodná jeví metoda nejvzdálenějšího souseda a Wardova metoda.



Obr. 5. Srovnání shlukovacích metod s metodou S-LEV6.

Dalším testem, který jsme provedli bylo srovnání výše uvedených testů u metod S-LEV6 a E-KLD. Jako příklad uvádíme graf na obrázku 6. Z grafu je patrné, že výsledky obou metod jsou srovnatelné. U metody S-LEV6 oproti E-KLD nemusíme provádět prohledávání stromu shluků.



Obr. 6. Srovnání vektorového dotazu s metodami E-KLD a S-LEV6.

7 Závěr

Z výsledků provedených testů vyplývá, že pomocí metod, které přeskládávají výsledek vektorového dotazu za pomoci vývoje tématu zjišťovaného v hierarchiích shluků lze dosáhnou až 22% zlepšení F-míry. Výraznější zlepšení oproti původnímu vektorovému dotazu se nejvíce projevuje v prvních 10–30 získaných dokumentech.

Z popsaných shlukovacích metod se ukázaly nejlepší metody, u kterých nedochází ke tvorbě řetězců. V případě, že při vytváření hierarchie shluků dochází k tvorbě řetězů je velikost indexu velká. Nejlepší se jeví vytváření hierarchií shluků pomocí metody průměrů, která v námi provedených testech dosahovala nejlepších výsledků.

V dalším výzkumu bychom chtěli provést srovnání shlukovacích metod, které provadí shlukovaní celé kolekce dokumentů a metod, které provádí shlukování až po provedení dotazu s kolekcí dokumentů, které daný dotaz vrátí [7, 10]. Dále se chceme věnovat redukci časové složitosti výpočtu shluků pomocí metod náhodné projekce [11].

Reference

1. Berry, M. W (Ed.): Survey of Text Mining: Clustering Classification, and Retrieval. Springer Verlag 2003.
2. Baeza-Yates R., Ribeiro-Neto B.: Modern Information Retrieval. Addison Wesley, New York, 1999.
3. Dvorský J., Martinovič J., Pokorný J., Snášel V.: Vyhledávání témat v kolekci dokumentu, Znalosti 2004.
4. Dvorský J., Martinovič J., Snášel V.: Query Expansion and Evolution of Topic in Information Retrieval Systems, DATESO 2004.
5. Christis Faloutsos, Dayglas Oard: A Survey of Information Retrieval and Filtering Methods, University of Maryland, College Park, MD 20742.
6. Tsunenori Ishioka: Evaluation of Criteria for Information Retrieval, The National Center for university Entrance Examinations, Japan.
7. Kummamuru K, Lotlikar R., Roy S., Singal K., Krishnapuram R.: A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results, WWW2004, New York, USA.
8. Pokorný J., Snášel V., Húsek D.: Dokumentografické informační systémy. Karolinum, Skriptum MFF UK Praha, 1998.
9. C.J. van Rijsbergen: Information Retrieval (second ed.). London, Butterworths, 1979.
10. Keith Van Rijsbergen: The Geometry of Information Retrieva, Cambridge University Press, 2004.
11. S. Vempala: The Random Projection Method, Dimacs Series in Discrete Mathematics and Theoretical Computer Science, 2004.

Anotation:

Using topic evolution to improve result of vector query

There are large collections of documents in today's world. One way as in those collections search is by using vector model. This article described the approch to improve result of vector query. In vector query are documents sorted by similarity to query. We are proposed method for distancing of nonrelevant documents from the query, with the help of questioning on the topic evolution.

Rozhodovací stromy pro distribuce a jejich vizualizace

Petr Máša

Katedra informačního a znalostního inženýrství VŠE

petrmasa@yahoo.com

Abstrakt. Rozhodovací stromy jsou často využívány pro svou jednoduchost a vizualizaci. Ve většině případů jsou však studovány stromy s diskrétní cílovou proměnnou. Pro mnohé metody existuje i rozšíření na spojitou cílovou proměnnou. V případě spojitých proměnných nás obvykle zajímá nejen střední hodnota či jiný ukazatel, ale zejména distribuce. Tento příspěvek definuje úpravu algoritmu pro učení rozhodovacího stromu právě s ohledem na distribuci spojité proměnné. Druhou, neméně podstatnou úlohou při použití spojité cílové proměnné, je vizualizace, která je zpravidla pro spojité proměnné v běžných SW vyřešena tak, že do uzlu je vepsán počet pozorování, průměr a směrodatná odchylka. Opět, obvykle nás zajímají nejen tyto tři údaje, ale jaká je v uzlu distribuce cílové spojité proměnné. Příspěvek navrhuje také vizualizaci spojité proměnné pomocí histogramů a možnosti využití znalosti distribuce v uzlech pro predikci[1].

Klíčová slova : rozhodovací stromy, cílová spojitá proměnná, distribuce, vizualizace.

1 Úvod

Základem vytváření rozhodovacích stromů je tzv. hladový (*greedy*) přístup, který začíná s celým souborem a hledá nejlepší atribut pro dělení (maximalizuje[2] se tzv. dělící kriterium – *splitting criterion*). Po rozdělení na dvě či více částí[3] se postupuje stejným způsobem na jednotlivých částech až do dosažení zastavovacích kriterií (*stopping criteria*).

V základních variantách se vyskytuje kategoriální cílová proměnná a kategoriální prediktory. Většina základních metod byla přizpůsobena pro práci se spojitými prediktory spočívající obvykle v rozdělení množiny podle hodnoty spojitého atributu na dvě kategorie v závislosti na tzv. dělícím bodu (*cutoff point*).

[1] Tato práce byla podporována projekty GAČR 201/05/0325: Nové metody a nástroje pro dobývání znalostí z databází a VŠE - IGA 17/04: Využití metod znalostního inženýrství pro aplikace a rozvoj systému LISp-Miner

[2] v některých případech se dané kritérium formálně minimalizuje, to lze však převést na maximalizaci opačné hodnoty

[3] závisí na konkrétním algoritmu

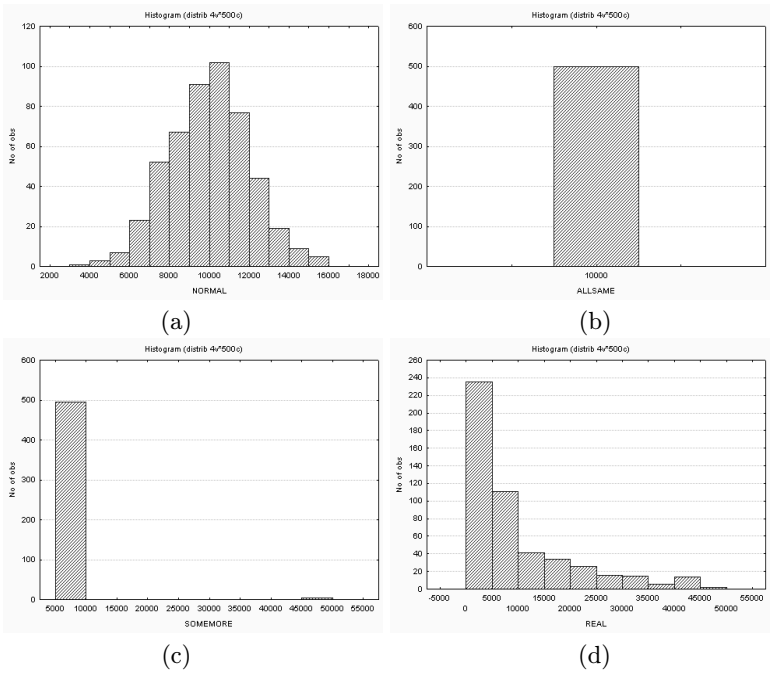
1. *if* není dosaženo zastavovacích kritérií
2. spočti pro atributy $a_1, \ldots, a_m$ hodnotu dělícího kritéria $sc(a_1), \ldots, sc(a_m)$
3. vyber atribut a_i s nejvyšší hodnotou $sc(a_i)$ (tj. $sc(a_i) \geq sc(a_j) \forall j$)
4. rozděl množinu všech pozorování na části dle hodnoty atributu a_i
5. na každé části opakuj stejný postup
6. *else*
7. z dané množiny utvoř list
8. *end if*

Obrázek 1. Hladový algoritmus pro učení rozhodovacího stromu

Tento bod je vybírán tak, aby bylo dělící kriterium maximalizováno. Přizpůsobení spojité cílové proměnné je však zpravidla obtížnější, vyžaduje totiž obvykle změnu dělícího kritéria. Také některé rozšířené komerční implementace rozhodovacích stromů neobsahují podporu pro spojitou cílovou proměnnou. Podívejme se na problém spojité cílové proměnné podrobněji. Pro kategoriální cílovou proměnnou můžeme odhadovat cílovou kategorii (klasifikace) či sledovat počty pro jednotlivé kategorie v jednotlivých listech (objevování znalostí). Při použití spojité proměnné můžeme opět odhadovat hodnotu cílové proměnné (zpravidla však průměrem, což vede na omezenou množinu predikovaných hodnot) či sledovat charakteristiky cílové proměnné v jednotlivých listech. Pro spojité proměnné je zpravidla u listů uveden průměr, případně směrodatná odchylka. V lepším případě je v listech uvedena jedna či několik jiných charakteristik, například medián či modus, případně je přidána vizualizace založená na základě některých z uvedených veličin (např. normální křivka z průměru a směrodatné odchylky či zjednodušený boxplot). Charakterizují však tyto základní ukazatele hodnoty cílové proměnné opravdu výstižně? V dalších částech se budeme snažit ukázat, že cílem (zejména pro objevování znalostí či vizualizaci stromu) by mělo být hledání distribuce cílové proměnné, nikoliv její jediné charakteristiky. Také bude navržen postup, jak lze daného cíle dosáhnout.

2 Průměr není vše

Průměr neboli střední hodnota je základní charakteristika spojité veličiny. Je pravděpodobně nejrozšířenější a často i jedinou používanou charakteristikou. Cílem by však nemělo být hledat právě průměr, ale rozložení atributu (distribuci). Podívejme se na následující příklad, převzatý z [4]. Mějme 4 populace a u nich budeme sledovat průměrný plat (viz obrázek 2). První z nich (a) je statistické normální rozložení platů (pokud by existovala jediná činnost). Kdo dělá více práce, má i více peněz. Průměrný plat je 10 000 Kč. Druhou z nich (b) je utopický komunismus. Všichni mají stejný plat. Opět je průměrný plat 10 000 Kč. Třetí (c) je reálný komunismus dle hesla všichni jsou si rovni a někteří jsou si rovnější. Standardní lidé mají standardní konstantní plat a „rovnější“ lidé mají plat 50 000 Kč. Průměr je opět 10 000 Kč. Konečně čtvrtá distribuce (d) znázorňuje reálné, silně asymetrické rozložení platů. Opět je průměr 10 000 Kč. Zde



Obrázek 2. Rozložení platů ve čtyřech populacích. Průměr je ve všech případech 10 000 Kč.

máme čtyři různé populace, s naprosto rozdílnými charakteristikami, avšak průměr je stejný. To, co bychom měli hledat, není průměr, ale distribuce. Pro další analýzy budeme uvažovat data ve struktuře, jak je naznačeno na obrázku 1. Pro

Tabulka 1. Ukázka struktury dat

Segment	Plat
ALLSAME	10 000,00
NORMAL	9 432,50
ALLSAME	10 000,00
ALLSAME	10 000,00
REAL	4 254,30
SOMEMORE	50 000,00
SOMEMORE	9 595,96
...	...

každý ze 4 segmentů budeme mít 500 pozorování, jejichž průměr je 10 000 Kč a jejichž distribuce je znázorněna na obrázku 2.

V předchozím odstavci jsme si ukázali, že průměr jakožto jediná charakteristika není vše. V některých případech je kromě průměru uváděna i směrodatná odchylka, která je nejčastěji používanou mírou vzdálenosti od průměru[4]. Pokud použijeme spolu s průměrem i směrodatnou odchylku, nezískáme tím principiální zlepšení, neboť ze dvou charakteristik je jednoznačně definována jen distribuce z nejvýše dvou hodnot. Tedy mohli bychom ukazovat podobné příklady.

3 Učení rozhodovacích stromů podle distribuce

Pro učení rozhodovacích stromů lze použít standardní greedy algoritmus uvedený v úvodu tohoto příspěvku s tím, že jako dělící kritérium je brána míra odlišnosti distribucí.

Míru odlišnosti založíme na (p-)hodnotě Kolmogorovova-Smirnovova (KS) testu pro dvě distribuce (budeme dělit soubor na dvě části). KS test je ve statistice využíván jako test shody dvou distribucí. Testová statistika je dána vztahem

$$D_{m,n} = \sup_x \|F_m(x) - G_n(x)\|,$$

kde $F_m(x)$ a $G_n(x)$ jsou empirické distribuční funkce pro první a druhou skupinu. Jejich výpočet je velmi jednoduchý – pro dané x ji spočítáme tak, že vydělíme počet pozorování nabývající hodnotu cílové proměnné menší nebo rovnu x počtem pozorování (vše v rámci dané skupiny – potenciálního uzlu).

Použití přímo hodnoty $D_{m,n}$ však preferuje rozdělení na velmi malou množinu a zbytek. Proto zahrneme do dělícího kriteria opravu pro poměr velikosti

[4] směrodatná odchyla je odmocnina ze střední čtvercové vzdálenosti od průměru

jednotlivých skupin. Tato oprava je převzata z aproximace p-hodnoty KS testu[5] vycházející ze Smirnovovy věty (viz [3]). Dělící kriterium Sp je definováno jako

$$Sp = D_{m,n}^2 \frac{mn}{m+n} = (\sup_x \|F_m(x) - G_n(x)\|)^2 \frac{mn}{m+n},$$

kde m, n je počet prvků v porovnávaných skupinách – potenciálních uzlech. Toto kriterium se snažíme maximalizovat.

Výsledné stromy budou binární. Dělíme-li podle nominálního atributu s více kategoriemi, vyzkoušíme všechny možnosti rozdělení na dvě skupiny a vybereme to dělení, kde je hodnota kritéria nejvyšší.

Tento postup lze upravit i na nebinární stromy. K tomu lze zaujmout dvě strategie.

1. Pokud dělíme podle atributu a_i a další dělení v některém listu podle stejného atributu a_i je dostatečně významné (použijeme stejný statistický test), má smysl toto dělení zahrnout již na vyšší úroveň, čímž získáme obecně n-ární strom.
2. Daný list rozdělíme podle všech hodnot nejvýznamnější proměnné a na základě stejného statistického testu slučujeme ty kategorie, kde jsou distribuce shodné.

Podívejme se zde na jednu zajímavou vlastnost našeho kritéria. Uvedené kriterium nabývá nejvyšší možné hodnoty právě tehdy když dochází k separaci hodnot cílového atributu, tj. existuje γ tž. hodnoty cílové proměnné v jednom uzlu jsou větší než γ a hodnoty cílové proměnné ve druhém uzlu jsou menší než γ. Tato vlastnost lze snadno odvodit z definice kritéria. Dále platí, že čím blíže je hodnota daného kritéria svému maximu, tím méně je třeba vynechat prvků, aby k separaci hodnot cílového atributu došlo.

4 Srovnání s jinými metodami

Uvažujme náš dataset s proměnnou určující typ populace a hodnotami pro jednotlivé populace. Podívejme se, jak se na něm zachovají standardní metody (budeme uvažovat CART a CHAID) v konkrétní SW implementaci[6].

Některé metody provedou dělení, jiné nikoliv, v závislosti na dělícím kritériu, které používají. Například CART používá jako dělící kriterium pro spojité cílové proměnné největší redukci v součtu čtverců. Proto lze nalézt protipříklad, kdy budou distribuce principiálně odlišné, avšak nedojde k dělení. V příspěvku byl použit Kolmogorovův-Smirnovův test, který zkoumá právě odlišnost distribucí. Proto je dané kritérium optimální (z hlediska greedy algoritmu), pokud je naším cílem sledovat rozdělení (distribuci) cílové proměnné.

[5] a je tedy teoreticky korektní pro velká m, n – to však není na závadu, neboť to odpovídá požadavku na minimální počet prvků v každém uzlu

[6] byl použit SW balík Statistica, verze 6; v jiných SW balících dojdeme k obdobným výsledkům

Tabulka 2. Standardní metody v SW balíku se standardním nastavením

Metoda	Výsledek
CART	dvě dělení, prořezání vede zpět na 1 list
CHAID	nedojde k dělení

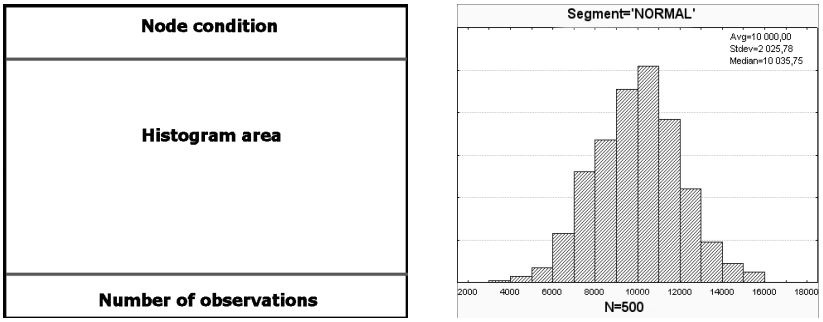
Na reálných datech se často stává, že nejrozšířenější kriteria CHAID a CART volí stejné atributy pro dělení, zejména na nejvyšších úrovních. Volba hodnoty, podle které se dělí do jednotlivých listů v případě spojitého prediktoru, se však liší. Jen ve specifických případech jiné volí atributy pro dělení. Podobně je tomu i v případě navrženého kriteria. Na nejvyšších úrovních se obvykle volí stejné atributy, podle kterých se bude dělit[7]. K odlišným a užitečným výsledkům dochází, pokud cílová proměnná nabývá v obou potenciálních listech podobných hodnot, avšak s jinou distribucí (např. opačně sešikmené).

V uvedeném modelovém příkladě je uvažována směs distribucí. V reálných datech se často tato modelová situace vyskytuje – kategoriálním atributem dobře odlišující distribuce bývá segment zákazníků. Každý segment má pak jinou distribuci cílové proměnné, i když některé ukazatele (jako např. průměr či rozptyl) jsou podobné. Příkladem může být výše účtu za telefonní služby u operátora. Existuje segment zákazníků, kteří mají sjednaný nějaký tarif s volnými minutami a jejich útrata se rovná právě měsíčnímu paušálu (volné minuty nejsou vyčerpány). Toto nastává u různých výší tarifů a tento segment tedy obsahuje několik konkrétních diskrétních čísel. Většina ostatních segmentů pak obsahuje „klasické“ rozložení spojité veličiny, tj. nejedná se jen o několik izolovaných kategorií. Pro popis či predikci útraty zákazníka se také využívají spojité atributy. Příkladem takového atributu je poměr přenesených dat ku době připojení u datových služeb tarifikovaných dle doby připojení. Někteří zákazníci se připojují na krátkou dobu s tím, že se snaží za tuto dobu přenést co nejvíce dat. Jiní se připojí na delší dobu, přičemž je objem přenesených dat malý. Tento prediktor (resp. jeho diskretizace) také ovlivňuje typ distribuce atributu výše měsíčního účtu za telefon. Tedy v reálných datech o zákaznících se vyskytují směsi distribucí podobné našemu modelovému příkladu velice často. Důvodem jsou nastavení jednotlivých produktů, jako např. omezená sada tarifů, tabulka určující výši pojistného podle několika mála atributů či předdefinované částky pro výběr z bankomatů.

5 Vizualizace výsledků

Pro vizuální zobrazení rozhodovacích stromů se spojitou cílovou proměnnou (nejen při rozpadu podle distribuce) je vhodné používat v každém listu histogram, nejen textový údaj o průměru, příp. směrodatné odchylce cílové proměnné. Zobrazení jednotlivého listu je uvedeno na obrázku 3.

[7] tedy navržené kriterium nepopírá použitelnost přibližně 20 let používaných metod



(a) (b)

Obrázek 3. Uzel stromu: (a) layout uzlu; (b) konkrétní uzel.

Vezměme náš dataset a zobrazme (již rozpadlý vzhledem ke 4 distribucím) ho pomocí obvyklého zobrazení v SW – obrázek 4(a) a pomocí zobrazení s histogramy – obrázek 4(b).

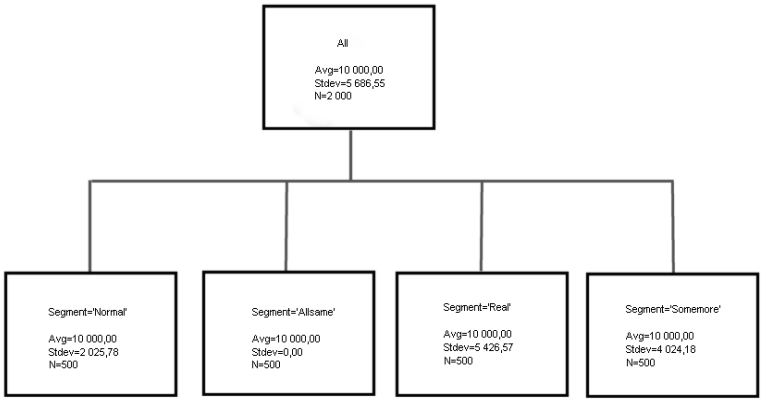
Zobrazení s histogramem dává uživateli více informací, navíc obvykle používanou grafickou cestou – zobrazení distribuce hodnoty v rámci skupiny. Pro vizualizaci by měl být histogram dělen pokud možno co nejjemněji (viz např. [5]), neboť jinak můžeme dojít k zavádějícím výsledkům. Pro snadnou srovnatelnost v rámci skupin je vhodné mít ještě možnost používat společné měřítko na vodorovné ose ve všech uzlech, případně možnost sjednotit měřítko v uzlu a všech jeho potomcích.

6 Využití distribuce pro predikci/regresi

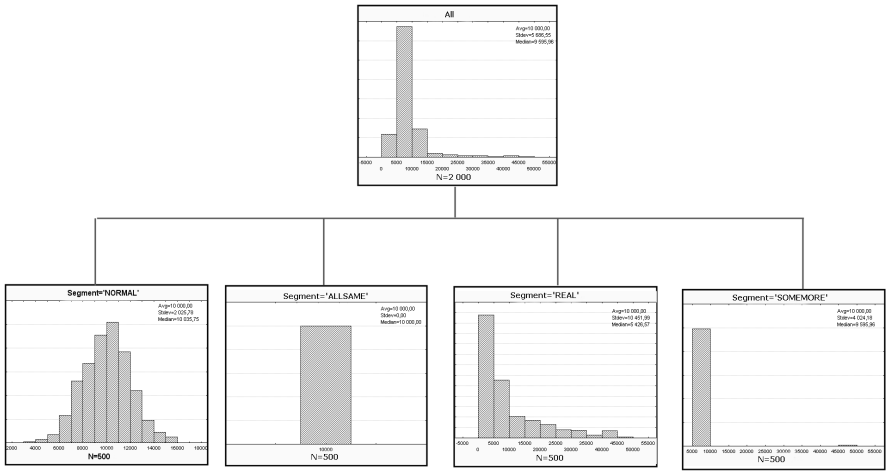
V příspěvku bylo předvedeno dělení rozhodovacích stromů s ohledem na distribuci a dále vizualizace stromů s ohledem k distribuci cílové proměnné. Pokud budeme používat rozhodovací strom pro predikci či regresi cílové proměnné, pak nás pro nový záznam obvykle zajímá nejen odhadovaná průměrná hodnota, ale i její distribuce. Můžeme se tedy ptát, s jakou pravděpodobností přesáhne hodnota cílové proměnné zadanou mez. To je zajímavé zejména v případech, kdy cílová proměnná odráží riziko. Tato vlastnost standardním implementacím stromů chybí. Právě otázka, s jakou pravděpodobností přesáhne hodnota v uzlu danou mez x_0 pro každé $x_0 \in \mathbb{R}$ je jen jiný náhled na distribuci proměnné.

7 Závěr

Při použití spojitých cílových proměnných nás často zajímá rozdělení této proměnné, nejen průměr či směrodatná odchylka. Tomuto cíli by mělo být přizpůsobeno dělící kritérium a také vizualizace rozhodovacího stromu. V textu byl uveden přístup, který řeší jak vizualizaci stromů se spojitou cílovou proměnnou



(a)



(b)

Obrázek 4. Rozhodovací strom se spojitou cílovou proměnnou: (a) obvyklé zobrazení v SW – v uzlech zobrazeno jen několik málo kvantitativních ukazatelů; (b) zobrazení pomocí histogramů, s možností zobrazit také dané kvantitativní ukazatele.

(pomocí histogramů v jednotlivých uzlech), tak i způsob dělení listů vzhledem k této distribuci. Byl uveden i přístup k predikci využívající právě distribuci cílové proměnné.

Dělící kritérium, vizualizace a predikce byly 3 uvedené oblasti, kde distribuce cílové proměnné sehrává významnou roli (může mít přínos oproti použití jen střední hodnoty, případně směrodatné odchylky). Tyto oblasti byly řešeny obecně, tedy např. daná vizualizace lze využít i v případě použití standardních dělících kritérií.

Reference

1. Berry, M., Linoff, G.: Data Mining Techniques: For Marketing, Sales, and Customer Support, John Wiley & Sons, 1997
2. Murthy, S., Salzberg, S.: Investigations of the greedy heuristic for classification tree induction, `http://citeseer.ist.psu.edu/130352.html`
3. Anděl, J.: Statistické metody, Matfyzpress 1998
4. Fabián, Z.: Z Čech až na konec hotovosti, Bulletin České statistické společnosti č. 4/1997
5. Theus, M.: 1001 Graphics, in Antoch,J.: Compstat, proceedings in Computational Statistics, Springer 2004
6. Statistica 6 Electronic Manual, Statsoft 2003
7. AnswerTree Algorithm Summary, SPSS White Paper, 1999

Annotation.

Decision Trees For Distributions and Their Visualization

Decision trees are widely used technique for its simplicity, easy understanding and easy visualization. Most methods are focused to discrete target variable, but some of them are extended to continuous target. In case of continuous target, we should be interested in its distribution, not only the mean and standard deviation respectively. This article is focused on explanation, why distribution should be object of exploration, defines splitting criterion which respects distribution as a goal, defines layout for vizualization of distribution in nodes (and shows difference comparing to standard layout used in major software packages) and mentions some benefits of knowing the distribution when predicting continuous variable (more tasks can be solved). Many available vizualizations of continuous target tries to approximate leaf distribution from some summary statistics (for example, normal curve from mean and standard deviation). This can be misleading when interpreting specific leaf. Benefits of using real target distribution (not some approximation) instead of some statistics are also mentioned.

Využití LSI a M-stromu při indexování a vyhledávání obrázků

Tomáš Skopal[1], Michal Kolovrat[2], Václav Snášel[2]

[1]Katedra softwarového inženýrství, MFF, UK Praha,
Malostranské nám. 25, 118 00, Praha
[2] Katedra informatiky, FEI, VŠB – Technická Univerzita Ostrava,
17. listopadu 15, 708 33, Ostrava–Poruba
tomas@skopal.net, {michal.kolovrat,vaclav.snasel}@vsb.cz

Abstrakt. Při práci s multimediálními dokumenty musíme často řešit problém, jak v nich efektivně a rychle vyhledávat. Multimediální dokumenty většinou reprezentujeme vektory ve vysokodimenzionálním prostoru, neboť v takto modelované kolekci dokumentů lze jednodušeji definovat sémantiku i mechanismus samotného vyhledávání. V tomto příspěvku prezentujeme vytváření vektorů obrázků (extrakci vlastností z obrázků) v modelu LSI (resp. pomocí singulárního rozkladu). Současně ukážeme vliv takové extrakce na efektivitu indexování/vyhledávání pomocí datové struktury M-strom. Vzhledem k aplikaci modelu LSI na obrázky poukážeme rovněž na některé zajímavé souvislosti, které při klasickém použití LSI na textových kolekcích nejsou přímo patrné.

Klíčová slova: LSI, podobnostní vyhledávání obrázků, M-strom

1 Úvod

V současné době, s nárůstem informačních technologií, vzniká obrovské množství multimediálních dat – mezi tato data patří zejména textové, obrazové a zvukové dokumenty.

Multimediální data je potřeba nějakým způsobem vhodně reprezentovat a organizovat, aby v nich následně šlo jednoduše a rychle vyhledávat [6]. Jednotlivé multimediální dokumenty většinou reprezentujeme vektory ve vysokodimenzionálním prostoru, neboť v takto modelované kolekci dokumentů lze jednodušeji definovat sémantiku i mechanismus samotného vyhledávání podle obsahu (content-based search). V současnosti se stále ve velké míře používá sekvenční způsob vyhledávání, kdy je celá kolekce dokumentů (resp. její reprezentace, tj. kolekce vektorů) postupně sekvenčně načtena a hledané dokumenty jsou nacházeny pomocí vzájemného porovnáváním všech dokumentů v kolekci s dotazovým dokumentem (pomocí nějaké funkce podobnosti, definované pro dva vektory dokumentů). Tento způsob vyhledávání však již v současné době není možný, neboť kolekce dokumentů jsou velmi rozsáhlé a sekvenční průchod takovou kolekcí by trval neúnosně dlouho. Proto byly navrženy přístupy a metody, které nejdříve

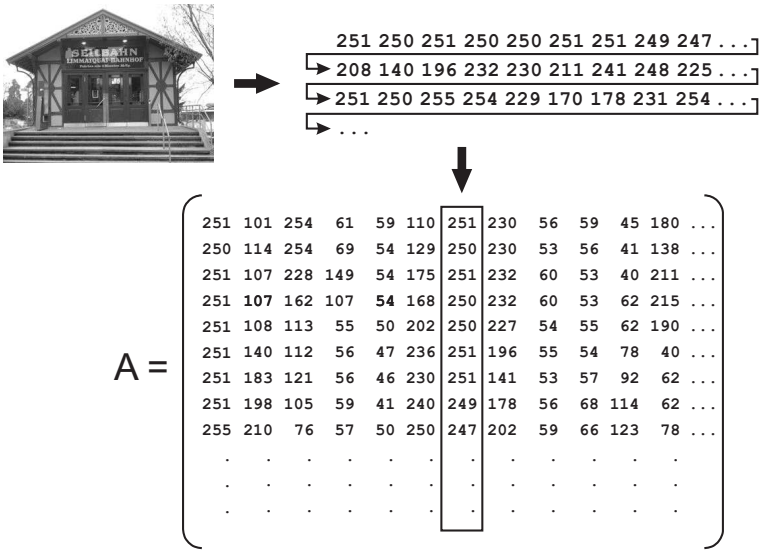
kolekci předzpracují do předem navržené struktury – indexu. Cílem indexování je tedy umožnit rychlé vyhodnocení uživatelského dotazu pomocí indexu. Na Obr. 1 uvádíme vzorek z experimentální kolekce 730 obrázků budov (posléze transformovaných do velikosti $80 \cdot 60$ pixelů v 256 odstínech šedi).

Obr. 1. Vzorek z kolekce obrázků budov.

2 Extrakce vektorů vlastností pomocí LSI

Metod pro extrakci vlastností z obrázků (a tím vytvoření jejich vektorové reprezentace) existuje celá řada. Nejčastěji jsou používány různé histogramy barev [3], textur [17], apod. Pro doménově specifické kolekce (např. otisky prstů, oční duhovky, obličeje) pak existuje mnoho speciálních přístupů. Pro přehled současných metod extrakce vlastností z obrázků odkazujeme na [13].

V našem příspěvku jsme použili jiný přístup [11, 12], kdy je celý obrázek velikosti $x \cdot y$ (resp. jasy/stupně šedi všech jeho pixelů) předeven na vektor dimenze $x \cdot y$. Transformace obrázku do vektoru je provedena zcela triviálně, a to "naskládáním" řádků obrázku za sebe, čímž vznikne vysokodimenzionální "jasový" vektor obrázku. V této fázi reprezentace obrázku ještě nejde o extrakci vlastností, podle kterých by se dalo posléze smysluplně vyhledávat, obrázek (resp. jeho jasová verze) lze bezezbytku zrekonstruovat z jasového vektoru. Nicméně díky tomu, že obrázky jsou reprezentovány jasovými vektory, můžeme celou kolekci obrázků (o velikosti n) reprezentovat maticí A řádu $x \cdot y \times n$, kde jednotlivé vektory obrázků tvoří sloupce matice (viz Obr. 2, $x = 80$, $y = 60$, $n = 730$, tj. matice A je řádu 4800×730).



Obr. 2. Transformace obrázku do vektoru jasů a jeho umístění v matici A.

2.1 Klasické LSI jako rozšíření vektorového modelu DIS

V oblasti *Dokumentografických informačních systémů* (Information Retrieval) [1, 10] se s modelem LSI (latent semantics indexing) setkáváme v kontextu reprezentace a indexování textových dokumentů ve vektorovém modelu [7, 2]. Textová kolekce obsahuje dohromady m unikátních termů, přičemž m-rozměrné vektory dokumentů (sloupce matice A) reprezentují frekvence/váhy jednotlivých termů v textových dokumentech. Pomocí singulárního rozkladu (SVD) matice A:

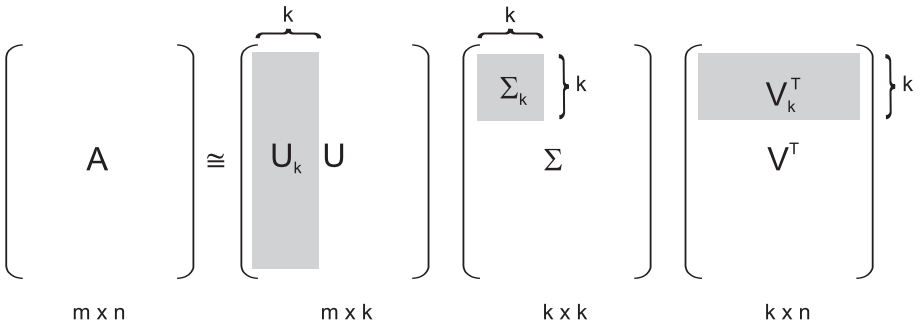
$$A = U\Sigma V^T$$

se v textové kolekci naleznou tzv. *vektory konceptů* (levé singulární vektory – sloupce v U), které lze interpretovat jako jednotlivá témata přítomná v kolekci. Vektory konceptů tvoří bázi v původním vysokodimenzionálním prostoru a jsou to v zásadě lineární kombinace termů (termy jsou nezávislé). Důležitou vlastností singulárního rozkladu je fakt, že vektory konceptů jsou uspořádány podle "důležitosti" (důležitost vyjadřují hodnoty singulárních čísel σ_i, uložené sestupně v diagonální matici Σ) – podle toho, v jaké míře a v kolika dokumentech se (ne)vyskytují. Tím je určeno, které koncepty jsou vzhledem ke kolekci sémanticky důležité (odtud indexování *latentní* (skryté) sémantiky), a které nedůležité, ty působí jako statistický šum. Sloupce matice ΣV^T obsahují vektory dokumentů (tzv. *vektory pseudo-dokumentů*), které jsou ale nyní vyjádřené v bázi U, tj. v *bázi konceptů*. Jeden vektor pseudo-dokumentu odpovídá nějaké lineární kombinaci všech nalezených vektorů konceptů, tj. příslušný dokument v nějaké míře (třeba i nulové nebo záporné) obsahuje každý z konceptů.

Vzhledem k tomu, že pouze prvních k konceptů lze považovat za sémanticky důležité (příslušná singulární čísla jsou vysoká), můžeme původní rozklad aproximovat jako

$$A \approx U_k \Sigma_k V_k^T$$

kde v U_k je prvních k nejdůležitějších vektorů konceptů, v Σ_k jsou příslušná singulární čísla a ve $\Sigma_k V_k^T$ vektory pseudo-dokumentů vyjádřené pouze pomocí prvních k vektorů konceptů (viz Obr. 3). Jinými slovy, SVD promítá původní m-rozměrné vektory dokumentů do prostoru dimenze k ($k \ll m$). Aproximace SVD rozkladu (rank-k SVD) matice A lze dosáhnout tak, že buď provedeme úplný SVD rozklad a posléze příslušné matice "ořežeme", anebo zvolíme některou z časově méně náročných numerických metod, která počítá přímo redukovaný rozklad (např. Lanczos, Arnoldi). Pokud chceme v kolekci dokumentů vyhledávat,



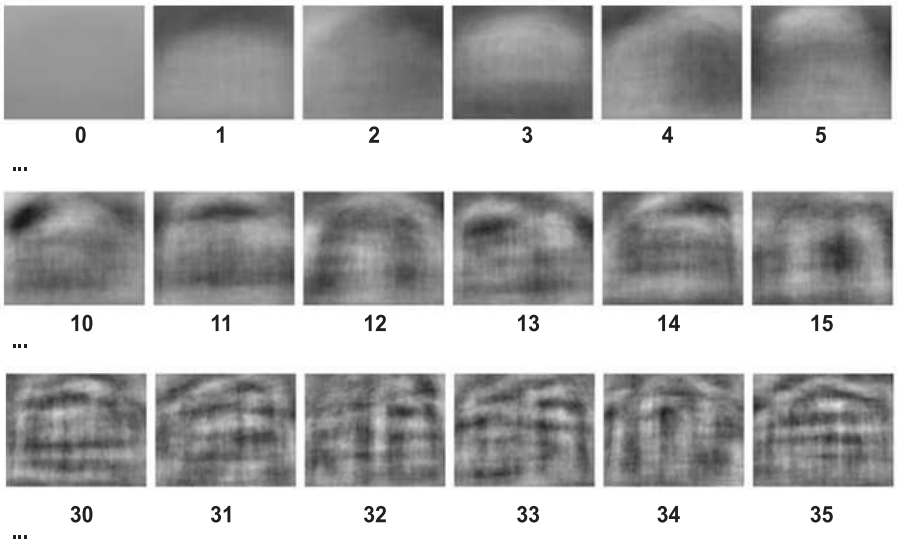
Obr. 3. Aproximace SVD rozkladu (rank-k SVD).

porovnáváme vektory pseudo-dokumentů s *vektorem pseudo-dotazu* na základě nějaké míry podobnosti (v klasickém vektorovém modelu DIS i v modelu LSI pro textové kolekce je to např. kosinová míra). Abychom mohli porovnávat vektory pseudo-dokumentů, potřebujeme z klasického vektoru dotazu q (reprezentovaného stejným způsobem jako jsou vektory dokumentů v A) zkonstruovat jeho projekci do U_k, tj. $U_k^T q$

2.2 LSI pro kolekce obrázků

Vzhledem k tomu, že SVD rozklad můžeme aplikovat na libovolnou matici, je model LSI aplikovatelný na libovolné typy dat. Interpretace matice A jako matice termů v dokumentech je pouze předmětem aplikace, tj. aplikace LSI jako rozšíření klasického vektorového modelu DIS. Obecně tedy můžeme model LSI adaptovat pro indexování libovolné multimediální kolekce, kterou lze reprezentovat sadou vektorů shodné dimenze. Přitom nezáleží na konkrétním způsobu extrakce vlastností z dokumentů, jedinou podmínkou je sestavení matice A tak, aby vektory vlastností dokumentů tvořily sloupce.

V našem případě nic nebrání sestavit matici A z jasových vektorů obrázků, jak bylo popsáno na začátku této kapitoly. Pomocí SVD matici A rozložíme stejně jako při klasickém použití LSI, tj. získáme aproximaci rozkladu $A \approx U_k \Sigma_k V_k^T$.



Obr. 4. Bázové obrázky (vizualizace konceptů, resp. singulárních vektorů) s příslušným pořadím singulárních čísel σ_i.

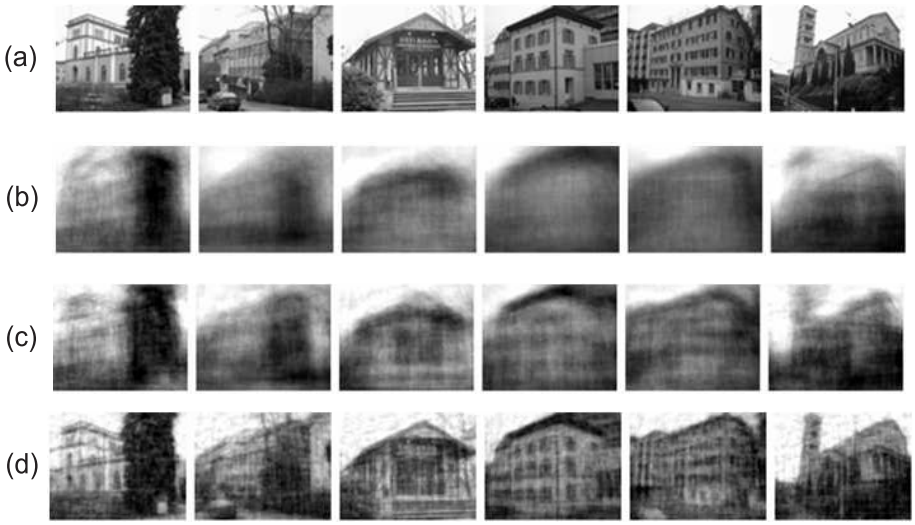
Zajímavá je interpretace a zejména vizualizace vektorů konceptů (báze U), které přestavují jakési "bázové obrázky", ze kterých je každý obrázek v kolekci složen (viz Obr. 4). Jinými slovy, vhodnou kombinací bázových obrázků dostaneme libovolný obrázek z kolekce (čím vyšší k, tím přesnější rekonstrukce obrázku).

Je zde jistá spojitost s diskrétní kosinovou transformací (DCT), využívanou při kompresi JPEG, kde je používáno např. 64 bázových obrázků pro obrazové segmenty velikosti 8×8, ovšem těchto 64 bázových obrázků je konstantních (jedná se o kombinace diskrétních kosinových funkcí s různou frekvencí) a dají se vygenerovat nezávisle na kolekci.

Na Obr. 4 vidíme několik nejdůležitějších bázových obrázků – ty můžeme interpretovat následovně. První bázový obrázek tvoří průměr jasu v celé kolekci (průměrný jas budovy), další báze reprezentují hrubé tvary (obrysy budov, rozhraní budov a pozadí) a báze s nejnižšími singulárními čísly σ_i (resp. s vyšším pořadím i) postupně popisují čím dál méně frekventované detaily v různých obrázcích (např. konkrétní tvary oken a dveří na budovách).

Rekonstrukce obrázků. Abychom získali vizuální představu o tom, jak velké (resp. malé) k je ještě dostačující pro popis obsahu obrázku, můžeme obrázky zpětně rekonstruovat tak, že pro dané k provedeme rozklad a matici A rekonstruujeme jako $A \approx U_k \Sigma_k V_k^T$.

Na Obr. 5a vidíme původní obrázky, resp. obrázky, které byly rekonstruované plným rozkladem $A = U\Sigma V^T$ (kde k je rovno hodnosti matice U^T). Naopak, pro velmi malá k je rekonstrukce velmi nedokonalá, viz Obr. 5b, kde $k = 15$. Přesto lze i z takto hrubě reprezentovaných obrázků rozeznat tvary původních budov. Uvědomme si, že rekonstruovaný obrázek je kombinací pouze 15 bázových ob-



Obr. 5. (a) Původní obrázky. Rekonstrukce obrázků z aproximace $A \approx U_k \Sigma_k V_k^T$ pro (b) $k = 15$, (c) $k = 50$, (d) $k = 250$.

rázků, tj. namísto 4800 hodnot jasu pixelů je potřeba pouze 15 "vah" konceptů! V případě $k = 50$ (viz Obr. 5c) je rekonstrukce dokonalejší, lze rozeznat přítomnost některých hrubých detailů, např. oken. Pro $k = 250$ je již rekonstrukce na takové úrovni, že původní budovy lze jednoznačně rozpoznat (viz Obr. 5d).

Z uvedeného příkladu lze usoudit, že pokud budeme chtít vyhledávat v kolekci obrázků podle hrubých obrysů, bude výhodnější použít pouze několik málo souřadnic vektorů pseudo-obrázků, naopak v případě, že budeme vyhledávat objekty také podle detailů, zvolíme větší počet souřadnic, tj. vyšší k.

Poznámka: Z příkladu lze rovněž usoudit, že celá metoda by mohla být využita pro kompresi celých kolekcí obrázků – výhodou oproti kompresi JPEG, resp. použití DCT, jsou bázové obrázky "šité na míru" dané kolekci (ne pouhé kombinace kosinů), což by se mělo projevit menším počtem bázových obrázků potřebných pro dostatečně kvalitní rekonstrukci. Nevýhodou je fakt, že spolu s vektory pseudo-obrázků potřebujeme uchovávat i bázové obrázky kolekce.

2.3 Funkce podobnosti pro porovnávání obrázků

Po provedení SVD rozkladu obdržíme kolekci vektorů pseudo-obrázků, které je potřeba nějak porovnávat s vektorem pseudo-dotazu (vzniklým transformací dotazového obrázku). Obecně se v oblasti vyhledávání v obrazových kolekcích používá mnoho nejrůznějších podobnostních měr, některé z nich nemají ani analytické vyjádření (např. ohodnocení natrénovanou neuronovou sítí).

V případě reprezentace obrázků histogramy barev/jasů se pro jejich porovnávání často používá kvadratická forma [8, 14], ve které je možné příbuznost

barev zachytit korelačními váhami mezi jednotlivými barvami (obecně souřadnicemi vektoru). Kvadratická forma je vlastně zobecněním Euklidovské vzdálenosti (kde neexistují korelace mezi souřadnicemi, všechny dimenze jsou považovány za nezávislé).

Jelikož v případě modelování obrázku vektorem pseudo-obrázku jsou souřadnice nekorelované (souřadnice obsahují váhy vektorů konceptů, které tvoří bázi a tudíž jsou lineárně nezávislé), postačí, když použijeme pro porovnání vektorů obyčejnou Euklidovskou vzdálenost, nebo jinou Minkowského metriku (L_p metriku). Podobnost modelovaná vzdáleností je interpretována tak, že čím vzdálenější jsou oba vektory, tím méně podobné jsou si obrázky. Nulovou vzdálenost mají identické obrázky (resp. jejich vektory). Výhoda Euklidovské vzdálenosti spočívá také v tom, že je to metrika, a proto lze celou kolekci obrázků (jejich vektorovou reprezentaci) indexovat pomocí metrických přístupových metod (*metric access methods* [4]), což posléze umožňuje v kolekci rychle vyhledávat. Rychlost vyhledávání je dána dvěma typy nákladů spořebovaných během vyhodnocování dotazu. Je to jednak počet diskových přístupů a jednak počet aplikací dané metriky (množství výpočtů vzdáleností).

3 M-strom

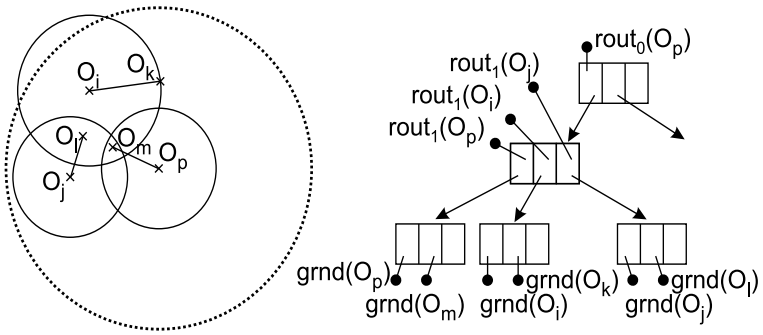
Jednou z metrických přístupových metod je M-strom [5, 9, 16], umožňující indexovat kolekci dokumentů modelovaných objekty metrického prostoru $\mathcal{M} = (\mathbb{U}, d)$, kde $\mathbb{U}$ je univerzum objektů (např. vektorů dokumentů) a d je metrika. Podobně jako mnoha struktur v jiných oblastech indexování, i struktura M-stromu je založena na myšlence B^+-stromu, tj. je vyvážená, dynamická a stránkovaná (umožňující efektivní perzistenci). Konkrétní index M-stromu představuje hierarchii metrických regionů (každý uzel představuje jeden region), respektive hierarchii shluků objektů v těchto regionech.

3.1 Struktura

Listy M-stromu obsahují záznamy $grnd(O_i)$ (ground entries) samotných indexovaných objektů O_i, zatímco vnitřní uzly obsahují tzv. *směrovací záznamy* $rout(O_j)$ (routing entries). Směrovací záznamy popisují tzv. *metrické regiony*, které vymezují v metrickém prostoru oblast, v níž se nacházejí objekty uložené v listech příslušného podstromu. Metrický region je popsán hyper-koulí se středem v nějakém objektu a příslušným pokrývajícím poloměrem (covering radius). Příklad hierarchie metrických regionů (pro Euklidovskou vzdálenost a 2D prostor) a příslušného M-stromu je uveden na Obr. 6.

3.2 Vyhodnocování dotazů na podobnost

Indexování objektů v M-stromu je realizováno pouze pomocí dané metriky d. Díky tomu je snadné implementovat dva základní typy dotazů na podobnost. Je to jednak *rozsahový* dotaz, který slouží k nalezení takových objektů, pro které



Obr. 6. Hierarchie regionů v metrickém prostoru a příslušný M-strom.

je vzdálenost od objektu dotazu menší než daná prahová hodnota a dále dotaz na h *nejbližších sousedů*[1] (h-NN dotaz), kterým získáme prvních h nejméně vzdálených objektů od objektu dotazu.

Vyšší efektivita (rychlost) vyhledávání v M-stromu (vůči např. prostému sekvenčnímu průchodu množiny objektů) spočívá v postupném odfiltrování těch větví M-stromu, které obsahují (vzhledem k dotazu) irelevantní objekty. Korektnost filtrování zaručuje zejména axiom trojúhelníkové nerovnosti metriky, což je klíčová vlastnost využívaná všemi metrickými přístupovými metodami.

3.3 Použití M-stromu pro indexování obrázků

Do M-stromu lze zaindexovat libovolnou kolekci reprezentovanou objekty v metrickém prostoru, můžeme jej tedy použít k indexování vektorů pseudo-obrázků podle vzdálenosti – zvolili jsme Euklidovskou vzdálenost. Podobnostní vyhledávání v kolekci obrázků provádíme pomocí rozsahových dotazů nebo dotazů na h nejbližších sousedů.

4 Experimenty

S výchozí kolekcí 730 obrázků budov jsme provedli dva druhy experimentů. Z hlediska přesnosti a úplnosti (precision and recall) jsme zkoumali úspěšnost samotné metody extrakce vlastností z obrázků pomocí LSI, z hlediska efektivity jsme pak testovali náklady na vyhodnocení dotazu pomocí M-stromu.

4.1 Kvalita vyhledávání

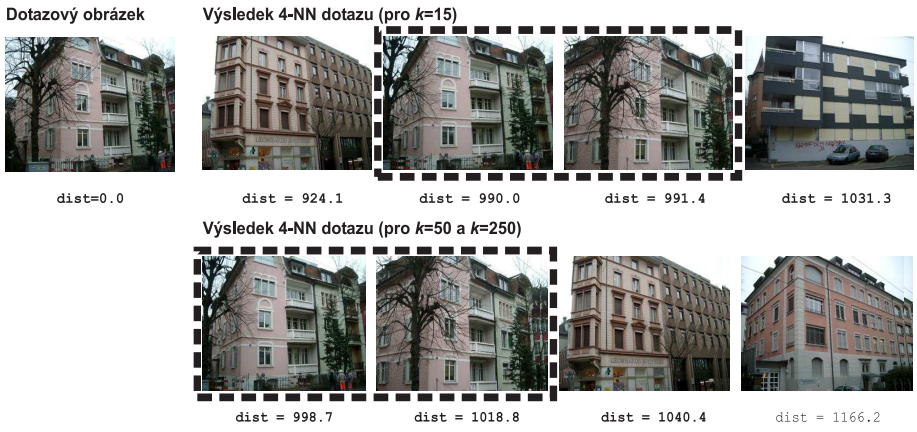
Vzhledem k tomu, že kolekce byla složena ze skupin po pěti obrázcích zobrazujících stejnou budovu z různých pohledů (viz Obr. 7), rozhodli jsme se pro úlohu

[1] Obvykle se uvádí parametr k, tj. dotaz na k nejbl. sousedů (k-NN dotaz). Parametr k jsme již zavedli u SVD, proto u dotazů budeme používat h, aby nedošlo k záměně.

identifikace budovy (jedna z nejtěžších úloh vyhledávání podle podobnosti). Zaindexovaná byla samozřejmě celá kolekce 730 obrázků, členění na skupiny uvnitř kolekce bylo pouze logické.

Obr. 7. Skupina obrázků téže budovy z různých pohledů.

Testování probíhalo tak, že jsme náhodně vybrali 50 skupin obrázků, přičemž z každé skupiny jsme dále náhodně vybrali jeden obrázek, položili 4-NN dotaz k tomuto obrázku a očekávávali, zda ve výsledku dotazu obdržíme zbývající 4 obrázky z dané skupiny. Pokud byly nalezeny všechny 4 obrázky, byla přesnost odpovědi 100%, pro 3 správně nalezené obrázky byla přesnost 75%, atd. Ačkoliv pořadí obrázků v odpovědi určuje míru podobnosti k dotazovému obrázku, pro stanovení přesnosti nám na pořadí nezáleželo vzhledem k malému počtu obrázků v odpovědi. Úplnost odpovědi v tomto případě odpovídala přesnosti, neboť velikost odpovědi se rovnala počtu relevantních obrázků v kolekci, tj. 4.

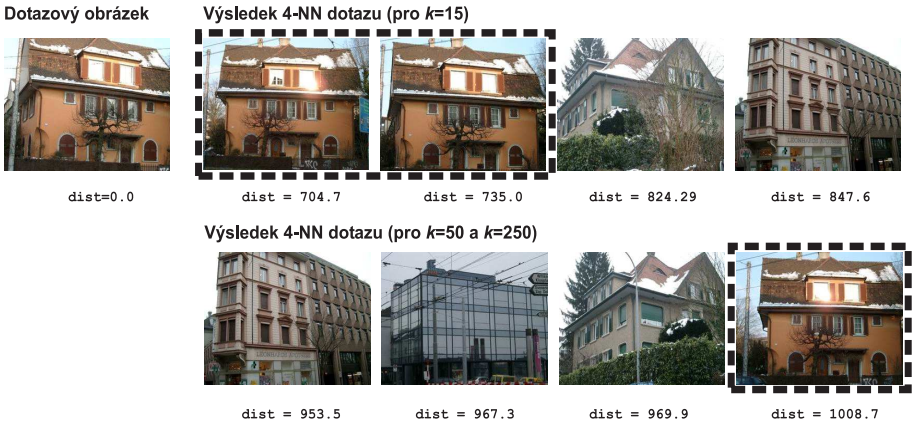


Obr. 8. Odpověd na první dotaz.

Na Obr. 8 vidíme příklad dotazu a odpovědí pro různá k (tj. pro popis obrázků pomocí k bázových obrázků). Přesnost odpovědí byla 50%, pro $k = 50$ a $k = 250$ navíc hledaná budova obsadila první dvě místa v odpovědi.

Na Obr. 9 vidíme 4-NN dotaz pro jiný obrázek. Přesnost odpovědí byla pro $k = 15$ opět 50%, pro $k = 50$ a $k = 250$ pouze 25%. Tento výsledek je zdánlivě v rozporu s faktem, že pro vyšší k získáme přesnější rekonstrukci obrázků, a tedy

by měla být i vyšší přesnost vyhledávání. Taková představa je ale mylná, protože, jak jsme podotkli v kap. 2.2, důležitější (frekventované) koncepty reprezentují hrubé tvary, kdežto méně důležité (tj. méně frekventované) koncepty popisují různé detaily. V uvedeném příkladu zřejmě došlo k situaci, kdy dotazování na příliš velké detaily dotazového obrázku použitím vyššího k způsobilo vyhledání irelevantních obrázků, které měly (ačkoliv to není přímo vizuálně patrné v samotných obrázcích) vyšší zastoupení detailů přítomných v obrázku dotazu. Naopak, relevantní obrázek (při vyhledání zařazený až na 4. místo) měl díky jinému pohledu na budovu poněkud odlišné detaily.



Obr. 9. Odpověď na druhý dotaz.

Zhodnocení. Pro 50 4-NN dotazů bylo dosaženo průměrné přesnosti 43% (tj. v každé odpovědi byly v průměru 1–2 správné obrázky), přičemž jako nejvýhodnější se ukázalo použití $k = 15$. Vzhledem k tomu, že jednotlivé obrázky v rámci skupin představovaly budovu z různých pohledů, často i velmi odlišných, lze považovat přesnost 43% jako velmi dobrou. Z toho lze také usoudit, že uvedená metoda LSI pro obrázky je poměrně odolná vůči prostorovým transformacím. Pokud bychom ještě zmírnili identifikační požadavek tak, že stačí, aby se ve 4 vyhledaných obrázcích vyskytoval alespoň jeden správný, dosáhli bychom identifikace téměř každé budovy.

4.2 Rychlost vyhledávání

V druhé skupině experimentů jsme testovali náklady na vyhledávání obrázků pomocí indexů M-stromu. Pro parametry $k = 15, k = 50, k = 250$, tj. pro každou sadu k-rozměrných vektorů, byl sestaven jeden index. Konstrukce probíhala metodou `MinMax + MultiWay`, detaily viz [16, 15]. Výsledky efektivity dotazování jsou uvedeny v Tabulce 1. Kapacita uzlů M-stromu byla stanovena na 10

objektů, přičemž průměrná naplněnost uzlů (v tabulce označeno jako UTIL) dosahovala cca 70%.

Při dotazování jsme sledovali počet diskových přístupů k uzlům (v tabulce označeno jako I/O) a počet aplikací metriky (v tabulce označeno COMP). Náklady jsou vyjádřeny procentuelně jako (a) podíl přístupů/aplikací vzhledem k velikosti indexu M-stromu, resp. počtu záznamů v indexu M-stromu (b) podíl přístupů/aplikací vzhledem k velikosti sekvenčního souboru, resp. počtu obrázků. Sledovány byly průměrné, maximální a minimální hodnoty nákladů (bylo provedeno 50 4-NN dotazů).

Tabulka 1. Efektivita vyhledávání v M-stromu.

Parametr	UTIL	4-NN dotazy											
k	%	I/O			COMP			I/O			COMP		
		% M-stromu			% M-stromu			% sekv.			% sekv.		
		Avg	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*	*Avg*	*Min*	*Max*
$k = 15$	68	**54**	17	76	**41**	11	58	**93**	29	131	**48**	13	68
$k = 50$	71	**63**	29	88	**49**	20	74	**104**	48	145	**58**	24	88
$k = 250$	70	**65**	24	91	**51**	16	75	**112**	41	157	**60**	19	88

Zhodnocení. Z tabulky můžeme usoudit, že nejlepších výsledků bylo dosaženo pro $k = 15$, kdy bylo potřeba načíst polovinu indexu M-stromu, tj. ekvivalent 93% sekvenčního souboru. Počet aplikací metriky byl nižší, v přepočtu 48% aplikací nutných pro sekvenční vyhodnocení. Uvedené výsledky nejsou zcela přesvědčivé z hlediska výhodnosti M-stromu oproti sekvenčnímu zpracování dotazu, nicméně je třeba podotknout, že experimenty byly prováděny na velmi malé kolekci (pouze 730 obrázků). V budoucnu bychom chtěli metodu vyzkoušet na kolekcích o deseti- až statisících obrázků, kde by se měl potenciál M-stromu projevit v podstatně větší míře.

5 Závěr

V tomto příspěvku jsme popsali specifika použití modelu LSI pro reprezentaci obrázků spolu s využitím M-stromu jako indexovací struktury pro efektivní vyhledávání. Díky aplikaci LSI na obrázky jsme také vizualizovali některé aspekty LSI, které nejsou při klasickém použití ve vektorovém modelu DIS přímo patrné.

Tento výzkum je částečně podporován grantem GAČR 201/05/P036.

Reference

1. R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., 1999.
2. M. W. Berry and M. Browne. *Understanding search engines: mathematical modeling and text retrieval.* Society for Industrial and Applied Mathematics, 1999.

3. C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.
4. E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Compututing Surveys*, 33(3):273–321, 2001.
5. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd Athens Intern. Conf. on VLDB*, pages 426–435. Morgan Kaufmann, 1997.
6. S. Deb. *Multimedia Systems and Content-Based Image Retrieval*. Information Science Publishing, 2003.
7. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
8. C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems (JIIS)*, 3(3/4):231–262, 1994.
9. M. Patella. *Similarity Search in Multimedia Databases*. PhD thesis, Dipartmento di Elettronica Informatica e Sistemistica, Bologna, `http://www-db.deis.unibo.it/Mtree/index.html`, 1999.
10. J. Pokorný, V. Snášel, and D. Húsek. *Dokumentografické informační systémy*. Karolinum, Praha, 1998.
11. P. Praks, J. Dvorský, and V. Snášel. Latent Semantic Indexing for Image Retrieval Systems. In *SIAM Conference on Applied Linear Algebra (LA03), The College of William and Mary, Williamsburg, USA*, 2003.
12. P. Praks, L. Machala, and V. Snášel. Iris Recognition Using the SVD-Free Latent Semantic Indexing. In *Workshop on Multimedia Data Mining (MDM/KDD 2004), Seattle, WA, USA*, 2004.
13. Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.
14. T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, pages 506–515, 1997.
15. T. Skopal. *Metric Indexing in Information Retrieval*. PhD thesis, Technical University of Ostrava, `http://urtax.ms.mff.cuni.cz/~skopal/phd/thesis.pdf`, 2004.
16. T. Skopal, J. Pokorný, M. Krátký, and V. Snášel. Revisiting M-tree Building Principles. In *Proceedings of the 7th East-European conference on Advances in Databases and Informations Systems (ADBIS), LNCS 2798, Springer-Verlag, Dresden, Germany*, pages 148–162, 2003.
17. H. Tamura, S. Mori, and T. Yamawaki. Textual features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(6), 1978.

Annotation:

An Application of LSI and M-tree in Image Retrieval

In this paper we present a method of LSI-based image feature extraction and, simultaneously, we demonstrate how such an extraction can be used for image retrieval using the M-tree. Moreover, we show several interesting properties, which are not directly visible when using the LSI on text collections.

Hodnocení kvality sumarizátorů textů

Josef Steinberger, Karel Ježek

Katedra informatiky a výpočetní techniky, FAV, ZČU - Západočeská Univerzita v Plzni, Univerzitní 22, 306 14 Plzeň

{jstein, jezek_ka}@kiv.zcu.cz

Abstrakt. Příspěvek se zabývá možnostmi hodnocení kvality výsledků sumarizátorů textů. Jsou zde popsány jednotlivé třídy metod hodnocení. Větší pozornost je věnována třídě metod založených na podobnosti obsahu. Uvádíme dvě nové metody hodnocení kvality extraktů, které využívají singulární dekompozici. Článek prezentuje výsledky testování tří sumarizátorů hodnocených z hlediska standardní kosinové metody a dvou nových metod založených na singulární dekompozici. Podobnost obsahu extraktu je měřena jak k plnému textu tak i k jeho abstraktu.

Klíčová slova: sumarizace, extrakt, abstrakt, singulární dekompozice

1 Úvod

Automatická sumarizace textů je vědecká disciplína, která v dnešní době poutá stále větší pozornost. Obrovské množství elektronických informací se musí redukovat, aby se s nimi uživatel mohl efektivněji vypořádat. Existuje mnoho sumarizačních metod. První skupinu tvoří extraktivní metody (*Summarization by Extraction*), jejichž cílem je vybrat nejvýznamnější věty v sumarizovaném textu a výsledkem je tedy extrakt. V dnešní době se začínají vyvíjet také algoritmy, které by umožňovaly automatickou tvorbu abstraktu generováním nových vět (*Summarization by Generation*). Tyto metody jsou však stále málo dokonalé. Pokud však chceme vylepšovat sumarizační metody, musíme nejprve vyvinout způsoby, jak posuzovat jejich kvalitu. Tento příspěvek se zabývá přístupy k hodnocení extraktů. Metody lze rozdělit do dvou skupin. První tvoří metody, které zjišťují, nakolik je automatický systém schopný zachytit hlavní myšlenky zdrojového dokumentu (*content evaluation*). Metody druhé skupiny měří textovou kvalitu výstupu sumarizátoru – hodnotí čtivost, gramatiku a logickou souvislost extraktu (*text quality evaluation*).

Dále je možné rozdělit metody na vnitřní (*intristic*) a vnější (*extrinsic*). První skupina metod vychází z obecných požadavků na abstrakty. Zkoumá tedy informativnost, pokrytí a správnost abstraktů (extraktů). Druhá zkoumá jejich kvalitu vzhledem k určitému cílovému použití.

V aktuálním výzkumu jsou dominantní následující řešení:

1. Srovnání extraktu s referenčním textem (vnitřní hodnocení) – viz kap. 2, 3 a 5
2. Subjektivní hodnocení soudců (vnitřní hodnocení)
3. Aplikačně založené hodnocení (vnější hodnocení) – viz kap. 4

Pro extrakty vět je často měřena podobnost výběru vět extraktu s referenčním extraktem (*co-selection*). Jako hlavní metriky se používají přesnost, úplnost a F-skóre [5]. Tyto metody sebou však přináší také některá omezení. Například lze je použít pouze pro extraktivní sumarizátory. Tuto nevýhodu odstraňují metody založené na podobnosti obsahu (*content-based*), které měří podobnost s referenčním extraktem na úrovni slov [5].

Aplikačně založené metody (*task-based*) měří kvalitu extraktů z hlediska jejich použití pro určitý úkol. Příkladem je kategorizace textů (viz kap. 4).

2 Metody založené na společném výběru částí textu

Tyto metody měří kvalitu z hlediska společného výskytu částí textu (většinou vět) v extraktech. Je zde nutné mít k dispozici referenční extrakt neboli extrakt, který se považuje za správný (*gold standard*). Tento extrakt se nejčastěji získává tak, že několik lidí (soudců) vybere věty, které by dle jejich mínění neměly v extraktu chybět. Věty, které jsou vybrány nejvíce soudci se zařadí do referenčního extraktu. V další části uvádíme tři možnosti měření kvality.

2.1 Přesnost a úplnost

Koeficienty přesnosti a úplnosti jsou definovány jako:

Koeficient přesnosti (Precision): $$P(E) = \frac{A}{A + C} \quad (1)$$

Koeficient úplnosti (Recall): $$R(E) = \frac{A}{A + B} \quad (2)$$

E = vyhodnocovaný extrakt
A = počet shodných vět ve vyhodnocovaném a referenčním extraktu
C = počet vět, které jsou ve vyhodnocovaném extraktu, ale nejsou v referenčním
B = počet vět, které jsou v referenčním extraktu, ale nejsou ve vyhodnocovaném

2.2 Kappa

Kappa je statistická míra, která má následující výhody:

- Vylučuje náhodnou shodu, která je definována jako úroveň kvality, která bude dosažena náhodným výběrem vět.
- Umožňuje určení pravděpodobnosti shody mezi soudci.

Koeficient Kappa (K) je definován následovně:

$$K = \frac{P(S) - P(N)}{1 - P(N)}, \quad (3)$$

kde *P(S)* je pravděpodobnost shody mezi systémem (resp. soudcem) a referenčním extraktem (resp. jiným soudcem) a *P(N)* je pravděpodobnost náhodné shody.

Tedy K=0, pokud je shoda sumarizačního systému stejná jako při náhodné shodě a K=1, pokud je hodnocený extrakt shodný s referenčním (perfektní shoda). Pokud je shoda horší, než by se očekávalo při náhodné shodě, kappa může být také záporné.

2.3 Relativní přínos

Jednotliví soudci nejprve ohodnotí věty (0-10). Ohodnocení automatických extraktů se potom zvyšuje přítomností vět s vysokým ohodnocením a snižuje přítomností redundantních vět [5].

3 Metody založené na podobnosti obsahu

Manuální extrakty (vytvořené soudci) však obecně nesdílejí mnoho stejných vět, a proto je problém získat referenční extrakt. Další nevýhodou výše popsaných metod je neschopnost zachytit sémantiku vět. Porovnejme například význam následujících dvou vět:

S1: „Návštěva prezidenta Spojených států amerických v Číně“

S2: „Americký prezident navštívil Čínu“

Do referenčního extraktu bude zařazena pouze jedna z těchto vět. Při hodnocení metodami založenými na společném výběru vět se bude přítomnost druhé věty v extraktu brát jako chyba.

S těmito nepříjemnými vlastnostmi se lze vypořádat použitím metod založených na podobnosti obsahu. Tyto metody počítají podobnost extraktů na nižší úrovni než jenom na úrovni celých vět.
Nabízí se zde tři možnosti:

a) počítat podobnost vyhodnocovaného extraktu s referenčním extraktem,
b) počítat průměr z podobností vyhodnocovaného extraktu s jednotlivými manuálními extrakty,
c) počítat podobnost vyhodnocovaného extraktu s plným dokumentem.

Následují tři běžná měření podobnosti.

3.1 Kosinová podobnost

Kosinová podobnost (*Cosine Similarity*) se spočítá podle následujícího vzorce:

$$\cos(X,Y) = \frac{\sum x_i * y_i}{\sqrt{\sum (x_i)^2} * \sqrt{\sum (y_i)^2}}, \quad (4)$$

kde X a Y jsou reprezentace textu ve vektorovém modelu. U kosinové podobnosti se pro reprezentace textu používají slova nebo lemmata.

3.2 Překrytí obsahu

Překrytí obsahu (*Unit Overlap*) lze spočítat následujícím způsobem:

$$overlap(X,Y) = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}, \quad (5)$$

kde X a Y jsou množinové reprezentace textu. $|S|$ je mohutnost množiny S. Zde se jako jednotky reprezentace textu používají množiny slov nebo lemmat.

3.3 Nejdelší subsekvence

Nejdelší subsekvenci (*Longest Common Subsequence*) můžeme spočítat následující formulí:

$$lcs(X,Y) = \frac{length(X) + length(Y) - edit_{di}(X,Y)}{2}, \quad (6)$$

kde *length(X)* je počet slov řetězce X a $edit_{di}(X,Y)$ je minimální počet operací vložení a vymazání slov potřebný k transformaci X na Y.

4 Aplikačně založené metody

Aplikačně založené metody zjišťují kvalitu skrze užití automatických extraktů pro daný praktický úkol. Testovat je možné například zvýšení rychlosti či přesnosti vyhledávání dokumentů, pokud je vyhledávání založené na extraktech místo na plných dokumentech (Korelace relevance). Dalším možným měřením je úspěšnost kategorizace dokumentů do tématických skupin, pokud se indexují extrakty místo původních dokumentů.

4.1 Korelace relevance

Korelace relevance (*RC - Relevance correlation*) je nová technika, která umožňuje měřit relativní pokles výkonu získávání informací, pokud se indexují extrakty místo plných dokumentů [5]. Předpokládejme, že máme dotaz Q a kolekci dokumentů D_i, vyhledávací systém seřadí dokumenty D_i podle jejich relevance k dotazu Q. Potom provedeme substituci plných dokumentů za extrakty S_i a stejný vyhledávací systém seřadí dokumenty S_i podle jejich relevance k dotazu Q. Pokud jsou extrakty dobrou náhradou původních dokumentů, předpokládá se, že pořadí v obou případech budou podobná. Existuje několik metod pro měření podobnosti pořadí (Kendall´s tau [8], Spearman´s rand correlation [8]). Protože však máme navíc k dispozici (z vyhledávacího systému) relevanci jednotlivých dokumentů k dotazu, můžeme spočítat *RC* následujícím způsobem:

$$RC = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}, \quad (7)$$

kde x_i je relevance dokumentu D_i k dotazu Q, y_i je relevance dokumentu S_i k dotazu Q. $\bar{x}$ (resp. $\bar{y}$) je průměrná relevance dokumentů D_i (resp. S_i) k dotazu Q.

4.2 Kategorizace dokumentů

Tato metoda zjišťuje vhodnost použití extraktů místo plných textů pro pozdější kategorizaci. Pro měření je potřebná zatříděná kolekce dokumentů. Při tomto způsobu testování se ke klasifikaci používá automatický klasifikátor. Z důvodu oddělení chyby klasifikátoru a chyby sumarizátoru je pak nutné použití některých základních hodnot pro porovnání. Výsledné hodnoty klasifikace extraktů jsou proto porovnávány např. s výsledky hodnocení původních dokumentů nebo hodnocení náhodně vybraných vět.

Posledním problémem zůstává míra určující kvalitu extraktu. Obecně se používají koeficienty přesnosti kategorizace P a úplnosti kategorizace R:

$$P = \frac{p}{q}, \quad (8)$$

$$R = \frac{p}{r}, \quad (9)$$

kde p je počet tříd, do kterých je dokument správně zatříděn klasifikátorem, q je celkový počet tříd, do kterých je dokument klasifikátorem zařazen a r je počet relevantních tříd, do kterých byl dokument klasifikovaný při předchozím ručním zatřiďování. Potom P a R pro celou kolekci je průměrem P a R přes všechny dokumenty. Z definice je možné vidět, že oba ukazatele spolu souvisí a zvyšováním jednoho se druhý bude snižovat. Při zařazení dokumentu do co nejvyššího počtu tříd bude vysoká úplnost, při snižování počtu tříd se bude zvedat přesnost. Z toho důvodu se pak používá pro hodnocení klasifikace např. průměr z obou hodnot [4].

5 Hodnocení singulární dekompozicí

Singulární dekompozice je proces, který je schopen najít v dokumentu hlavní témata (n-tice slov) a hodnoty jejich významnosti. Tuto vlastnost využívají naše dvě nová hodnocení, která zjišťují podobnost hlavního tématu (respektive n hlavních témat) automatického extraktu a referenčního dokumentu. Protože jsou tyto metody založeny na podobnosti obsahu, lze jako referenční dokument použít i manuální abstrakt, který může obsahovat i nově vytvořené věty.

5.1 Singulární dekompozice (Singular Value Decomposition)

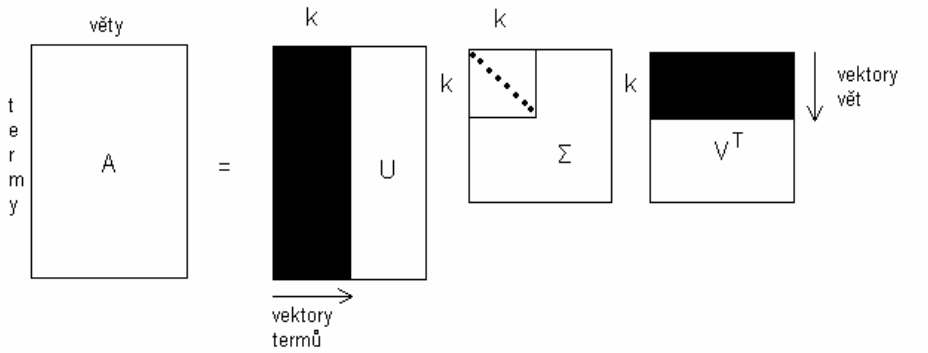
Singulární dekompozice (SVD) je numerický proces, který se hojně používá při redukci dat. V poslední době byly navrženy algoritmy, které singulární dekompozicí řeší klasifikaci nebo vyhledávání dokumentů (latentní sémantické indexování). SVD byla již použita i pro sumarizaci [3]. Na této myšlence jsou založeny i naše hodnotící algoritmy.

Proces začíná vytvořením matice termů proti větám $A = [A_1, A_2, \ldots, A_n]$, kde každý sloupcový vektor A_i reprezentuje vektor frekvencí termů ve větě i v dokomponovaném dokumentu. Pokud dokument obsahuje m termů a n vět, získá se matice A o rozměrech $m \times n$. Matice A je zpravidla řídká, protože normálně se každé slovo v každé větě nevyskytuje. Singulární dekompozice matice A je potom definována jako:

$$A = U\Sigma V^T, \tag{10}$$

kde $U = [u_{ij}]$ je $m \times n$ sloupcově ortonormální matice, jejíž sloupce se nazývají levé singulární vektory, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ je $n \times n$ diagonální matice, jejíž diagonální prvky jsou nezáporná singulární čísla seřazená sestupně a $V = [v_{ij}]$ je $n \times n$ ortonormální matice, jejíž sloupce se nazývají pravé singulární vektory (viz obr. 1). Pokud r je řád matice A, potom (viz [2]) Σ splňuje:

$$\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0 \,. \tag{11}$$



Obr. 1. Singulární dekompozice

Na interpretaci aplikování SVD na matici termů proti větám se můžeme dívat ze dvou pohledů. Z transformačního pohledu SVD zprostředkovává mapování mezi m-dimenzionálním prostorem vektorů frekvencí termů a r-dimensionálním singulárním vektorovým prostorem. Ze sémantického pohledu SVD poskytuje latentní sémantickou strukturu dokumentu reprezentovaného maticí A. Tato operace vyjadřuje rozklad originálního dokumentu do r lineárně nezávislých bázových vektorů. Každý term i věta dokumentu jsou indexovány těmito bázovými vektory. Unikátní vlastností singulárního rozkladu je schopnost zachytit a modelovat vnitřní vztahy mezi termy tak, že může sémanticky shlukovat termy a věty. Dále, jak je demonstrováno v [2],

pokud se v dokumentu často vyskytuje určitá kombinace slov, pak bude tato kombinace zachycena a reprezentována jedním ze singulárních vektorů. Velikost odpovídajícího singulárního čísla indikuje významnost kombinace v dokumentu. Každá věta obsahující tuto kombinaci slov bude promítnuta blízko odpovídajícího singulárního vektoru a věta, která nejlépe reprezentuje tuto kombinaci bude mít největší hodnotu v tomto vektoru. Každá kombinace slov popisuje určité téma dokumentu. Lze tedy na základě předchozích faktů říci, že každý singulární vektor reprezentuje určité téma dokumentu a velikost korespondujícího singulárního čísla reprezentuje významnost tohoto tématu [3].

Na základě předchozí diskuse jsme navrhli metodu hodnotící kvalitu extraktů. Tato metoda využívá singulární rozklad matice termů proti větám, konkrétně matici *U*, která popisuje míru významnosti termů v hlavních tématech (singulárních vektorech) dokumentu. Při hodnocení měříme podobnost mezi maticí *U* získanou singulárním rozkladem originálního dokumentu (nebo referenčního extraktu nebo abstraktu) a maticí *U* získanou singulárním rozkladem hodnoceného extraktu. Pro zjištění této podobnosti jsme navrhli následující dvě měření.

5.2 Podobnost hlavního tématu

Tento způsob měření porovnává první levé singulární vektory získané singulárním rozkladem referenčního dokumentu (plný text, referenční extrakt nebo abstrakt) a extraktu. Tyto vektory odpovídají hlavnímu tématu dokumentu. Čím více se tedy podobá hlavní téma extraktu hlavnímu tématu referenčního dokumentu, tím je extrakt kvalitnější. Podobnost měříme úhlem mezi vektory, které jsou normalizované, takže můžeme použít následující vzorec:

$$\cos\varphi = \sum_{i=1}^{n} ue_i \cdot ur_i \,, \tag{12}$$

kde *ur* je první levý singulární vektor rozkladu referenčního dokumentu, *ue* je první levý singulární vektor extraktu (hodnoty odpovídající termům jsou uspořádány podle referenčního dokumentu a na místě chybějících termů jsou nuly), *n* je počet různých termů referenčního dokumentu.

5.3 Podobnost *n* nejvýznamnějších témat

Tento způsob hodnotí podobnost z pohledu *n* hlavních témat porovnávaných dokumentů. Nejprve vytvoříme singulární rozklady referenčního dokumentu a hodnoceného extraktu. Potom pro oba dokumenty vynásobíme matice *U* a *Σ*. Získáme tím matice S_e (pro extrakt) a S_r (pro referenční dokument):

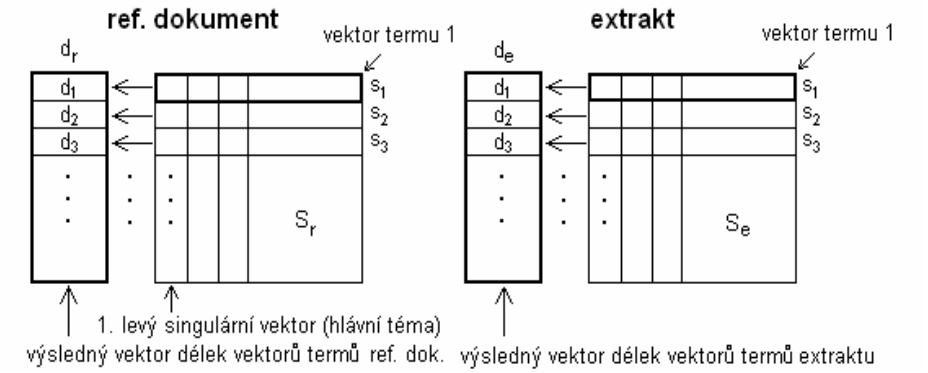
$$S_e = U_e \cdot \Sigma_e, \tag{13}$$

$$S_r = U_r \cdot \Sigma_r \,. \tag{14}$$

Důvodem tohoto násobení je zvýhodnění hodnot v matici U, které patří k nejvyšším singulárním číslům (nevýznamnějším tématům). Pro každý vektor termu (viz obr. 2) pak spočítáme jeho délku (pro extrakt i referenční dokument):

$$d_k = \sqrt{\sum_{i=1}^{n} s_{k,i}^2}\,, \tag{15}$$

kde d_k je délka vektoru k-tého termu, n je počet témat. V našich experimentech jsme zvolili konstantní počet tří nejvýznamnějších témat.



Obr. 2. Vytvoření výsledných vektorů ref. dokumentu a extraktu

Z délek vektorů termů sestavíme výsledný vektor délek termů v latentním prostoru vzniklém singulární dekompozicí (viz obr. 2). Získáme tím tedy dva výsledné vektory – jeden pro extrakt (*de*) a druhý pro referenční dokument (*dr*). Tyto vektory potom znormalizujeme. Pro změření jejich podobnosti použijeme opět kosinovou míru – analogicky s (**12**):

$$\cos\varphi = \sum_{i=1}^{n} de_i \cdot dr_i \cdot \tag{16}$$

Tato metoda má následující výhodu oproti předchozí. Pokud budeme uvažovat originální dokument jako referenční a ten bude obsahovat dvě přibližně stejně důležitá témata (odpovídající singulární čísla budou mít přibližně stejnou hodnotu), pak se může stát, že v extraktu převládne druhé nejvýznamnější téma nad prvním a hlavní téma bude vyhodnoceno jako velmi rozdílné. Tuto nevýhodu odstraníme, pokud hodnotíme podle více témat.

6 Testování

6.1 Testovací kolekce

K testování nových metod jsme použili kolekci Computation and Language Collection (cmp-lg) [7], která byla vytvořena korporací MITRE a univerzitou v Edinburghu. Tato kolekce formátu xml slouží jako zdroj pro testování v oblasti

dolování znalostí a sumarizace textů. Vyskytují se zde vědecké dokumenty dlouhé průměrně 169 vět a také jsou zde jejich abstrakty.

Tabulka 1. Detaily o testovací kolekci cmp-lg

Počet dokumentů	178
Minimální počet vět v dokumentu	45
Maximální počet vět v dokumentu	750
Průměrný počet vět v dokumentu	169
Průměrný počet slov v dokumentu	4504
Průměrný počet významových slov v dokumentu	1653
Průměrný počet různých významových slov v dokumentu	529

6.2 Výsledky

Úkolem testování bylo porovnání nových hodnocení, založených na singulární dekompozici, se standardní kosinovou podobností. Porovnávali jsme extrakty tří sumarizátorů. Prvním je náš sumarizátor, založený na myšlence sumarizování singulární dekompozicí [6]. Druhým je náhodný sumarizátor (RANDOM), který vybírá náhodné věty dokumentu. Ten slouží ke stanovení spodní hranice úspěšnosti. Třetí je sumarizátor se zabudovanou poziční heuristikou (LEAD) [5]. Ten preferuje věty na začátku dokumentu. Z výsledků lze tedy zkoumat, zda a nakolik se vyskytují důležité informace na začátku dokumentu. Nejprve jsme jako referenční dokument vzali plný text. Zkoumali jsme tedy podobnost plného textu a jeho extraktu. Potom jsme porovnávali abstrakt, který byl v kolekci připojen ke každému článku, a automatický extrakt. Sumarizační poměr byl stanoven na 3%, 5% a 10%. Jako sumarizační jednotku jsme nejprve brali větu, ale toto nastavení penalizovalo náhodné a poziční extrakty, které mohou vybrat i velmi krátké věty a celé extrakty tedy mohou být podstatně kratší. Proto bylo nakonec jako sumarizační jednotka nastaveno slovo a všechny extrakty byly přibližně stejně dlouhé. Důležitým faktorem kvality singulárního rozkladu je kvalita lematizačního slovníku. Při našem testování jsme použili lematizační slovník (SMART's English Stoplist), jehož autory jsou M.W. Berry a S. Dumais [1]. Hodnoty v následujících tabulkách udávají kosinus odchylky vektorů.

Tabulka 2. Hodnocení kvality 3% extraktů (ref. dokument je plný text)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	**0.651**	0.521	0.426
Podobnost hlavního tématu	**0.694**	0.390	0.371
Podobnost třech hlavních témat	**0.650**	0.420	0.378

Tabulka 3. Hodnocení kvality 5% extraktů (ref. dokument je plný text)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	**0.712**	0.603	0.504
Podobnost hlavního tématu	**0.767**	0.453	0.449
Podobnost třech hlavních témat	**0.653**	0.486	0.449

Tabulka 4. Hodnocení kvality 10% extraktů (ref. dokument je plný text)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	**0.788**	0.733	0.617
Podobnost hlavního tématu	**0.858**	0.608	0.565
Podobnost třech hlavních témat	**0.722**	0.600	0.556

V předchozích tabulkách je vidět, že u kosinové podobnosti není příliš velký rozdíl mezi propracovanou sumarizační technikou (SVD) a náhodným výběrem vět. U obou hodnocení singulární dekompozicí je rozdíl podstatně větší. Poziční sumarizátor byl vyhodnocen jako horší než náhodný.

Tabulka 5. Hodnocení kvality 3% extraktů (ref. dokument je abstrakt)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	0.543	0.240	**0.551**
Podobnost hlavního tématu	**0.325**	0.095	0.295
Podobnost třech hlavních témat	**0.308**	0.100	0.284

Tabulka 6. Hodnocení kvality 5% extraktů (ref. dokument je abstrakt)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	0.572	0.271	**0.604**
Podobnost hlavního tématu	**0.353**	0.110	0.341
Podobnost třech hlavních témat	0.319	0.119	**0.335**

Tabulka 7. Hodnocení kvality 10% extraktů (ref. dokument je abstrakt)

Hodnocení	Sumarizátor		
	SVD	RANDOM	LEAD
Kosinová podobnost	0.599	0.308	**0.669**
Podobnost hlavního tématu	0.387	0.149	**0.403**
Podobnost třech hlavních témat	0.353	0.147	**0.380**

Dále jsme zjišťovali podobnost extraktů s abstrakty. U 3% extraktů byl vyhodnocen SVD sumarizátor jako nejlepší, ale u delších extraktů (5% a 10%) jej již převýšil poziční sumarizátor. Toto zdánlivě překvapující zjištění potvrdilo, že na začátku článků se vyskytují významné informace. Je to vlastnost, které využívají např. heuristické sumarizátory (poziční heuristika – vyšší ohodnocení vět v úvodech článků).

7 Závěr

Tento příspěvek představil moderní trendy při hodnocení kvality sumarizátorů. Popsali jsme také novou hodnotící metodu založenou na singulární dekompozici. Testování prokázalo, že nová metoda je schopna rozlišit kvalitní extrakty od náhodných výběrů vět lépe než dnes používaná kosinová metoda. Dále jsme zaznamenali, že SVD je velice citlivá na kvalitu stoplistu a lematizační proces. V dalším výzkumu plánujeme zkvalitnění výstupu našeho SVD sumarizátoru. Budou zkoumány následující možnosti – rezoluce anafor, redukce a kombinace vět a vyloučení podobných vět. Cílem je vyvinout inteligentní sumarizační systém, který bude schopen prezentovat koherentní extrakty, které by správně vystihovaly původní dokument. Je zřejmé, že bez kvalitních hodnotících metod se neobejdeme.

Příspěvek vznikl za částečné podpory výzkumného záměru MSM 235200005 a ME494.

Reference

1. Berry M.W., Dumais S. http://www.cs.utk.edu/~lsi/.
2. Berry M.W., Dumais S.T., O'Brien G.W. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*. 1995.
3. Gong Y., Liu X. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. *Sborník 24. konference ACM SIGIR 2001*. New Orleans, USA 2001.
4. Hynek J., Ježek K. Practical Approach to Automatic Text Summarization. *Sborník 7. konference ELPUB '03*. Guimaraes, Portugalsko 2003.

5. Radev R., Teufel S., Saggion H., Lam W., Blitzer J., Qi H., Celebi A., Liu D., Drabek E. Evaluation Challenges in Large-scale Document Summarization. *Sborník 41. konference ACL 2003*. Sapporo, Japonsko 2003.
6. Steinberger J., Ježek K. Text Summarization and Singular Value Decomposition. *Sborník 3. konference ADVIS '04*. Izmir, Turecko 2004.
7. TIPSTER Text Summarization Evaluation Conference (SUMMAC). http://www-nlpir.nist.gov/related_projects/tipster_summac/cmp_lg.html
8. Siegel S., Castellan N. J. Jr. Nonparametric Statistics for the Behavioral Sciences. Berkeley, CA: McGraw-Hill, 2nd edn., 1988

Annotation:

Quality Evaluation of Text Summarizers

This paper describes possibilities of summary quality evaluation. Firstly, we mention the taxonomy of evaluation approaches. Further, we pay close attention to content-based methods. Then, two new evaluation methods that use singular value decomposition are proposed. Finally, we present evaluation results for three different summarizers from the angle of standard cosine content-based method and the two new evaluation methods based on the singular value decomposition. In the tests the content similarity between an original document and its summary and between the summary and its corresponding abstract was compared.

Web Service Composition for Deductive Web Mining: A Knowledge Modelling Approach

Vojtěch Svátek[1], Annette ten Teije[2], and Miroslav Vacura[1]

Department of Information and Knowledge Engineering, University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic

svatek@vse.cz, vacura@chello.cz

Department of Artificial Intelligence, Vrije Universiteit Amsterdam, De Boelelaan 1081A, 1081HV Amsterdam, The Netherlands

annette@cs.vu.nl

Abstract. Composition of simpler web services into custom applications is understood as promising technique for information requests in a heterogeneous and changing environment. This is also relevant for applications analysing the content and structure of the web. We discuss the ways the problem-solving-method approach studied in artificial intelligence can be adopted for template-based service composition for this problem domain; main focus is on the classification task.

1 Introduction

Composition of simple *web services* into sophisticated (distributed) applications recently became one of hottest topics in computer science research. Three alternative research streams can be identified:

1. *Programming in the large*, i.e. composition of services by (more-or-less) traditional procedural programming in languages such as BPEL4WS[3], inspired by workflow research. This stream is the only one recognised by most of the industrial IT community, to date; its main advantage is perfect control over the choice and linkage of different services, at design time. This however, on the other hand, entails a rather low degree of flexibility at run time.
2. *Planning* in artificial intelligence style, based on pre- and post-conditions of individual services without pre-specified control flows, as in OWL-S[4]. This approach offers extreme flexibility; however, the results may be quite unpredictable if all conditions are not perfectly specified, which may often be difficult in real environments.
3. *Template-based* composition, in which concrete services are filled in run time into pre-fabricated templates [9, 19].

The area of application for (composite) web-services is potentially quite wide. While the focus is most often on B2B transactions and financial services, the

[3] http://www-128.ibm.com/developerworks/library/ws-bpel

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 108–115, ISBN 80–248–0755–6. VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

general paradigm appears useful even for less critical tasks such as organisation of scientific events [19] or information harvesting from the *surface web*, which is the focus of the current paper.

The way information is presented on the web typically combines multiple types and representations of data. Free text is interleaved with images and structured lists or tables, pages are connected with hyperlinks, labelled with URLs (often containing meaningful tokens) and endowed with explicit meta-data (in specialised tags). Different methods of web data analysis (focusing each on a different data type/representation) may provide complementary and/or supplementary information. Reducing the analysis on a single method, which is typically done e.g. in text categorisation or information extraction projects, may thus lead to significant information loss. On the other hand, a monolithic application encompassing many methods would be impossible to maintain (in view of permanent changes in web data standards and conventions) as well as reuse in different domains. The only solution thus seems to be to combine multiple, relatively independent, tools, for which web services offer a collection of relatively mature and widespread interface standards (such as SOAP and WSDL). This is the approach taken in the *Rainbow* project [16], in which web-analysis services based on diverse principles (statistics, linguistic, graph theory etc.) have been designed and equipped with hand-crafted or inductively trained knowledge bases; adoption of third-party services is also envisaged.

As the number of available web-analysis tools increases, their composition by traditional programming becomes cumbersome. On the other hand, the space of suitable tools will hardly be as borderless as in semantic-web scenarios of information search, which are assumed amenable to planning approaches. The *template-based approach* thus looks as a reasonable compromise.

Most recently, ten Teije et al. [19] suggested to view web service composition templates as analogy to *problem solving methods* (PSMs), i.e. abstract descriptions of knowledge-based reasoning scenarios, which have been intensely studied in the knowledge modelling community for nearly two decades (see e.g. [5, 12]). In addition, they suggested to view the *configuration* of the template again as a kind of reasoning task, namely, that of *parametric design*. Each concrete application is assumed to be specified (in sufficient detail) merely as combination of values assigned to a fixed set of parameters. The configuration process is carried out by a so-called *broker* tool, and employs the *propose-critique-modify* (PCM) reasoning method[4], taking advantage of *background knowledge* of the broker. Independently, Svatek et al. [18] designed a collection of *PSMs* abstracted from real *deductive web mining* applications, with individual components (services) positioned in a *multi-dimensional space*. This space could play a similar role as the space of template parameters from [19], no reasoning method had been however formulated for automated configuration.

The goal of the current paper is thus to evaluate the possibility to adapt the parametric design approach from [19] to the (specific features of) web analysis PSMs from [18]. Main focus is on *classification*, which is the only task considered

[4] I.e. a 'meta-level' PSM with respect to that incorporated in the template itself.

in [19] and also one of tasks studied in [18]. The nature of the research, being joint venture of two related but independent projects, influences the structure of the paper. Section 2 explains the idea of web service composition based on parametric design [19], while section 3 describes the multi-dimensional framework and web-analysis PSMs from [18]. Section 4 then suggests a modification of the parametric design approach suitable for DWM, and shows examples of templates and background knowledge to be possibly used by a broker. Finally, section 5 surveys some related projects, and section 6 wraps up the paper and suggests directions for future research.

2 Configuration of Web Services as Parametric Design

2.1 Motivations

Current approaches to Web service configuration are often based on pre/post-condition-style reasoning. Given descriptions of elementary Web services, and the required functionality of the composite Web service, they aim to try to construct a 'plan' of how to compose the elementary services in order to obtain the required functionality. Planning techniques are heavily investigated for this purpose [13]. In [19], we instead proposed a *knowledge intensive* approach to the creation of composite Web services. We described a complex Web service as a fixed template, which must be configured for each specific use. Web service configuration can then be regarded as *parametric design*, in which the parameters of the fixed template have to be instantiated with appropriate component services. During the configuration process, we exploit detailed knowledge about the template and the components, to obtain the required composite web service. Whereas in other work the main metaphor is "Web service configuration = planning" (i.e. generalised reasoning based on only component specifications), our approach is based on the metaphor "Web service configuration = brokering" (i.e. reasoning with specialised knowledge in a narrow domain). A *planner* is assumed to be *domain-neutral*: it is supposed to work on any set of components, simply given their descriptions. A *broker* on the other hand exploits specific knowledge about the objects it is dealing with. In the remainder of this section, we describe how such a broker can be equipped with configuration knowledge on how to combine these web services.

2.2 Parametric Design

Parametric design is a simplification of general configuration. It assumes that the objects to be configured (in our case: complex Web services) have the same overall structure that can be captured by templates. Variations on the configuration can only be obtained by choosing the values of given parameters within these templates. We will show that for specific type of web services, namely classification services, this is indeed possible.

An existing reasoning method (PSM) for parametric design is *Propose-Critique-Modify*, or PCM for short [6]. The PCM method consists of four steps:

- The *propose step* generates an initial configuration. It proposes an instance of the general template used for representing the family of services.
- The *verify* step checks if the proposed configuration satisfies the required properties of the service. This checking can be done by both pre/post-condition reasoning, or by running the service.
- The *critique* step analyses the reasons for failure that occurred in the verification step: it indicates which parameters may have to be revised in order to repair these failures.
- The *modify* step determines alternative values for the parameters identified by the critique step. The method then loops backto the verify step.

The propose-critique-modify method for Parametric Design requires specific types of configuration knowledge to drive the different steps of the configuration process. The question is whether this configuration knowledge (PCM knowledge) can be identified for large classes of Web services. It turns out that this is indeed possible for a specific class of web services, namely, *classification* ones.

2.3 Application on Classification Services

The common definition of classification is [15]: "Classification problems begin with data and identify classes as solutions. Knowledge is used to match elements of the data space to corresponding elements of the solutions space, whose elements are known in advance." More formally, classification uses knowledge to map observations (in the form of ⟨*feature,value*⟩-pairs) to classes.

We address the question whether classification services can be described in a single template. [10] does indeed present such a general template:

1. First the observations have to be verified whether they are legal (*Check*).
2. All legal observations (⟨*feature,value*⟩-pairs) have to be scored on how they contribute to every possible solution in the solution space (*MicroMatch*).
3. Individual scores are then aggregated (*Aggregate*).
4. Candidate solutions are determined via aggregated scores (*Admissibility*).
5. Final solutions are selected among candidate solutions (*Selection*) .

This structure constitutes the overall template for classification services, which can be easily captured in current Web service description languages such as OWL-S [4]. Example values of *Admissibility* parameter are (see [19] for more):

- *weak-coverage*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution.
- *strong-coverage*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution and *explained* by them.
- *strong-explanative*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution, *explained* by them, and all features specified in the solution are *present*.

The value of *Selection* parameter then decides whether e.g. the number of unexplained and missing features is considered in ranking candidate solutions.

The broker may employ e.g. the following pieces of knowledge:

- Propose knowledge for the *Admissibility* parameter: if many ⟨*feature,value*⟩ pairs are irrelevant then do not use *strong-coverage*.
- Critique knowledge for the *Selection* parameter: if the solution set is too small or too large then adjust the *Admissibility* or the *Selection* parameter.
- Modify knowledge for the *Admissibility* parameter: if the solution set has to increased (reduced) in size, then the value for the *Admissibility* parameter has to be moved down (up) in the following partial ordering: *weak-coverage* $\prec$ *strong-coverage* $\prec$ *strong-explanative*.

A prototype PCM broker has been successfully applied on real data in the domain of conference paper classification (for reviewer assignment).

3 Framework and PSMs for Deductive Web Mining

3.1 The *TODD* Framework

In [18], *deductive web mining* was defined as 'all activities where pre-existing patterns are matched with web data'; the patterns may be either hand-crafted or learnt. We proposed a framework that positions any DWM tool or service within a space with four dimensions:

1. Abstract *task* accomplished by the tool:
   - *Classification* of a web object into one or more pre-defined classes.
   - *Retrieval* of one or more web objects.
   - *Extraction* of desired information content from (within) a web object.

   The *Classification* of an object takes as input its identifier and the list of classes under consideration. It returns one or more classes. The *Retrieval* of desired objects takes as input the (syntactic) *type* of object and *constraints* expressing its class membership as well as (part–of and adjacency) relations to other objects[5]. It outputs the *addresses* (based on URIs, XPath expressions and the like) of relevant objects. The *Extraction* task takes as input the class of information to be extracted and the scope (i.e., an object) within which the extraction should take place[6]. It outputs some (possibly structured, and most often textual) *content*. In contrast to Retrieval, it does not provide the information about location from where the content was extracted.
2. Type of *object* to be classified or retrieved[7]. The types, such as Document, Hyperlink, or Phrase, represent an upper-level of abstraction of web objects, and are defined by the Upper Web Ontology (see below). The basic assumption is that the type of object is always known, i.e. its assignment is not by itself subject of DWM.

[5] For example: "Retrieve (the XPath addresses of) those HTML tables from the given website that are immediately preceded with a possible 'Product Table Introduction Phrase' (containing e.g. the expression `product*`)".

[6] For example: "Extract the occurrences of Company Name within the Website".

[7] *Extraction* is not unambiguously associated with a particular object.

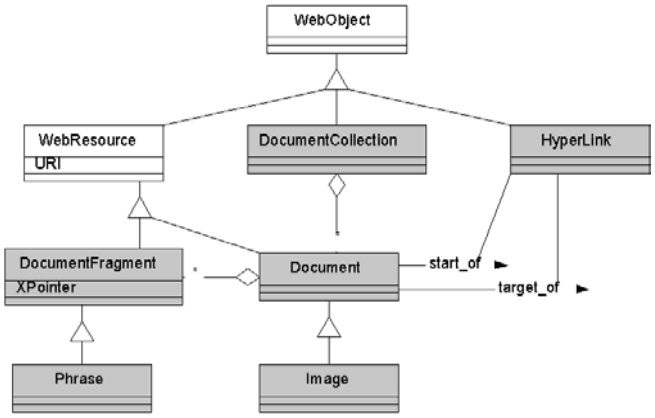


Fig. 1. UML diagram of Upper Web Ontology

3. *Data type and/or representation*, which can be e.g. full HTML code, plain text (without tags), HTML parse tree (with/without textual content), hyperlink topology (as directed graph), frequencies of various sub-objects or of their sequences (n-grams), image bitmaps or even URL addresses.
4. *Domain* in which the service is specialised.

We thus denote the framework as '*task-object-data(type)-domain*' (TODD). Its dimensions are to high degree independent, e.g. *object type* is only partially correlated with *data type*. For example, a document may be classified based on its HTML code, URL, META tag content or position in topology. Similarly, a hyperlink can be classified based on its target URL or the HTML code of source document (e.g. the menu structure containing the respective `<a>` tag).

The TODD framework by itself does not offer any added value to DWM application design until augmented with appropriate *ontologies*. Due to lack of space, we only show the *Upper Web Ontology* (see Fig. 1), which acts as general basis for more specific ontologies. Within the *Rainbow* project [16], domain ontologies (for bicycle product information and for web pornography) as well as ontologies related to analysis of specific types of web data have been designed.

3.2 Problem-Solving Methods

A characteristic feature of the web space is lack of clear object-feature-value structures, since, in a sufficiently comprehensive model, most features deserve to become objects of their own. As consequence, *structural* (say, recursive) PSMs arise. Eight PSMs have been formulated in [18]. Here we only concentrate on the (three) *Classification* and (two) *Retrieval* ones, and omit the (three) *Extraction* PSMs, for the sake of brevity.

Look-up based Classification amounts to picking the whole content of the given object, and comparing it with content constraints (such as look-up table),

which yields the class; for example, a phrase is a Company Name if listed in business register. *Compact Classification* also corresponds to a single inference, it is however not based on simple content constraints but on some sort of computation (e.g. Bayesian classification), which is out of the scope of the knowledge modelling apparatus. Finally, *Structural Classification* corresponds to classification of an object based on the classes of related objects (sub–objects, super–objects and/or neighbours). It is thus decomposed to *retrieval* of related objects, their *individual classification*, and, finally, evaluation of *global classification patterns* for the current object. It is therefore *recursive*: its 'inference structure' typically contains full-fledged (Direct) Retrieval and Classification tasks. Compared to the generic Classification template from section 2, this notion of classification is slightly simplified and more goal-driven. Some parts of Structural Classification PSM can be mapped on the generic template: classification from lower level of recursion is similar to MicroMatch, while evaluation of global pattern unites the Aggregage, Admissibility and Selection steps. There is no Check step (since no observations are known a priori), but an extra step of Retrieval (since objects relevant for classification of current object have first to be determined).

In *Direct Retrieval*, relevant objects are first retrieved based on parthood and adjacency constraints and then classified. Objects whose classes satisfy the class constraints are output. In *Index-based Retrieval*, (abstract) class constraints are first operationalised so that they can be directly matched with the content of objects. Then the objects are retrieved in an index structure (which is separate from the web space itself), possibly considering structural constraints (provided structural information is stored aside the core index). Corresponding CommonKADS [12] inference diagrams can be found in [18].

As an example of use of such PSMs, Table 1 shows the pseudo-code of pornography-recognition application consisting of tools developed within the PhD thesis [20]; descriptions of other applications can be found in [18]. The pseudo-code is assumed to clarify the structure of an application for a human user rather than to be run by machine. The 'predicate' corresponds to the first dimension in the TODD framework, i.e. *task name*; an extra letter is used to distinguish the PSMs. The first 'argument' is a variable referring to the *current object* of the task instance: input object in the case of Classification and output object/s in the case of Retrieval. The second to fourth 'arguments' reflect the three remaining TODD dimensions: *object type*, *data type* and *domain*. Finally, the fifth 'argument' contains additional specifications: list of classes distinguished in the classification task, and list of logical expressions determining the set of objects to be retrieved, respectively.

The upper level of the pornography-recognition process is an instantiation of the *Structural Classification* PSM. In order to classify the whole website (i.e. document collection), symptomatic 'out-tree' topology structures are first sought; their sources (local hubs) can possibly be identified with 'index' pages with image miniatures. To verify that, the hub is examined for presence of 'nudity' PICS rating in META tags (Look-up Classification PSM), for presence of indicative strings in the URL, and its whole HTML code is searched for 'image gallery'-like

Table 1. PSM-based pseudo-code representation of pornography application

```
ClaS(DC, DocCollection, _, Pornography, [PornoSite,@other]) :-
   RetD(D1, Document, topology, General, [D1 part-of DC, LocalHub(D1)]),
   ClaS(D1, Document, _, Pornography, [PornoIndex,@other]),
   RetD(D2, Document, topology, General, [D2 follows D1]),
   ClaS(D2, Document, _, Pornography, [PornoContentPage,@other]).
% classification of index page
ClaS(D, Document, _, Pornography, [PornoIndex,@other]) :-
   ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
   ClaS(D, Document, url, Pornography, [PornoResource,@other]),
   RetD(DF, DocFragment, html-txt, General, [DF part-of D, ImgGallery(DF)]),
   ClaC(DF, DocFragment, freq, General, [ScarceTextFragment,@other]).
% classification of content page
ClaS(D, Document, _, Pornography, [PornoContentPage,@other]) :-
   ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
   RetD(Im, Image, html-txt, General, [Im referenced-in D]),
   ClaC(Im, Image, image, Pornography, [PornoImage,@other]).
```

structures with low proportion of text (which distinguishes pornography from regular image galleries). The analysis further concentrates on individual pages referenced by the hub, and attempts to identify a single dominant image at each of them. The images are then analysed by (bitmap) image analysis methods; in particular, the proportion of body colour and the central position of a dominant object are assessed. In the description, we omit the 'evaluation of global classification pattern' subtasks, for brevity; their inclusion would be straightforward.

4 DWM Service Configuration as Parametric Design

4.1 Limitations of Fixed Template

The methods presented in the previous two sections share the interest in *configuring* a wide collection of *web services* into a functional application, and both rely on application templates in the form of *problem-solving methods*. The inventory applied on general classification PSMs (section 2) is more advanced, as it already comprises an operational meta-level, itself based on the problem-solving modelling paradigm. The *propose-critique-modify* method of *parametric design*, implemented by means of a configuration broker, enables to effectively search the space of the classification method family with respect to the task at hand. It thus seems obvious to apply a similar approach in the area of deductive web mining, which is of equally *analytic* nature and even comprises *classification* as one of underlying tasks. However, as we outlined in section 3, the PSMs for deductive web mining tend to involve *recursion*: a reasoning process starting at one object is successively redirected to other objects in its parthood or neighbourhood. This more-or-less disqualifies reasoning methods relying on a *single and entirely fixed*

feature template, of which parametric design is a typical representative. There seem to be at least two possible solutions to this problem:

1. to allow for *multiple templates per task*, differing in the number of 'sibling' sub-tasks and degree of recursion, and to include *heuristics for template selection* as part of broker knowledge.
2. to modify the *parametric design algoritm* to involve, in addition to setting parameter values, also *template-restructuring operations* such as subtask replication and recursive unfolding (i.e. replacement of parameter with a whole template for processing a different object).

In the rest of this section, we outline the first solution, since it is easier to design and implement in its rudimentary form; it obviously oversimplifies many aspects of real-world settings.

4.2 Multiple-Template Solution

Table 2 shows four versions of template for the classification task: the first one amounts to single way of (presumably, Look-Up or Compact) classification of the current object, the second aggregates two different ways of classifying the current object, the third relies on another object in order to classify the current object, and the fourth combines (presumably Look-Up or Compact) classification of current object with its structural classification (via classification of another object). The template must have its arguments filled for concrete use, such as in the pornography recognition application in section 3.

For simplicity, we again omit the evaluation of *global classification pattern*, which would be an additional component in the template. For example, the presence of sub-object of certain class determines the class of the super-object in a certain way; similarly, classification of the same object by different methods has to be compared and the result computed (e.g. by voting or weighing).

Let us formulate, analogously to section 2, some examples of broker knowledge. We will limit ourselves to *Propose* knowledge, which is relevant for initial setting of parameters. Note that, in our multiple-template version, broker knowledge relates to *template selection* as well as to *specification of arguments* for all subtasks within the template:

- Templates with lower number of distinct objects (`X`, `Y`, `Z`, ...) should be preferred.
- Non-recursive templates should be preferred; moreover, look-up classification should be preferred to compact classification.
- Default partial ordering of data types with respect to object classification, for *Document* object (may be overridden in a domain context):
  $frequency \succ URL \succ topology$, $free\_text \succ metadata$
- URL-based or topology-based classification (as rather unreliable kinds of services) should never be used alone, i.e. can only be filled into a template with 'parallel' classification of same object, such as SC2 or SC4

- Default partial ordering of types of relations (`@rel`) to be inserted into classification template (may be overridden in a domain context):
  part-of $\succ$ *is-part* $\succ$ *adjacent*
- Preference of domains used in structural classification,with respect to the domain of current object: *same domain* $\succ$ *super-domain* $\succ$ *other_domain*.
- The class of object determined by a Classification sub-task should be (according to domain knowledge) sub-class of the class of objects determined by the immediately preceding Retrieval sub-task in the template.

These heuristics are merely tentative, to illustrate the variety of possible broker knowledge for DWM applications. Let us further show a hypothetical scenario of their use, in connection with the pornography-recognition application from section 3. Imagine a web pornography ontology[8] containing among other the following description-logic axioms:

```
PornoSite same-class-as (WebSite and (has-part some PornoIndex))
PornoIndex same-class-of (LocalHub and (followed-by >1 PornoContentPage))
```

For an application recognising pornography websites, the broker would select the template SC3, which is simpler than SC4; neither SC1 nor SC2 would be applicable (assuming no service were able to recognise `PornoSite` by Look-Up or Compact Classification). In attempting to fill SC3 in, it would seek a class of related object that could help determine the class of current object. With the help of the first axiom, it finds out that `PornoIndex` could serve for the purpose (as part of sufficient condition); it will thus accordingly instantiate the Classification sub-task. Then it will determine, by the second axiom, a suitable class of objects to be retrieved in the preceding (Retrieval) sub-task as `LocalHub`; since this is not a pornography concept but generic concept, `Domain1` will be set to `General`. Finally, it finds out that `LocalHub` cannot be recognised as `PornoIndex` merely by Look-Up or Compact Classification. It will thus have to create another SC3 template, on the second level, in order to recognise `PornoIndex` by means of `PornoContentPage`s following it in the link topology.

5 Related Work

In the *IBrow* project [1], operational PSM libraries have been for developed for two areas of document search/analysis: Anjewierden [3] concentrated on *analysis of standalone documents* in terms of low-level formal and logical structure, and Abasolo et al. [2] dealt with information search in multiple external resources. Direct mining of websites was however not addressed; IBrow libraries thus do not cope with the problem of web heterogeneity and unboundedness, which motivated the development of the TODD framework. In contrast, the Armadillo system [7] attempts to integrate many website analysis methods; it currently relies on workflows manually composed from scratch by the user, although a template-based solution is also being envisaged.

[8] This ontology has actually been developed in DAML+OIL, by the authors [17].

Table 2. Sample templates for classification task

```
SC1: Cla(X, TypeX, _, Domain, GoalClass) :-
       Cla(X, TypeX, DataType1, Domain1, Class1),

SC2: Cla(X, TypeX, _, Domain, GoalClass) :-
       Cla(X, TypeX, DataType1, Domain1, Class1),
       Cla(X, TypeX, DataType2, Domain2, Class2).

SC3: Cla(X, TypeX, _, Domain, GoalClass) :-
       Ret(Y, TypeY, DataType1, Domain1, [Y @rel X  | OtherExps]),
       Cla(Y, TypeY, DataType2, Domain2, ClassY).

SC4: Cla(X, TypeX, _, Domain, GoalClass) :-
       Cla(X, TypeX, DataType1, Domain1, Class1),
       Ret(Y, TypeY, DataType2, Domain2, [Y @rel X  | OtherExps]),
       Cla(Y, TypeY, DataType3, Domain3, ClassY).
```

6 Conclusions and Future Work

Configuration of web services can be considered as *parametric design*, which enables to use the *propose-critique-modify* PSM. Thanks to *templates* we avoid configuring a webservice from scratch. Furthermore, such knowledge-intensive approach does not need complete functional descriptions of the components and of the required composite service but 'only' *configuration knowledge*. We attempted to apply this framework on service composition in the restricted domain of *deductive web mining*, in connection with a generic framework of DWM services. Due to the specific nature of web as underlying data structure, service templates tend to involve *recursion*, which also impacts the process of template-filling. Examples of specialised broker knowledge are shown, and a scenario of composing part of a pornography-recognition application is outlined.

Future research includes specification of templates for other DWM tasks, in particular those with nature of *extraction*, taking models of applications from [18] as starting point. The next step would be the development and evaluation of simple prototype of *problem-solving broker*. We will probably not able to test it on real data (as was done in [19] for parametric design over the original classification template), since processing real-world web data would require functionalities (cleaning, complex parsing) not worth implementing in a throw-away prototype. We will rather rely on artificial, simplified data, and only proceed to real data when switching to a *functional architecture* incorporating independently-developed (often third-party) tools, as envisaged in the *Rainbow* project [16]. We also expect to implement and test the solution based on *automatic template restructuring*. Finally, we would like to compare the efficiency of both template-based variants with pure pre/post-condition *planning* approach.

The research is partially supported by the grant no.201/03/1318 of the Czech Science Foundation. The authors would like to thank Frank van Harmelen and Martin Labský for comments on drafts of this paper.

References

1. IBROW homepage, `http://www.swi.psy.uva.nl/projects/ibrow`
2. Abasolo, C. et al.: Libraries for Information Agents. IBROW Deliverable D4, online at `http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html`.
3. Anjewierden, A.: A library of document analysis components, IBrow deliverable D2b. Online at `http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html`.
4. Ankolekar, A. et al.: DAML-S: Semantic markup for web services. In: Proc. ISWC 2002, *LNCS* 2342, pp. 348–363.
5. Benjamins, R., et al. (eds.): *IJCAI Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1999.
6. Brown, D., Chandrasekaran, B.: Design problem solving: knowledge structures and control strategies. *Research notes in AI*, 1989.
7. Ciravegna, F., Dingli, A., Guthrie, D., Wilks, Y.: Integrating Information to Bootstrap Information Extraction from Web Sites. In: IJCAI'03 Workshop on Intelligent Information Integration, 2003.
8. Kiefer, M.: Message to swsl-committee@daml.org, May 14, 2003.
9. Mandell, D. J., McIlraith, S. A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In: Proc. ISWC2003.
10. Motta, E., Lu, W.: A Library of Components for Classification Problem Solving. In: Proceedings of PKAW 2000, Sydney, Australia, December 2000.
11. Narayanan, S., Mcllraith, S.: Simulation, verification and automated composition of web services. In: Proc. WWW 2002.
12. Schreiber, G., et al.: Knowledge Engineering and Management. The CommonKADS Methodology. MIT Press, 1999.
13. Sheshagiri, M., desJardins, M., Finin, T.: A planner for composing service described in DAML-S. In: Workshop on Planning for Web Services, at ICAPS 2003.
14. Sirin, E., Hendler, J., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: Web Services: Modeling, Architecture and Infrastructure workshop at ICEIS2003.
15. Stefik, M.: *Introduction to knowledge systems*, Morgan Kaufmann, 1995.
16. Svátek, V., Kosek, J., Labský, M., Bráza, J., Kavalec, M., Vacura, M., Vávra, V., Snášel, V.: Rainbow - Multiway Semantic Analysis of Websites. In: 2nd International DEXA Workshop on Web Semantics (WebS03), IEEE Computer Society 2003.
17. Svátek, V., Kosek, J., Vacura, M.: Ontology Engineering for Multiway Acquisition of Web Metadata. LISP-2002-1 Technical Report, 2002. Available from `http://rainbow.vse.cz/papers.html`.
18. Svátek, V., Labský, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
19. ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of Web Services as Parametric Design. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
20. Vacura, M.: Recognition of pornographic WWW documents on the Internet (in Czech), PhD Thesis, University of Economics, Prague, 2003.

Updates of Nonmonotonic Knowledge Bases

Ján Šefránek

Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia

sefranek@fmph.uniba.sk

Abstract. Dynamic aspects of knowledge representation has been tackled recently by a variety of approaches in the logic programming style. We consider the approaches characterized by the causal rejection principle (if there is a conflict between rules, then more preferred rules override those less preferred). A classification and a comparison of the approaches is presented in the paper. We compare them also to our own approach based on Kripke structures.

Keywords: multidimensional logic programming, causal rejection principle, Kripke structure

1 Introduction

A theoretical investigation of dynamic aspects of knowledge representation is presented in this paper. Our attention is focused on evolving knowledge bases. Recently the problem of evolving knowledge bases has been tackled by a variety of approaches in the logic programming style, see [6, 3] and others. A detailed and comprehensive information is available in [5]. The presented model consists of a set of logic programs (each of them represents a module of the knowledge base) and of a preference relation on modules. The conflicts between the modules are resolved according to the preference relation: if there is a conflict between rules, then more preferred rules override those less preferred.

Goals (and contributions) of this paper are

- an introduction of a point of view useful for a classification of the approaches based on the causal rejection principle,
- a comparison of those approaches to our approach [8, 9],
- theorem 6 provides a new characterization of the strategy of rules rejection presented in [3].

The paper is structured as follows: Technical prerequisites are presented in the Section 2. Two basic types of rules rejection (those of [2] and [3]) are described in the Section 3. Then, in the Section 4 two strategies of accepting the default assumptions (when updating) are described. We combined them with the strategies of rule rejection and obtained four types of semantics of logic program updates. The relations between these four types are summarized. Finally, our approach, based on a Kripkean semantics is compared to the approaches based on the causal rejection principle.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 120–131, ISBN 80–248–0755–6. VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

2 Preliminaries

Consider a set of propositional symbols (atoms) $\mathcal{A}$. A *literal* is an atom (a positive literal) or an atom preceded by the default negation (a negative literal), *not* A. The set $\{notA : A \in \mathcal{A}\}$ will be denoted by $\mathcal{D}$ (defaults, assumptions). For each atom A, A and *not* A are called *conflicting literals*. A set of literals is called *consistent*, if it does not contain a pair of conflicting literals. A convention as follows is used: if literal L is of the form *not* A, where $A \in \mathcal{A}$, then *not* $L = A$.

A *rule* is a formula r of the form $L \leftarrow L_1, \ldots, L_k$, where $k \geq 0$, and L, L_i are literals (for each i). We will denote L also by $head(r)$ and the set of literals $\{L_1, \ldots, L_k\}$ by $body(r)$. If $body(r) = \emptyset$, then r is called a *fact*. The subset of all positive (negative) literals of $body(r)$ is denoted by $body^+(r)$ ($body^-(r)$). For each rule r we denote the rule $head(r) \leftarrow body^+(r)$ by r^+. We say that two rules, r and r', are *conflicting rules* if $head(r) = not\ head(r')$, notation: $r \bowtie r'$.

The set of all rules (over $\mathcal{A}$) forms the language $\mathcal{L}$. A finite subset of $\mathcal{L}$ is called a *generalized logic program* (program hereafter).[1]

A *partial interpretation* of the language $\mathcal{L}$ is a consistent set of literals. The set of all partial interpretations of the language $\mathcal{L}$ is denoted by $Int_{\mathcal{L}}$. A *total interpretation* is a partial interpretation I such that for each $A \in \mathcal{A}$ either $A \in I$ or *not* $A \in I$. Sometimes it will be convenient to speak about interpretations and sets of facts interchangeably. For this reason, we introduce the notation as follows: Let M be an interpretation, then $rule(M) = \{L \leftarrow : L \in M\}$.

We accept a convention as follows: All programs use only propositional symbols from $\mathcal{A}$. Interpretations of all programs are subsets of $Int_{\mathcal{L}}$.

A literal L is *satisfied* in a partial interpretation I if $L \in I$. A set of literals S is satisfied in a partial interpretation I if each literal $L \in S$ is satisfied in I. A rule r is satisfied in a partial interpretation I if $head(r)$ is satisfied in I whenever $body(r)$ is satisfied in I. Notation: $I \models L$, $I \models S$, $I \models r$.

A *total* interpretation I is a *model* of a program P if each rule $r \in P$ is satisfied in I.

Notice that (propositional generalized logic) programs may be treated as Horn theories: each literal *not* A may be considered as a new propositional symbol. The least model of a Horn theory H is denoted by $least(H)$.

Definition 1 (Stable model, [2]) Let P be a program and S be a total interpretation. Let $S^- = \{not\ A \in S : A \in \mathcal{A}\}$.

Then S is a *stable model* of P iff $S = least(P \cup rule(S^-))$. □

A program P is *consistent* iff there is a stable model of P, otherwise it is *inconsistent*.

Multidimensional dynamic logic program is defined as a set of generalized logic programs together with a preference relation on the programs [6]. A specification of the relation can be based on the edges of a graph.

[1] In this paper only the language of generalized logic programs is used. We incorporate some ideas of [3] into this framework.

Definition 2 ([6]) A *multidimensional dynamic logic program* (also *multiprogram* hereafter) is a pair $(\mathcal{P}, G)$, where $G = (V, E)$ is an acyclic digraph, $|V| \geq 2$, and $\mathcal{P} = \{P_v : v \in V\}$ is a set of generalized logic programs.

We denote by $v_i \prec v_j$ that there is a directed path from v_i to v_j and $v_i \preceq v_j$ means that $v_i \prec v_j$ or $i = j$. If $v_i \prec v_j$, we say that P_{v_j} is *more preferred* than P_{v_i}.

We denote the set of programs $\{P_{v_i} : v_i \preceq s\}$ by $\mathcal{P}_s$. □

If G is a directed path, the multidimensionality is collapsed and we speak simply about dynamic logic programs. The elementary case is represented by $V = \{u, v\}$ and $E = \{(u, v)\}$.

3 Strategies of the rules rejection

We account for two strategies how to formalize the idea of causal rejection. They lead to the sets $reject(\mathcal{P}, s, M)$ and $rjct(\mathcal{P}, s, M)$, see below. The former has been defined for example in [2, 6], the later for example in [3].

Definition 3 Let $(\mathcal{P}, G)$ be a multidimensional dynamic logic program, where $G = (V, E)$ and $\mathcal{P} = \{\mathcal{P}_v : v \in V\}$. Let M be an interpretation, $s \in V$.

$$
\begin{aligned}
reject(\mathcal{P}, s, M) = \{r \in P_i : \exists j \in V\ \exists r' \in P_j\ (i \prec j \preceq s\ \wedge\ r \bowtie r' \\
\wedge\ M \models \{body(r), body(r'))\}\}, \\
rjct^{\mathcal{P}}(s, M) = \emptyset \\
rjct^{\mathcal{P}}(i, M) = \{r \in P_i : \exists j \in V\ \exists r' \in P_j \setminus rjct^{\mathcal{P}}(j, M) \\
(i \prec j \preceq s \wedge r \bowtie r' \wedge M \models \{body(r), body(r')\}\} \\
rjct(\mathcal{P}, s, M) = \bigcup_{i \preceq s} rjct^{\mathcal{P}}(i, M)
\end{aligned}
$$

A multiprogram is denoted by $\mathcal{P}$ in what follows. A graph $G = (V, E)$ is implicitly assumed.

Theorem 4 *Let $\mathcal{P}$ be a multiprogram, $s \in V$, M be an interpretation. It holds that $rjct(\mathcal{P}, s, M) \subseteq reject(\mathcal{P}, s, M)$.* □

The inclusion $reject(\mathcal{P}, s, M) \subseteq rjct(\mathcal{P}, s, M)$ does not hold for some $\mathcal{P}$, s, M:

Example 5 ([7]) Let $\mathcal{P} = P, U, P_3$ be a dynamic logic program, $1 \prec 2 \prec 3$.

$$
\begin{aligned}
P &= \{a \leftarrow b, b \leftarrow\} \\
U &= \{not\ b \leftarrow a\} \\
P_3 &= \{b \leftarrow a\}
\end{aligned}
$$

Let w be $\{a, b\}$. Then

$$reject(\mathcal{P}, 3, w) = \{b \leftarrow, not\ b \leftarrow a\}$$
$$rjct(\mathcal{P}, 3, w) = \{not\ b \leftarrow a\}$$

□

The rejection done by *rjct* is a minimal one, in a reasonable sense:

Theorem 6 *Let a multi-program $\mathcal{P}$ and $s \in V$ be given. Then for each w holds that $rjct(\mathcal{P}, s, w)$ is a minimal subset of $\mathcal{P}_s$ containing at least one rule from each pair $(r, r') \in \mathcal{P} \times \mathcal{P}$ of conflicting rules, where $w \models \{body(r), body(r')\}$ and the set of all non-rejected rules is a consistent one, if each $P \in \mathcal{P}$ is consistent.* □

Remark 7 Recently, a new strategy of rules rejection has been introduced in [1]:

$reject^*(\mathcal{P}, s, M) = \{r \in P_i : \exists j \in V\ \exists r' \in P_j\ (i \preceq j \preceq s \wedge r \bowtie r' \wedge M \models \{body(r), boc$

It means, also conflicts between rules of the same program are solved. Operator $rjct$ can be modified in the same style. This modification solves problems with cyclic updates, but more essential problems with causal rejection principle (discussed in [9, 10]) are not touched by this modification. Moreover, the refined extension principle behaves well only for dynamic logic programs, see [11]. Therefore, we do not devote an attention to corresponding strategy of rules rejection.

4 Strategies of assumptions accepting

The meaning of a program depends both on rules and on default assumptions (see Definition 1). Therefore, the approach to default assumptions accepting is an essential one for updates of logic programs. Again, two strategies are discussed. Observe, that two strategies of rules rejection may be combined with two strategies of the acceptance of default assumptions.

The first strategy is as follows: Consider a multi-program $\mathcal{P}$. The updated program consists of all rules of $\mathcal{P}$ except those rejected (according to the selected strategy). If S is a stable model of an updated program, then S^- is the corresponding set of default assumptions. The concepts of justified updates [5] and backward-justified updates use this strategy.

Definition 8 Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a multiprogram, $s \in V$.

An interpretation M is a *justified update* of $\mathcal{P}$ at state $s \in V$ iff M is a stable model of the program $\mathcal{P}_s \setminus reject(\mathcal{P}, s, M)$. An interpretation M is a *backward-justified update* of $\mathcal{P}$ at state $s \in V$ iff M is a stable model of the program $\mathcal{P}_s \setminus rjct(\mathcal{P}, s, M)$. □

Theorem 9 ([5]) *Let $\mathcal{P}$ be a program and $s \in V$. If S is a justified update of $\mathcal{P}$ at state s, then S is its backward-justified update at s.* □

The Theorem 9 does not hold in the converse direction:

Example 10 Consider the multiprogram $\mathcal{P}$ from the Example 5. The interpretation $\{a, b\}$ is a backward-justified update of $\mathcal{P}$ at state 3, but it is not its justified update at 3. □

(Backward-)justified updates suffer from some unpleasant properties:

Example 11 ([5]) Let $\mathcal{P} = \langle P, U \rangle$, where $1 \prec 2$,

$$P = \{a \leftarrow\}$$
$$U = \{not\ a \leftarrow not\ a\}$$

Both $M_1 = \{a\}$ and $M_2 = \{not\ a\}$ are justified (and backward-justified) updates of $\mathcal{P}$ at state 2.

There is no reason to accept the interpretation M_2 (more precisely, to accept the default assumption *not* a in M_2) and, consequently, to reject the fact $a \leftarrow$. Troubles are caused also by (more general) cyclic updates. □

The (implicit) policy of accepting default assumptions which is behind the (backward-)justified updates is not an adequate one. Certainly, a more subtle policy is required. Such a policy is proposed within the next strategy:

Definition 12 (Dynamic stable model at state s, [6]) Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a multidimensional dynamic logic program, where $D = (V, E)$ and $\mathcal{P}_D = \{\mathcal{P}_v : v \in V\}$. Let M be an interpretation, A be an atom, $s \in V$. Then

$$default(\mathcal{P}, s, M) = \{not\ A : \neg\exists r \in \mathcal{P}_s : (head(r) = A\ \wedge\ M \models body(r))\}$$

An interpretation M is a *dynamic stable model* of $\mathcal{P}$ at state $s \in V$, iff

$$M = least((\mathcal{P}_s \setminus reject(\mathcal{P}, s, M)) \cup default(\mathcal{P}, s, M))$$

and M is a *backward-dynamic stable model* of $\mathcal{P}$ at state $s \in V$, iff

$$M = least((\mathcal{P}_s \setminus rjct(\mathcal{P}, s, M)) \cup default(\mathcal{P}, s, M)).$$

Remark 13 If the condition $M \models \{body(r), body(r')\}$ (from the definition of *reject* or *rjct*) is simplified to $M \models body(r')$ we get an equivalent notion of dynamic stable model: rules whose body is not satisfied in M do not affect the least model. □

Theorem 14 ([3, 5]) *If S is a dynamic stable model of $\mathcal{P}$ at state s, then it is its justified update at s.* □

The converse implication doesn't hold:

Example 15 ([5]) Consider the program from the Example 11: $M_2 = \{not\ a\}$ is not a dynamic stable model of $\mathcal{P}$: $default(\mathcal{P}, 2, M_2)) = \emptyset$, but $M_2^- = M_2$. □

Theorem 16 *If S is a backward-dynamic stable model of $\mathcal{P}$ at state s, then it is its backward-justified update.* □

Also the Theorem 16 does not hold in the converse direction, see the Example 11. (Notice that for the sequences of two programs *reject* coincide with *rjct*, therefore their backward-justified updates coincide with justified updates, and their backward-dynamic stable models coincide with dynamic stable models.)

Theorem 17 *Let $\mathcal{P}$ be a multiprogram, $s \in V$. If S is a dynamic stable model of $\mathcal{P}$ at state s then it is its backward-dynamic stable model at s.* □

The Theorem 17 does not hold in the converse direction, see again the examples 5 and 10: $w = \{a, b\}$ is not a dynamic stable model of $\mathcal{P}$ at state 3, but it is its backward-dynamic stable model.

The summary: For each given multi-program $\mathcal{P}$ holds: dynamic stable models of $\mathcal{P}$ (at each state s) create a proper subset of justified updates of $\mathcal{P}$ and of backward-dynamic stable models of $\mathcal{P}$. And both last mentioned interpretations create a proper subset of backward-justified updates.

Remark 18 A detailed comparison of semantics based on causal rejection of rules is presented in [4]. The semantics studied in [4] coincide on a restricted class of programs (sufficiently acyclic programs and acyclic programs).

5 Kripkean semantics

The problems with tautological and cyclic updates are not removed completely by the introduction of dynamic stable model semantics:

Example 19 ([5]) Let $\mathcal{P}$ be $\langle P, U\rangle$, where $1 \prec 2$ and

$$P = \{not\ a \leftarrow \qquad a \leftarrow\} \qquad U = \{a \leftarrow a\}$$

$S = \{a\}$ is the (backward-)dynamic stable model of $\mathcal{P}$ at state 2.

Similarly, if $\mathcal{P}'$ is $\langle P, U'\rangle$, where $U' = \{a \leftarrow b; b \leftarrow a\}$, then $S = \{a, b\}$ is the only dynamic stable model of $\mathcal{P}'$ at state 2. □

A recent semantics, called refined dynamic stable model semantics [1], solves the problem of tautological (cyclic) updates which can resolve inconsistencies in a program by a simple straightforward method – conflicting rules in the program are rejected mutually. However, the solution holds only for sequences of programs. The problem is not resolved for the multidimensional case, see [11].

Moreover, the semantics of (refined) dynamic stable models suffers form other fundamental problems. It enables irrelevant updates[2], see Example 20, and – on the other hand – it is not able to recognize such conflicts between programs that are not manifested by conflicts between rules, see Example 21.

Example 20 ([3]) Let $\mathcal{P}$ be $\langle P, U\rangle$, with $1 \prec 2$, where

$$P = \{a \leftarrow b \qquad b \leftarrow\} \qquad U = \{not\ b \leftarrow not\ a\}$$

Dynamic stable models of $\mathcal{P}$ at state 2 are $S_1 = \{a, b\}$ and $S_2 = \{not\ a, not\ b\}$. If the information from P and U is given, there is no sufficient reason to believe in S_2 and to reject the fact $b \leftarrow$.

$P \cup U$ is a consistent program, its only stable model is S_1. □

Suppose that we accept a "static" semantics *Sem* (for example the stable model semantics) and a "dynamic" semantics *DynSem* (for example dynamic stable model semantics). It is very natural to accept the postulate as follows: If $P \cup U$ is consistent then $Sem(P \cup U) = DynSem(\langle P, U\rangle)$. Observe that this very natural requirement is not fulfilled by dynamic stable model semantics, see Example 20 (and also by the other semantics based on the causal rejection principle).

Moreover and most importantly, there are some conflicts between the programs that are principally not solvable on the level of the conflicts between rules:

Example 21 Let U be a more preferred program than P.

$$P = \{a \leftarrow b, c\} \qquad U = \{b \leftarrow not\ a \qquad c \leftarrow b\}$$

There is no conflict between the rules of both programs. No rule can be rejected and the meaning of $P \cup U$ cannot be updated according to the dynamic logic programming [2] paradigm and according to a semantics based on rejection of rules.. However, there is a sort of conflict between the programs (between their meanings): union of two consistent programs is inconsistent. It is not natural to solve only conflicts (inconsistencies) caused by conflicts of rules and ignore the other sources of inconsistency.

In our example, the literal a in the less preferred program P depends on a set of assumptions (on the set of literals $w = \{b, c\}$). On the other hand, every literal in w is dependent on a default assumption (on the literal *not* a) in the more preferred program U. Notice that there is a circular dependency of a on *not* a in $P \cup U$. The problem of circular dependency can be resolved by rejecting the less preferred dependency of a on $\{b, c\}$.

[2] If union of the set of all programs in a multidimensional logic program $\mathcal{P}$ is consistent, but the set of all dynamic stable models of $\mathcal{P}$ and the set of stable models of the union of all programs are different sets, we say that the update is irrelevant. Similarly for other semantics based on rejection of rules.

Hence, our goal is to define rejection of dependencies. We need a more rich semantic structure, in order to be able to do it. □

Our approach provides a semantic treatment of dependencies between sets of literals (belief sets). The dependencies are encoded in (rather nonstandard) Kripke structures. The intuition is as follows: if the world (or our knowledge of the world) is represented by an interpretation w then (the meaning of) a program P may be viewed as a set of transitions to other worlds, compatible, in a sense, with w. The transitions are specified as follows: if $body(r)$ is satisfied in w for some $r \in P$ then the world $w \cup \{head(r)\}$ is compatible with w, if it is consistent. It is a natural choice to require that the compatibility relation is transitive and irreflexive.

We present a simplified (but a sufficient one for our current needs) version of the definition of a Kripke structure associated with a program. For the more complicated version see [9].

Definition 22 (Kripke structure associated with a program) Let P be a program. A Kripke structure $\mathcal{K}^P$ associated with P is a a pair (W, ρ), where:

- $W = Int_{\mathcal{L}} \cup \{w_\perp\}$, W is called the set of possible worlds, $w_\perp$ is the representative of the set of all inconsistent sets of literals,
- ρ is a binary relation on $W \times W$, it is called the accessibility relation and it contains the set of all pairs (w, w'), where $w \neq w'$, satisfying exactly one of the conditions:
  1. $w' = w \cup \{head(r)\}$ for some $r \in P$ such that $w \models body(r)$,
  2. $w' = w_\perp$ iff $\exists r \in P\ (w \models body(r) \wedge not\ head(r) \in w)$.

Definition 23 If $e = (u, v) \in \rho$, it is said that e is a *ρ-edge* and u (v) is called the *source* (the *target*) of e. A *ρ-path* is a sequence σ of ρ-edges $(w_0, w_1), (w_1, w_2), \ldots, (w_{n-}$ in a Kripke structure $\mathcal{K}$.

We say that this σ is *rooted* in w_0 (also w_0-rooted). If there is no ρ-edge (w_n, w) in $\mathcal{K}$, we say that σ is *terminated* in w_n, w_n is called a *terminal* of σ. □

Paths are usually denoted by a shorthand of the form $\langle w_0, w_1, \ldots, w_n \rangle$. We are now ready to state (in terms of nodes and paths in $\mathcal{K}^P$) conditions of being a stable model of a program P.

Definition 24 (Distinguished paths, good worlds) Let P be a program, σ be a ρ-path $\langle w_0, \ldots, w_n \rangle$ in $\mathcal{K}^P$. We say that σ is *correctly rooted*, if $w_0 \subseteq \mathcal{D}$.

A correctly rooted ρ-path σ terminated in a total interpretation w is called a *distinguished* path and w is called a *good world*. □

Theorem 25 *Let P be a program, $\mathcal{K}^P$ be the Kripke structure associated with P.*

Then w_n is a good world in $\mathcal{K}^P$ iff it is a stable model of P. □

Remark 26 If $\mathcal{D}$ is a terminal in $\mathcal{K}^P$, it is the (only) stable model of P. The trivial sequence $\langle \mathcal{D} \rangle$ is correctly rooted and terminated in $\mathcal{D}$. Suppose that a total interpretation $w \neq \mathcal{D}$ is a stable model of P, we get $(\mathcal{D}, w_\perp) \in \rho$. □

6 Updates of Kripke structures

While the semantics presented in Sections 3 and 4 are based on rules rejection, our semantics of updates is based on a rejection of some edges in Kripke structures. In other words, the updates change the compatibility relation between possible worlds.

A removal of some edges may be interpreted as overriding the corresponding dependencies between belief sets. On the other hand, connecting the edges from one Kripke structure to the edges from another may be interpreted as an enrichment of the dependencies between belief sets.

Suppose two programs,[3] P and U, and the Kripke structures, $\mathcal{K}^P = (W, \rho^P)$ and $\mathcal{K}^U = (W, \rho^U)$, associated with P and U, respectively. Moreover, U (the updating program) is more preferred than P (the original program).

Our approach enables to recognize conflicts between programs even if there are no conflicts between rules. In such cases some edges are rejected, but there is no reason to reject a rule. We regard this as an essential observation:

Example 27 Consider the Example 21. We "translate" its idea into Kripke structures. The world $\{a, b, c\}$ (and, consequently, the literal a) is dependent on the world $w = \{b, c\}$ in the less preferred Kripke structure $\mathcal{K}^P$. Similarly, the world $w_\perp$ is dependent on the world $w' = \{not\ a, b, c\}$ in the less preferred structure and w' is dependent on the world $\{not\ a\}$ in the more preferred Kripke structure $\mathcal{K}^U$. We do not want to accept the circular dependency of a on $not\ a$, therefore we propose to reject the (less preferred) edge $(\{not\ a, b, c\}, w_\perp) \in \rho^P$ (leading to inconsistency). □

We intend to define an operation $\oplus$ on Kripke structures. The resulting Kripke structure $\mathcal{K}^{U\oplus P} = \mathcal{K}^U \oplus \mathcal{K}^P = (W, \rho^{U\oplus P})$ should be based on $\mathcal{K}^U$ while a reasonable part of $\mathcal{K}^P$ is preserved. Notice that the set of nodes, W, remains unchanged, but some ρ^P-edges should be rejected.

Definition 28 (Attacked edges) Let $W = Int_{\mathcal{L}}$. Let $\tau_1, \tau_2 \subseteq W \times W$ be binary relations. Let L be a literal.

We say that $e = (u, u \cup \{L\}) \in \tau_1$ is *attacked* by $e' = (u, u \cup \{not\ L\}) \in \tau_2$. □

Of course, there is a symmetry: if e is attacked by e' then e' is attacked by e, too. Nevertheless, we want to prefer "one side". We intend to reject an edge from ρ^P if it is attacked by an edge in ρ^U.

Sometimes it is needed to reject a ρ^P-edge of the form $(w, w_\perp)$, see the Example 21. In this case the analysis is a little bit more complicated.[4]

The basic rule is as follow: if $(w, w_\perp) \in \rho^P$ and w occurs on a ρ_2-path, then $(w, w_\perp)$ should be rejected, if the rejection is not blocked. The idea of blocking is illustrated in the next Example.

[3] In this paper only the elementary case of two programs is considered (because of the limited size). The more general theory is presented in [9].

[4] More motivations and examples are presented in [9].

Example 29 (Blocking of rejections) Let $\mathcal{P}$ be $\langle P, U\rangle$.

$$P = \{a \leftarrow \qquad U = \{not\ a \leftarrow not\ b$$
$$b \leftarrow \qquad\quad not\ b \leftarrow not\ c$$
$$c \leftarrow\} \qquad\quad not\ c \leftarrow not\ a\}$$

Let $w_1 = \{not\ a, not\ b, not\ c\}$. There are three ρ^U-paths to w_1. One of them is $\sigma = \langle\{not\ a\}, \{not\ a, not\ b\}, w_1\rangle$. Notice that $e_1 = (\{w_1, w_\perp) \in \rho^P$.

Observe that the ρ^U-paths mentioned above are rooted in default assumptions (in$\{not\ a\}$ or in $\{not\ b\}$ or in $\{not\ c\}$). There is no support for these assumptions. But the facts from P should override default assumptions from U. Therefore, the rejection of e_1 by σ should be considered to be blocked.

The facts from P are conflicting with respect to each root of a ρ^U-path to w_1, w_1 is not supported in U. On the contrary, $(\{not\ a\}, w_\perp), (\{not\ b\}, w_\perp), (\{not\ c\}, w_\perp)$ ρ^P and there is no reason to reject them. □

The idea of rejections blocking is the fundamental one.

Definition 30 (Blocking) Let programs P and U be given. The corresponding Kripke structures are denoted by $\mathcal{K}_P$ and $\mathcal{K}_U$. Let $(w_1, w_\perp) \in \rho^P$. Let L_1 and L_2 be conflicting literals.

A ρ^U-edge (w_0, w_1) is *blocked* iff $L_1 \in w_1$ and there is a ρ^P-path from $\emptyset$ to w such that $L_2 \in w$. □

Remark 31 The basic attitude behind the notion of blocking is as follows: default assumptions in a more preferred program may be falsified by facts from a less preferred program.

Notice that Definition 30 is sufficiently general: if a literal L_1 from w_0 is conflicting with a literal L_2 supported (by a path form $\emptyset$ in $\mathcal{K}_P$ (as in Example 29) then also $L_1 \in w_1$. □

Definition 32 (Overriding) A ρ^U-edge $e = (w_0, w_1)$ *overrides* a ρ^P-edge $(w_1, w_\perp)$ if e is not blocked. □

Definition 33 (Rejected edges) Consider $\mathcal{K}^P = (W, \rho^P)$ and $\mathcal{K}^U = (W, \rho^U)$. We say that $e \in \rho^P$ is rejected, if

- e is attacked by some $e' \in \rho^U$,
- or there is a ρ^U-edge $e' = (w_0, w_1)$, overriding $e = (w_1, w_\perp) \in \rho^P$.

The set of rejected edges is denoted by $Rejected_{\rho^U}(\rho^P)$. □

Definition 34 (Update on Kripke structures)

$$\mathcal{K}^U \oplus \mathcal{K}^P = \mathcal{K}^{U\oplus P} = (W, \rho^{U\oplus P})$$
$$\rho^{U\oplus P} = \rho^U \cup (\rho^P \setminus (Rejected_{\rho^U}(\rho^P))\square$$

The causal rule rejection principle is satisfied in updated Kripke structures:

Proposition 35 ([7]) *Let $\mathcal{P} = \langle P, U\rangle$ be a multiprogram, where $1 \prec 2$. Let $w \in Int_{\mathcal{L}}$, $r \in P$ and $w \models body(r)$.*

If $e = (w, w \cup \{head(r)\}) \in Rejected_{\rho^U}(\rho^P)$ then $r \in reject(\mathcal{P}, 2, w)$ and $r \in rjct(\mathcal{P}, 2, w)$.

Remark 36 ([7]) The converse of the Proposition 35 does not hold. If a rule is rejected, some edges "generated" by this rule may be not rejected. Let P be $\{a \leftarrow\}$, U be $\{not\ a \leftarrow b\}$ and w be $\{a, b\}$. Then $(\emptyset, \{a\}) \notin Rejected_{\rho^U}(\rho^P)$ but the rule $a \leftarrow$ is both in $reject(\mathcal{P}, 2, w)$ and in $rjct(\mathcal{P}, 2, w)$.

Therefore, the rejection defined within the frame of Kripkean semantics is more sensitive (able to make a more fine distinguishing) than a rejection of rules. □

Remark 37 There is no analogy of the Proposition 35 for the edges with the target $w_\perp$. It may happen that $(w, w_\perp) \in Rejected_{\rho^U}(\rho^P)$, but there is no rule $r \in P \cap reject(\mathcal{P}, 2, w)$ or in $P \cap rjct(\mathcal{P}, 2, w)$: According to the Definition 32 there is a rule $r \in P$ such that $w \models body(r)$ and $not\ head(r) \in w$. However, $not\ head(r)$ may be not introduced by a rule from U, it may be in the root of each path to w, see the Example 21.

Of course (similarly as in the Remark 36), if a rule is rejected, it is possible that $(w, w_\perp) \notin Rejected_{\rho^U}(\rho^P)$, where the transition from w to $w_\perp$ is generated by the rejected rule.

Problems with tautological and cyclic updates are removed in the Kripkean semantics. Tautologies do not influence Kripke structures.

Proposition 38 *Let P be program. Let $P' = P \cup \{r\}$, where $head(r) \in body(r)$. Then $\mathcal{K}^P = \mathcal{K}^{P'}$.* □

The following theorem shows that cycles from U do not cause an update of $\mathcal{K}_P$: paths generated by cycles are terminated in $\mathcal{K}^{P \oplus U}$ by $w_\perp$ if there is a conflict between a rule in P and a rule in the cycle of U:

Theorem 39 *Let for all $r \in U$ there is a $r' \in U$ such that $head(r) \in body(r')$. Let $w = \{head(r) : r \in U\}$.*

If there is a ρ^P-path $\langle \emptyset, \ldots, u\rangle$ such that $not\ head(r) \in u$ for some $r \in U$ then for each w' such that $w \subseteq w'$ holds that $(w', w_\perp) \in \mathcal{K}^{P \oplus U}$.

Notice now that good worlds of $\mathcal{K}^{U \oplus P}$ play the crucial role in the update semantics.

Finally, Kripkean semantics do not suffer from irrelevant updates of the type illustrated by Example 20 and it is able to recognize the conflicts not distinguishable by semantics based on the causal rejection principle.

Theorem 40 *Let $P \cup U$ be consistent. Then the good worlds in $\mathcal{K}^{P \cup U}$ coincide with good worlds in $\mathcal{K}^{U \oplus P}$.*

Observation 41 *If P and U are consistent, $P \cup U$ is inconsistent and there is no conflict between the rules then there can be a good world in $\mathcal{K}_{U \oplus P}$. See Example 21. We intend to express and to prove a generalization of this observation in a forthcoming paper.*

7 Conclusions

The results of the paper:

- a classification of the semantics based on the rule rejection principle is given,
- a Kripkean semantics of logic program updates is presented,
- the semantics enables to solve the problem of tautological, cyclic and irrelevant updates and it is able to recognize conflicts indistinguishable according to the causal rejection principle.

The work on this paper has been partially supported by a grant of the Slovak Agency VEGA, No 1/0173/03.

References

1. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A. *Semantics for Dynamic Logic Programming: a principled based approach.* In Proceedings of LPNMR 2004, Springer.
2. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C. *Dynamic Updates of Non-Monotonic Knowledge Bases.* The Journal of Logic Programming 45 (1-3):43-70, September/October 2000
3. Eiter, T., Fink, M., Sabbatini, G., Tompits, H. *On properties of update sequences based on causal rejection.* 2001
4. Homola,M. *Dynamic Logic Programming: Various Semantics are Equal on Acyclic Programs.* Proceedings of CLIMA V, 2004.
5. Leite, J. *Evolving Knowledge Bases.* IOT Press, 2003.
6. Leite, J.A., Alferes, J.J., Pereira, L.M. *Multi-dimensional dynamic knowledge representation.* In: Eiter, T., Faber, W., Truszczynski (Eds.): LPNMR 2001, Springer, 365-378
7. Mariničová, E. *Semantic Characterization of Dynamic Logic Programming.* Diploma Thesis, Comenius University, Bratislava, 2001
8. Šefránek, J. *A Kripkean Semantics for Dynamic Logic Programing.* Logic for Programming and Automated Reasoning, Springer, 2000
9. Šefránek, J. *Semantic considerations on rejection.* Proceedings of NMR 2004, Whistler, BC, Canada
10. Šefránek, J. *A dependency theory for logic program updates.* Proceedings of Znalosti 2005
11. Šiška, J. *Refined extension principle for multidimensional dynamic logic programs.* Proceedings of Znalosti 2005

A dependency theory for logic program updates

Ján Šefránek

Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia

sefranek@fmph.uniba.sk

Abstract. Dynamic aspects of knowledge represent a challenging research topic with many open problems. An approach to the field has been presented recently in the logic programming style. The approach is based on rejection of rules: if there is a conflict between rules then more preferred rules override those less preferred. However, meaning of a nonmonotonic knowledge base is determined both by rules and by nonmonotonic assumptions. The role of assumptions in the conflicts is also significant. Therefore, also beliefs dependent on assumptions might be affected by conflicts.

The approach presented in this paper is focused on the dependencies between beliefs. A dependency theory is introduced together with a set of postulates for solving conflicts between dependencies. The approach is evaluated with respect to well known troubles with cyclic and irrelevant updates in the approaches based on the causal rejection principle.

Keywords: dynamic logic programming, dependency theory, conflicts between dependencies, postulates for logic program updates

1 Introduction

Reasoning about dynamic knowledge (knowledge evolving with time, knowledge integrated/permanently integrating from multiple sources, knowledge reflecting the changing environment or changing point of view etc.) is a challenging research topic with a critical importance for practical applications. Moreover, it is a topic with many open theoretical problems of fundamental significance. A deeper understanding of the dynamics of knowledge belongs to the most interesting and hardest tasks in knowledge representation research.

Logic programming has been proved as a powerful framework suitable for theoretical investigation of knowledge representation [4, 2, 5]. Nevertheless, the topic of the changing knowledge has been studied only recently in the logic programming style, see [3, 6, 8, 9] and others.

The basic paradigm of the research is based on the observation that "not all the information borne by a logic program is contained within its set of models" [7] and focus has been shifted from interpretation updates to program updates. The main attention is devoted to the conflicts between rules. They are resolved according to a causal rejection principle: if there is a conflict between rules then more preferred rules override those less preferred.

Unfortunately, causal rejection principle does not apply to all relevant cases. There are some conflicts between programs that are not identifiable on the level of conflicts between rules. Moreover, the causal rejection principle applies with respect to some irrelevant – from the updates point of view – cases and produces undesirable consequences. Finally, there are some problems, not completely solved until now, caused by tautologies and cycles in the more preferred programs.

Our approach is sensitive not only to conflicts between rules, but also to conflicts between beliefs. Semantics of nonmonotonic knowledge bases is determined both by rules and by nonmonotonic assumptions. We argue that it is important to record in the semantics also dependencies on assumptions. A dependency theory is introduced in this paper in order to obtain a solid base for logic program updates. The language of generalized extended logic programs is used. This tool enables to express both assumptions that $\neg A$ is true and also that A is true.

The paper is structured as follows: A more detailed motivation and an analysis of the problem is given in Section 2. Basic prerequisites and elements of the dependency theory are presented in Section 3, semantics of *true because of* in Section 4. Then the core of the paper paper follows. It consists in an application of dependencies and of a notion of *true because of* on updates of logic programs. Postulates constraining updates are introduced. Finally, a discussion and evaluation of the approach is presented in the Section 7.

We summarize the main contributions of the paper: A dependency theory is constructed as a basis of logic program updates. Postulates identifying source of conflicts and specifying rejection of dependencies are presented. Our semantics [11] satisfies the postulates.

The approach of [11] enables to distinguish finer conflicts than the conflicts between the rules, therefore it does not suffer from the troubles caused by cyclic and irrelevant updates. As regards the relation to our Kripkean semantics presented in [11], we are aiming in this paper to proceed to more foundational issues – dependency theory and postulates for updates are not presented in [11].

2 Motivation

Two rules r, r' are considered as conflicting in dynamic logic programming (DyLoP hereafter) paradigm iff their heads are L and $not\ L$, respectively.

We are aiming to argue in this Section that it is not sufficient to be focused only on conflicts between rules.

Two examples are given and analyzed below. First of them describes a conflict between programs that is not manifested by a conflict between rules. The conflict can be characterized in terms of dependencies between literals. Second example is aiming to show that some influence of dependencies between literals is latent also in the (classical) stable model semantics.

Example 1 Let U be a more preferred program than P.

$$P = \{a \leftarrow c\} \qquad\qquad U = \{b \leftarrow not\ a$$
$$c \leftarrow b\}$$

There is no conflict (according to the DyLoP paradigm) between rules of both programs. No rule can be rejected and the meaning of $P \cup U$ cannot be updated according to the causal rejection principle.

However, there is a sort of conflict between the programs: The literal a in the less preferred program P depends on the literal c. On the other hand, the literal c depends on the default assumption $not\ a$ in the more preferred program U. Hence, there is a dependency of a on $not\ a$. It can be resolved by rejecting the less preferred dependency. □

Example 1 shows that there are conflicts not identifiable as conflicts between rules. However, they are identifiable on a level of dependencies. Meaning of logic programs (and meaning of nonmonotonic knowledge bases) is determined both by rules and by nonmonotonic assumptions. Therefore, it is important to distinguish also "finer" conflicts involving (dependencies on) nonmonotonic assumptions.

The dependency of beliefs on assumptions is implicit also in the stable model semantics. Examine the example as follows:

Example 2 ([2]) Let

$$P = \{a \leftarrow not\ b \qquad b \leftarrow not\ a$$
$$c \leftarrow not\ a \qquad c \leftarrow not\ c\}$$
$$P' = P \cup \{c \leftarrow\}$$

While P has the only stable model $S = \{not\ a, b, c\}$, P' has two stable models – besides S also $S' = \{a, not\ b, c\}$. This seems to be a strange behaviour of the stable model semantics: adding a fact, which is true in the only stable model, increases the number of stable models. However, observe that the only model of P encodes in a way that the truth of c is dependent on the assumption $not\ a$ (it can be said that c is *true because of* $not\ a$). On the other hand, c is true in P' without a dependence on an assumption and, therefore, either a or b may be considered as true. □

Summary: A semantic treatment of dependencies on assumptions and of the conflicts between those dependencies is relevant for understanding and formalizing updates. We emphasize the importance of distinguishing those more "fine" conflicts as a key to overcoming well-known troubles with updates based on the causal rejection principle. Moreover, we are aiming at providing clear foundations for a theory of logic program updates. The discussion about approaches to logic program updates is often focused on some hard examples and some intuitions are provided in favor of (or against) some solutions. The arguments are based mainly on intuitions. We are interested in explicit principles underlying the approaches to logic program updates instead of ad hoc solving clashes of intuitions. This goal is realized in presented paper by a set of postulates. The postulates specify how to identify and how to solve conflicts between dependencies.

3 Basics

We assume a language of generalized extended logic programs (with default and explicit negation, both are allowed in heads of rules). Let $\mathcal{A}$ be a set of atoms. *Objective* literals are defined as follows: $Obj = \mathcal{A} \cup \{\neg A : A \in \mathcal{A}\}$. $Subj = \{not\ L : L \in Obj\}$ is the set of subjective literals. The set of *literals* is defined as $Lit = Obj \cup Subj$. Notation: $\neg(\neg A)) = A$, $not\ (not\ L) = L$.

The set of conflicting pairs is denoted by CON and for each $A \in \mathcal{A}$ holds $(A, \neg A) \in CON$, similarly $(L, not\ L) \in CON$ for each $L \in Obj$. It is not excluded that other pairs of literals belong to CON. A set of literals S is called *consistent* iff $(S \times S) \cap CON = \emptyset$, otherwise it is called *inconsistent*.

A *rule* is each expression of the form

$$L \leftarrow L_1, \ldots, L_k,$$

where $k \geq 0$, L, L_i are literals. If r is a rule of the form as above, then L is denoted by $head(r)$ and $\{L_1, \ldots, L_k\}$ by $body(r)$; $body(r)$ consists of two disjoint sets $body^+ \subseteq Obj$ and $body^-(r) \subseteq Subj$.

A finite set of rules is called *generalized extended logic program* (program hereafter). Each program might be considered as a definite program, where literals of the form $\neg A, not\ L$ are handled as new atoms. Hence, for each program exists his least model.

An *interpretation* is a consistent set of literals. A *total* interpretation is an interpretation such that for each literal L either $L \in I$ or $not\ L \in I$.

Definition 3 Let P be a program and *least* be the operator assigning the least model to a definite program. Let S be an interpretation and $S^- = \{L \in Subj \mid L \in S\}$.

Then S is a *partial stable* model of P iff $least(P \cup S^-) = S$. If partial stable model is a total interpretation, it is called *stable model*.

A program is *consistent* iff it has a stable model.

Let P be a program. The set $\bigcup_{r \in P} body^-(r)$ is called the set of *assumptions* in P (we denote it by $Ass(P)$).

Definition 4 (Dependency) Let a program P be given. A literal L *depends directly* on a set of literals V with respect to P (notation $L <_P V$) iff there is a rule $r \in P$ such that $L = head(r)$ and $V = body(r)$.

A literal L *depends* on a set of literals V with respect to P (notation $L \ll_P V$) iff $L <_P V$ or there is a set of literals U such that $L <_P U$, $L' \in U$, $L' \ll_P W$ and $V = (U \setminus \{L'\}) \cup W$.

A literal L_1 *depends* on a literal L_2 with respect to a program P ($L_1 \ll_P L_2$) iff there is a set of literals V such that $L_1 \ll_P V$ and $L_2 \in V$.

A set of literals W_1 depends on a set of literals W_2 with respect to a program P ($W_1 \ll_P W_2$) iff for each literal $L \in W_1$ holds that $L \ll_P W_2$. □

The set of all dependencies with respect to a program P is denoted by Dep_P.

Fact 5 *If $L \ll_P W$ then there is a sequence $L <_P W_1, L_1 <_P W_2, \ldots, L_{k-1} <_P W_k = W$, $L_i \in W_i, i = 1, \ldots, k-1$.* □

Definition 6 (Initial witness) Let be $L \ll_P W$ and $L <_P W_1$, $W_1 \ll_P W$. It is said that $L <_P W_1$ is an *initial witness* of $L \ll_P W$.

Definition 7 Let $L \ll_P W$ and for no $L' \in W$ there is U such that $L' \ll_P U$. Then it is said that L is *P-grounded* (or grounded) in W.

A literal L is called *supported* in a program P iff $L \ll_P \emptyset$. A set of literals S is supported in P iff each of its members is supported in P. □

A dependency $L \ll_P W$ is called *cyclic* iff $L \in W$. A literal cannot be grounded because of a cyclic dependency.

Proposition 8 *Let $L \ll_P W$ and $L \in W$. Then L is not grounded in W.*

Example 9 Let P be as follows:

$$a \leftarrow$$
$$b \leftarrow a$$
$$c \leftarrow not\ d$$

Then $Dep_P = \{b \ll_P \{a\}, c \ll_P \{not\ d\}, a \ll_P \emptyset, b \ll_P \emptyset, b \ll_P a, c \ll_P not\ d\}$, a, b are supported in P. Finally, $Ass(P) = \{not\ d\}$ and c is grounded in $\{not\ d\}$. □

4 True because of

Stable model semantics of a logic program P is determined by two components: by a set $A \subseteq Ass(P)$ of default assumptions and by the deductive closure of $P \cup A$. Therefore, each literal true in a stable model can be considered as true with respect to the corresponding set of assumptions. On the other hand, we will see that the stable model semantics is not able to distinguish some important aspects of meaning which might be expressed in terms of *true because of*.

Example 10 Consider the set of objective literals $\{a, b\}$ and a program P:

$$a \leftarrow not\ b$$
$$b \leftarrow not\ a$$

A straightforward idea to consider simultaneously a true because of $not\ b$ and b true because of $not\ a$ is certainly not a sound one. The set $\{a, b, not\ b, not\ a\}$ (the set of all assumptions and their consequences) cannot serve as a coherent set of beliefs. □

We are aiming to base the concept of *true because of* on sets of assumptions, but carefully – maintaining consistency (in accordance with the stable model semantics).

Definition 11 Let P be a program, $A \subseteq Ass(P)$, L be a literal.

It is said that L is in P *true because* of A (notation $val_P^A(L) = t$) iff $L \in A$ or $L \ll_P A$, L is grounded in A.

A is called a *sound* set of assumptions with respect to P iff the set $\{L \in Lit : val_P^A(L) = t\}$ is consistent. □

The notion of true because of is "immune" to cyclic dependencies.

Proposition 12 *If L is not grounded in A then $val_P^A(L)$ cannot be t.*

Definition 13 Let A be a sound set of assumptions with respect to P.

If $(L_1, L_2) \in CON$ and $val_P^A(L_1) = t$ then $val_P^A(L_2) = f$. If neither $val_P^A(L_1) = t$ nor $val_P^A(L_1) = f$ then $val_P^A(L_1) = u$. □

Example 14 Consider $P = \{$

$$a \leftarrow b$$
$$b \leftarrow c$$
$$c \leftarrow a\}.$$

Observe that $Ass(P) = \emptyset$ and $val_P^\emptyset(L) = u$ for each literal L.

There are programs with more sets of assumptions allowing to define val_P^A:

Example 15 Consider Example 10. $Ass(P) = \{not\ b, not\ a\}$, sound sets of assumptions are $A_1 = \{not\ b\}, A_2 = \{not\ a\}, A_3 = \emptyset$.

It holds that $val_P^{A_1}(a) = t = val_P^{A_1}(not\ b)$, $val_P^{A_1}(not\ a) = f = val_P^{A_1}(b)$ and $val_P^{A_2}(b) = t = val_P^{A_2}(not\ a)$, $val_P^{A_2}(a) = f = val_P^{A_2}(not\ b)$ and $val_P^{A_3}(L) = u$ for each literal L. □

For some sets of assumptions (and for a given program P) function val_P^A is not defined:

Example 16 Let be $P = \{a \leftarrow, \neg a \leftarrow\}$. Then $val_P^\emptyset(a) = t = val_P^\emptyset(\neg a)$, but also $val_P^\emptyset(a) = f = val_P^\emptyset(\neg a)$. Therefore, $val_P^\emptyset$ is not defined. □

Definition 17 A set of assumptions $A \subseteq Ass(P)$ is called a *maximal sound set of assumptions* in P iff

- there is a finite set of sound sets of assumptions $A_1, A_2, \ldots, A_k$, $i \geq 1$, such that $A = A_1 \cup \cdots \cup A_k$,
- the set of all literals L such that $val_P^{A_i}(L) = t$, $i = 1, \ldots, k$, is consistent,
- there is no superset of A with this property. □

Example 18 Let P be

$$a_1 \leftarrow not\ b_1$$
$$b_1 \leftarrow not\ a_1$$
$$a_2 \leftarrow not\ b_2$$
$$b_2 \leftarrow not\ a_2$$

Maximal sound sets of assumptions: $A_1 = \{not\ b_1, not\ b_2\}, A_2 = \{not\ b_1, not\ a_2\}, A_3 = \{not\ a_1, not\ b_2\}, A_4 = \{not\ a_1, not\ a_2\}$.

Theorem 19 *Let A be a maximal sound set of assumptions of a program P. Then the set $S = \{L \in Lit \mid val_P^A = t\}$ is a partial stable model of P. If S is a total interpretation then S is a stable model of P.*

Definition 20 A literal L is true because of a maximal sound set of assumptions $A = \{A_1 \cup \cdots \cup A_k\}$ (notation: $val_P^A(L) = t$) iff there is A_i such that $val_P^{A_i}(L) = t$. If $val_P^A(L) = t$ and $(L, L') \in CON$, then $val_P^A(L') = f$. If neither $val_P^A(L) = t$ nor $val_P^A(L') = t$ then $val_P^A(L) = u = val_P^A(L')$. □

Definition 21 Let A be a maximal sound set of assumptions for a program P, A' be a set of subjective literals such that $A \subseteq A' \subseteq Subj$ and S be $\{L \in Lit \mid val_P^A(L) = t\}$.

It is said that A' is a *completion* of A for P w.r.t. S iff $S \cup A'$ is a total interpretation.

We extend the definition of val for completions:

- if $val_P^A(L) = t$ then $val_P^{A'}(L) = t$,
- if $val_M^A(L) = f$ then $val_P^{A'}(L) = f$,
- if $not\ L \in A' \setminus A$ then $val_P^{A'}(not\ L) = t$ and $val_P^{A'}(L) = f$. □

Example 22 Let P be $\{a \leftarrow b\}$. Suppose that Lit is $\{a, b, \neg a, \neg b, not\ a, not\ \neg a, not\ b, not\ \neg b\}$ and CON is

$$\{(a, \neg a), (b, \neg b), (a, not\ a), (b, not\ b), (\neg a, not\ \neg a), (\neg b, not\ \neg b)\}.$$

Empty set is a sound set of assumptions: it holds that $val_L^{\emptyset}(L) = u$ for each literal L. $A' = \{not\ a, not\ b, not\ \neg a, not\ \neg b\}$ is a completion of $\emptyset$ for P: all assumptions are true because of A', and all objective literals are false with respect to A'. □

Theorem 23 *If S is a completion of a maximal sound set of assumptions for a program P then S is a stable model of P.*

5 Updates and dependencies

We intend to apply the dependency theory to a specification (constraining) of logic program updates. Our main point is that it is necessary to proceed (within the mainstream

of research devoted to logic program updates) from the approach based primary on intuitions to the approach based on clear foundations when analyzing problems of logic program updates.

Let P and U be programs.[1] The former is called original program, the later updating program in dynamic logic programming paradigm (DyLoP hereafter) [3]. The result of the update is called updated program and denoted by $P \oplus U$.

Updates should obey a principle of *inertia*, see for example [3, 6]. The intuitive meaning of this principle is as follows: The truth of literals from P remains unchanged in $P \oplus U$ unless it is directly affected by the update. Another principle emphasized by researchers in belief revision is the minimality of change [6]: if an update is required then unnecessary losses of information are to be avoided.

We motivate our approach by some examples. First a very simple one.

Example 24 Let P be $\{\neg a \leftarrow\}$ and U be $\{a \leftarrow\}$. In terms of dependencies: $\neg a \ll_P \emptyset$, $a \ll_U \emptyset$.

U is more preferred program than P, therefore it is natural to reject the dependency $\neg a \ll_P \emptyset$. This principle is expressed by the postulate P2 in Section 6. The set of updated dependencies $Dep_{P \oplus U}$ is $\{a \ll_{P \oplus U} \emptyset\}$. □

Example 25

$$P = \{a \leftarrow \qquad\qquad U = \{not\ a \leftarrow not\ b\}$$
$$b \leftarrow a\}$$

In dynamic stable semantics [8] the meaning of updated program $P \oplus U$ is specified by two dynamic stable models $\{a, b\}$ and $\{not\ a, not\ b\}$, but the stable model of $P \cup U$ is $\{a, b\}$.

According to our postulate P1, if $P \cup U$ is a consistent program then no update is needed (this requirement fits well the principle of minimal change). Hence, if Dyn is a "dynamic" semantics based on a "static" semantics Sem and $P \cup U$ is a consistent program then should hold that $Dyn(P \oplus U) = Sem(P \cup U)$.

If $P \cup U$ is consistent then all dependencies from both P and U hold also in $P \oplus U$, i.e. $Dep_P \cup Dep_U \subseteq Dep_{P \oplus U}$:

$$a \ll_P \emptyset \qquad\qquad not\ a \ll_U \{not\ b\}$$
$$b \ll_P \{a\}$$
$$b \ll_P \emptyset$$

The crucial task is to extend concepts of *true because of* to $P \oplus U$.

$$val_P^{\emptyset}(a) = t = val_P^{\emptyset}(b)$$
$$val_P^{\emptyset}(not\ a) = f = val_P^{\emptyset}(not\ b)$$
$$val_U^{\{not\ b\}}(not\ a) = t = val_U^{\{not\ b\}}(not\ b)$$
$$val_U^{\{not\ b\}}(a) = f = val_U^{\{not\ b\}}(b)$$

[1] Only the elementary case is considered in this paper because of limited size and also because of a need to explain the basic ideas for a simple setting. An extension to the general case is a straightforward one, however the formulation is a little bit more complicated.

Resolving the conflicts between dependencies is the main problem of our paper. Our goal is to solve the conflicts between dependencies (and, hence in the framework of the semantics of *true because of*).

We propose a strategy as follows: If there is a conflict between dependencies then dependencies from U override those from P unless assumptions from U are falsified by "facts" from P (i.e. by literals supported by empty sets of assumptions from P). Therefore, $val^{\emptyset}_{P}(not\ b) = f$ and we do not want to accept $\{not\ b\}$ as a sound set of assumptions for $P \oplus U$ (see postulate P4).

$$val^{\emptyset}_{P\oplus U}(a) = t = val^{\emptyset}_{P\oplus U}(b)$$
$$val^{\emptyset}_{P\oplus U}(not\ a) = f = val^{\emptyset}_{P\oplus U}(not\ b)$$

The dependency of $not\ a$ on $not\ b$ holds also in $P \oplus U$ but we do not consider $\{not\ b\}$ as a sound set of assumptions. □

Two important types of conflicts are distinguished:

1. conflicting beliefs depend on the same belief set,
2. conflicts between assumptions.

The first type corresponds to conflicts between rules. The second cannot be expressed in terms of conflicts between rules.

6 Postulates

Postulates specify conditions for rejecting (better to say, ignoring) dependencies affected by conflicts. In dependency theories characterized by [12] conflict resolution is performed by the addition of a set of internally generated justifications. In DyLoP it is performed by rejection of some rules (justifications). We are aiming at conflict resolution on purely semantic level and no addition or rejection of rules is specified. Rules may contain an useful information, even if they are responsible for a conflict with respect to the current state of knowledge (or knowledge base).

We can now list the postulates. The conflicts generated by one (inconsistent) program are not handled by the postulates in this paper – we are not interested in this problem. However, it is easy to add a postulate of the form as follows. If $L_1 \ll_P W$, $L_2 \ll_P W$ and $(L_1, L_2) \in CON$, then both dependencies are ignored (rejected).

Common assumptions for all postulates and examples: $(L_1, L_2) \in CON$, S is a set of literals. We assume that $P \cup U$ is a consistent program in the postulate P1, in P2, P3 it is assumed that $P \cup U$ is inconsistent.

P1 If S is a (partial) stable model of $P \cup U$ then for each $L_1 \in S$ holds $val^{S^-}_{P\oplus U}(L_1) = t$ and for each L_2 holds $val^{S^-}_{P\oplus U}(L_2) = f$, otherwise – if S is not a (partial) stable model of $P \cup U$ – then $val^{S}_{P\oplus U}$ is not defined,

P2 Let $L_1 \ll_P W$, if $L_2 \ll_U W$, $L_1 \not\ll_U W$, then each initial witness $L_1 <_P W_1$ (denoted by d) is rejected ($d \in Rej$),

P3 If $L_1 \ll_P W$ and $W \ll_U \{L_2\}$ and $L_1 \not\ll_P \emptyset$ then each initial witness $d = (L_1 <_P W_1)$ is rejected ($d \in Rej$),

P4 If $val_P^{\emptyset}(L_1) = t$, $val_U^{\emptyset}(L_2) \neq t$ and $L_2 \in Ass(U)$ then $val_{P \oplus U}^{\{L_2\}}$ is not defined.

P5 If $val_P^{\emptyset}(L_1) = t$, $L_2 \ll_U \{L\}$, $L \in Ass(U)$, $L \not\ll_U \emptyset$ then $val_{P \oplus U}^{\{L\}}$ is not defined.[2]

The dependencies can be rejected only according to the postulates P1 – P3.

The following examples illustrate Postulates P3 and P5.

Example 26 (P3) Let P be $\{a \leftarrow c\}$ and U be $\{b \leftarrow not\ a; c \leftarrow b\}$.
$a <_P \{c\}$ is rejected because of $\{c\} \ll_U \{not\ a\}$. □

Example 27 (P5) Let P be $\{a \leftarrow; b \leftarrow a\}$ and U be $\{not\ b \leftarrow not\ c\}$. Then $val_{P \oplus U}^{\{not\ c\}}$ is not defined.

7 Discussion

Postulates P1 – P5 are intended to specify the semantics of the update of program P by the program U. The update is denoted by $P \oplus U$. The semantics is based on the notions of dependency and true because of. It should be emphasized that we do not propose an update on syntactic level (on the level of programs). The update is focused on the semantics.

We are going to sketch the basic definitions. First we define the set of all non-rejected *direct* dependencies in $P \oplus U$.

Definition 28 Let $Direct_P$ ($Direct_U$) be the set of all direct dependencies in Dep_P and Dep_U. Then $Direct_{P \oplus U} = (Direct_P \cup Direct_U) \setminus Rej$. □

The relation $\ll_{P \oplus U}$ is defined exactly according to Definition 4. Notions of sound, maximal sound set of assumptions w.r.t. $P \oplus U$ and three-valued or two-valued valuations are defined as in Section 4 except of the requirement to satisfy Postulates P4 and P5. Thus, we have a "semantic view" on $P \oplus U$.

A Kripkean semantics of (multidimensional) dynamic logic programs is presented in [11]. A Kripke structure is assigned to each program. The structure records – in a sense – dependencies between sets of literals. An update operation is defined on Kripke structures. We present (without the necessary prerequisites because of the limited size of the paper) the main idea of the theorem claiming that Kripkean semantics is correct with respect to our postulates. A detailed exposition will be presented in a forthcoming paper.

Theorem 29 *Postulates P1 – P5 are satisfied by the update operation on Kripke structures associated with logic programs.* □

[2] Postulates P4 and P5 are expressed for singletons because of the limited space, but they can be generalized to arbitrary sets of assumptions.

It is shown in [11] that Kripkean semantics does not suffer from tautological, cyclic and irrelevant updates and that it is able to recognize conflicts not distinguishable on the level of conflicts between rules. The same holds also for our dependency theory (for updates of dependencies and of the notion *true because of*).

We now proceed to two topics connected to the dependency theory. Our forthcoming research will be devoted to those topics.

First, dependency theory allows a more finer semantics than the stable semantics. Consider programs as follows.

Example 30

$$P_1 = \{a \leftarrow not\ b\}\ \ P_2 = \{a \leftarrow; not\ b \leftarrow\}$$
$$P_3 = \{a \leftarrow; b \leftarrow b\}\ \ P_4 = \{a \leftarrow not\ b; not\ b \leftarrow\}$$

$S = \{a, not\ b\}$ is the only stable model of all four programs. However, there are subtle differences in (the intuitive) meaning of the programs. We believe that the differences can be understood and described by the dependency theory presented in this paper. □

Second, there is a connection of dependency theory to the strong equivalence of logic programs, being a hot research topic in the field.

Definition 31 ([10]) Two programs P and Q are said to be *strongly equivalent* ($P \equiv_s Q$) iff for each program R the programs $P \cup R$ and $P \cup Q$ have the same sets of stable models.

A method of optimization of logic programs consists in replacing strongly equivalent programs in a context. A relation of dependencies to strong equivalence of logic programs is obvious. However, there are open and interesting problems connected to that relation.

8 Conclusions

A dependency theory for logic programs has been introduced in this paper. Dependencies are defined explicitly and they are used for a semantics based on the concept of *true because of*. Updates of logic programs are specified in our approach as rejections of some dependencies. Postulates P1 – P5 for specifying rational rejections of dependencies and for constraining sound sets of assumptions are introduced. We do not reject or insert new rules. Our goal is only to make a coherent view of the current state of a modular knowledge base (represented by logic programs). The rejection of dependencies is more sensitive (able to make a more fine distinguishing) than the rejection of rules.

The substantial improvement gained by our approach is based on the observation that the meaning of nonmonotonic knowledge bases is determined both by the rules and by the nonmonotonic assumptions. Therefore, updates should respect both conflicts between rules and conflicts caused by assumptions.

Main results:

- Our dependency theory enables to solve the problem of cyclic and irrelevant updates,
- It is able to identify the conflicts not distinguishable by the causal rejection principle.
- The postulates P1 – P5 (contrary to the AGM postulates [1] and other postulates of that sort) specify dependencies responsible for the conflicts between beliefs (and/or belief sets); moreover, they allow to handle nonmonotonic dependencies based on assumptions.

Finally, I thank Martin Homola for extensive comments to a previous version of the paper. I thank also Slovak agency VEGA for the support of the project 1/0173/03.

References

1. Alchourron, C., Gärdenfors, P., Makinson, D. *On the logic of theory change. Partial meet contraction and revision functions.* Journal of Symbolic Logic, 50:510-530, 1985
2. Alferes, J., Pereira, L. *Reasoning with Logic Programming.* Springer 1996
3. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C. *Dynamic Updates of Non-Monotonic Knowledge Bases.* The Journal of Logic Programming 45 (1-3):43-70, September/October 2000
4. Baral, C., Gelfond, M. *Logic Programming and Knowledge Representation.* Journal of Logic Programing 1994:19, 20: 73-148
5. Dix, J., Brewka, G. *Knowledge Representation with Logic Programs.* Universität Koblenz-Landau, 1996
6. Eiter, T., Fink, M., Sabbatini, G., Tompits, H. *On properties of update sequences based on causal rejection.* In: Theory and Practice of Logic Programming 2(6), 2002
7. Leite, J., Pereira, L. *Generalizing Updates: from models to programs.* In: Logic Programming and Knowledge Representation. LNCS 1471, Springer 1998
8. Leite, J. *Evolving Knowledge Bases.* IOT Press, 2003.
9. Leite, J.A., Alferes, J.J., Pereira, L.M. *Multi-dimensional dynamic knowledge representation.* In: Eiter, T., Faber, W., Truszczynski (Eds.): LPNMR 2001, Springer, 365-378
10. Lifschitz, V., Pearce, D., Valverde, A. *Strongly equivalent logic programs.* ACM Transactions on Computational Logic, 2(4):526-541, 2001
11. Šefránek, J. *Semantic considerations on rejection.* Proceedings of NMR 2004, Whistler, BC, Canada.
12. Witteveen, C., Brewka, G. *Skeptical reason maintenance and belief revision.* Artificial Intelligence 61 (1993), 1-36

Klasifikace Suffix Tree frázemi - srovnání s metodou Itemsets

Roman Tesař, Karel Ježek

Katedra Informatiky a výpočetní techniky, Západočeská Univerzita v Plzni,
Univezitní 8, 306 14, Plzeň

{romant, jezek_ka}@kiv.zcu.cz

Abstrakt. V článku je prezentován postup klasifikace pomocí Suffix Tree (ST) frází. Popsán je způsob jejich získání, ohodnocení a použití ke klasifikaci textů. Konzultovány jsou výhody a nevýhody tohoto postupu, které jsou průběžně srovnávány s metodou Itemsets, ze které princip klasifikace Suffix Tree frázemi vychází. Popsán je také způsob nastavení prahové hodnoty pro zařazení dokumentu do zvolených tématických tříd. Prostor je věnován i porovnání vlivu průměrné délky dokumentu na celkovou úspěšnost klasifikace a srovnáván je také vliv itemsetů a ST-frází vyšších řádů u obou metod. V závěru samozřejmě nechybí srovnání dosažených výsledků klasifikace s dalšími rozšířenými metodami klasifikace textů.

Klíčová slova: klasifikace, kolekce dokumentů, Itemsets, ST-fráze, Suffix Tree, ohodnocení dokumentu, nastavení prahu

1 Úvod

V elektronické podobě se v současné době vydává stále více článků, časopisů i knih. Je v nich obsaženo obrovské množství informací, avšak ne všechny jsou pro každého uživatele zajímavé. Aby bylo možné se v této záplavě orientovat, je potřeba stanovit, jaký typ informací a z jaké oblasti se v jednotlivých článcích nachází. Existují již komerční instituce, které se zabývají vyhledáváním pouze relevantních informací pro své klienty. Jsou však pryč doby, kdy na tuto činnost byla vyhrazena skupina lidí procítajících elektronické texty z různých zdrojů. Použití aplikací implementujících různé metody automatické klasifikace textu je vzhledem k stále většímu objemu dat nezbytností.

Klasifikátory využívající princip strojového učení ovšem potřebují ke svému natrénování a k následnému zařazení klasifikovaného článku množinu textů zastupující jednak jednotlivé oblasti, které jsou pro daného uživatele relevantní a jednak oblast, která je pro uživatele nezajímavá. Pokrýt ovšem opravdu všechny existující tématické okruhy, které nejsou pro daného uživatele zajímavé, není vzhledem k jejich obsahové šířce možné.

V tomto článku se věnujeme metodě využívající ke klasifikaci textu ohodnocených Suffix Tree frází, která tento nedostatek odstraňuje. Její princip a dosažené výsledky průběžně konzultujeme s metodou Itemsets, která je svým principem velmi podobná a která dovoluje několik modifikací.

2 Vlastnosti metody Itemsets

Při návrhu a konstrukci klasifikátoru využívajícího Suffix Tree fráze jsme vycházeli z principu metody Itemsets popsané v [3], [4] a [5], která úspěšně klasifikuje zejména krátké textové dokumenty. Její nevýhodu bohužel představuje především fáze trénování, kdy jsou pomocí Apriori algoritmu (viz [5]) hledány časté množiny slov (itemsety). Složitost tohoto algoritmu je pro 1-itemsety lineární. Pro nejhorší případ je s rostoucím K při hledání K-itemsetů vyšších řádů kombinatorická, a to jak složitost časová, tak i složitost paměťová. Výhodu ale představuje rychlá fáze klasifikace této metody spočívající v přičítání ohodnocení předem natrénovaných K-itemsetů, jejichž všechny jednotlivé termy (slova) se v dokumentu vyskytují současně.

Zajímavá je skutečnost, že použití K-itemsetů, kde K>1, nepřineslo žádné zlepšení úspěšnosti klasifikace (viz kapitola 4). V důsledku potom použití 1-itemsetů, tedy jednotlivých slov, přináší této metodě nejvyšší úspěšnost. Dle našeho názoru je to dáno faktem, že jednotlivá slova K-itemsetu nalezeného Apriori algoritmem spolu netvoří souvislou frázi, ale jen množinu slov často se společně vyskytujících v trénovacích dokumentech. To je bezpochyby použitelné pro krátké dokumenty obsahujících řádově desítky slov, které jsou pokud možno tématicky dostatečně vzdálené. Pro delší dokumenty obsahující větší počet slov již úspěšnost klasifikace klesá v důsledku zvětšující se pravděpodobnosti nalezení většího počtu K-itemsetů společných několika klasifikačním třídám.

Stejný problém nastává, pokud budeme chtít klasifikovat do více tříd dokumenty tématicky velmi podobné. Opět budou Apriori algoritmem nalezeny K-itemsety obsahující stejná nebo podobná slova pro více tříd, což může vést ke snížení úspěšnosti klasifikace. V případě, že bychom chtěli klasifikovat dokumenty jen do dvou tříd (například závadné/nezávadné), je navíc nezbytné mít k dispozici kolekce dokumentů pro obě třídy, abychom mohli klasifikátor založený na metodě Itemsets natrénovat.

Na základě těchto poznatků jsme vytvořili klasifikátor, který uvedené nedostatky umožňuje alespoň částečně odstranit. K jeho konstrukci jsme využili fráze získané algoritmem Suffix Tree.

2.1 Suffix Tree fráze versus Itemsety

Oproti K-itemsetům nejsou Suffix Tree fráze (dále jen ST-fráze) množiny slov současně se vyskytujících s určitou četností v trénovacích dokumentech (dokumentech zařazených do třídy). Jedná se o posloupnost po sobě následujících slov, které se v množině trénovacích dokumentů vyskytují se zadanou četností. Intuitivně by tedy ST-fráze měly lépe vystihovat tématické okruhy představující jednotlivé klasifikační třídy a to i v případě, že si budou velmi podobné. Vycházíme přitom z předpokladu, že tématicky blízké dokumenty budou obsahovat stejná nebo podobná slova. Na následujícím jednoduchém přikladu jsme si ověřili, že metoda Itemsets by většinu takovýchto dokumentů klasifikovala do stejné třídy.

Tab. 2.1 Několik vybraných natrénovaných 1, 2 a 3-itemsetů pro závadnou třídu

1-Itemsety	ohodnocení
free	93,25
adult	88,25
hardcore	83,06
teen	79,75
picture	70,81
xxx	67,43
gallery	66,25

2-itemsety	ohodnocení
you hardcore	73,18
amateur free	77,25
girl teen	72,93
free hot	67,25
xxx adult	60,53
gallery picture	56,34
site free	48,12

3-itemsety	ohodnocení
site hardcore free	70,32
adult girl teen	68,58
maiden babe your	59,15
free site male	53,80
woman school asia	47,65
guy great xxx	42,30
latina lady adult	34,24

K jejímu natrénování jsme použili 200 závadných internetových stránek sexuálního charakteru[2] a 2000 nezávadných stránek obsahujících různá témata. Jednotlivé třídy (závadné/nezávadné) charakterizovalo 60 nejčastějších K-itemsetů pro K≤3 vytvořených Apriori algoritmem. Testovali jsme nezávadné stránky, u kterých jsme očekávali chybnou klasifikaci na základě předpokladů vysvětlených v následujícím odstavci. K výpočtu kvalitativní váhy itemsetů jsme v průběhu veškerých testů použili stejný vztah, který se osvědčil a byl použit i v [5].

Ačkoli na adrese *www.bhs.org.uk/Horse_clip-art/icons.htm* žádný sexuálně orientovaný materiál nenajdeme, byla klasifikována metodou Itemsets jako závadná. Důvod je ten, že se v HTML kódu této stránky vyskytují termy "free, gallery, picture". Tyto termy se bohužel vyskytují mezi častými itemsety charakterizujícími závadnou třídu a protože zde již není mnoho textu, ve kterém by mohly být nalezeny další natrénované termy patřící do nezávadné třídy, je tato stránka chybně klasifikována. Této skutečnosti také napomáhá fakt, že váhu klasifikace nesou 1-itemsety jak již bylo zmíněno dříve. Nelze tedy účinně přenést váhu klasifikace na itemsety vyšších řádů nastavením nižších váhových koeficientů 1-itemsetům. Navíc i některé natrénované 2 a 3 itemsety, jak je patrné z Tab. 2.1, mohou být závádějící. Pokud se například v textu dokumentu objeví trojice termů "site hardcore free", nemusí se vůbec jednat o závadnou stránku, možná právě naopak. Příkladem jsou i následující stránky, u kterých dochází k podobným problémům:

http://spgm.sourceforge.net/
http://www.ag.ohio-state.edu/~vegnet
http://www.travelplan.com/gallery4.htm
http://www.hikyaku.com/gallery/gallery.html
a další…

Vznikají zde ovšem i jiné potíže. Ohodnocení přiřazené K-itemsetům závadné třídy je většinou velmi vysoké, protože stránky sexuálního charakteru jsou velmi úzce zaměřeny a obsahují často stejná slova (1-itemsety) a mnohdy i slovní spojení. To samozřejmě neplatí pro nezávadnou třídu, zejména pokud se snažíme jednotlivými dokumenty pokrýt co nejvíce tématických okruhů a oblastí lidské činnosti. V důsledku tedy, pokud se v závadné trénovací kolekci bude vyskytovat dostatečně často nezávadné

[2] Tento článek vznikl během projektu na filtraci závadných internetových stránek, proto kolekce takového typu

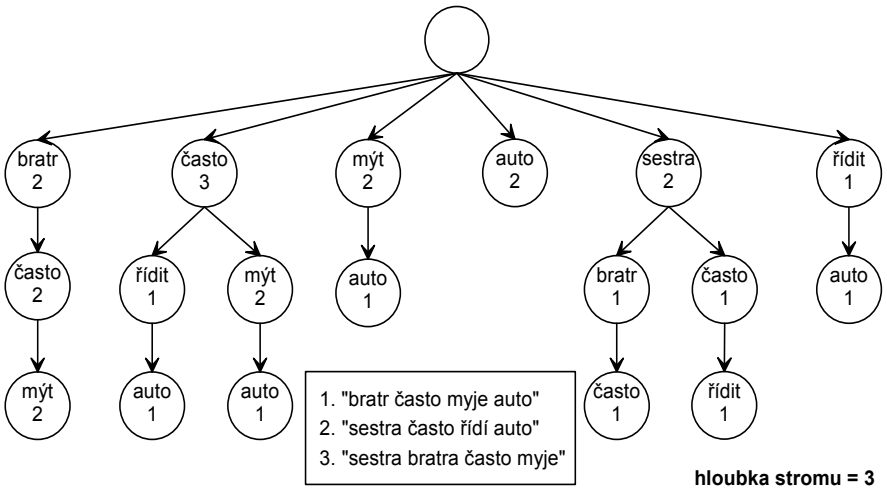
slovo, natrénuje se jako závadný itemset. Pokud se potom v klasifikovaném dokumentu takové slovo objeví, může toto slovo svým ohodnocením zapříčinit zařazení tohoto dokumentu do závadné třídy, protože i při současném výskytu několika natrénovaných nezávadných slov nemusí jejich ohodnocení stačit na zařazení dokumentu do správné třídy. Navíc ani pokud bychom měli k dispozici obrovskou kolekci nezávadných dat nemůžeme pokrýt tuto oblast zcela. Nebylo by ani možné použít Apriori algoritmus pro natrénování K-itemsetů vyšších řádů z důvodu jeho příliš velkých časových a paměťových nároků (viz [3] a [5]).

Z těchto poznatků jsme usoudili, že podobné problémy mohou nastat i při klasifikaci do více tříd. Jak bude uvedeno dále, ke klasifikaci pomocí ST-frází postačuje mít k dispozici jen data závadné třídy, není nutné vytvářet kolekci nezávadných dokumentů. Tento fakt přináší samozřejmě snížení celkové složitosti, protože postačuje vytvořit ST-fráze jen pro třídu zastupující úzký tématický okruh, který chceme rozpoznávat.

3 Suffix Tree klasifikátor

3.1 Fáze trénování

Při trénování jsou v kolekci vybraných dokumentů, které jsou tématicky zaměřeny na oblast, kterou budeme chtít v budoucnu rozpoznávat, vyhlédávány opakované fráze. K jejich získání využíváme postupného vytváření Suffix Tree struktury zadané hloubky (princip a postup viz [1] a [2]). V jednotlivých uzlech ovšem uchováváme jen informaci o počtu výskytů dané fráze v trénovací kolekci (viz Obr. 3.1) a název termu (slova). Další údaje nejsou pro získání ohodnocených ST-frází nutné.



Obr. 3.1 Příklad struktury Sufix Tree vytvořené z trénovací kolekce tří dokumentů

Trénovací kolekci dokumentů, ze které chceme strukturu vytvořit, je vhodné nejprve lematizovat. Na kolekci, která nebyla lematizována, jsme v průběhu testování dosahovali vždy průměrně o 5% horší úspěšnosti klasifikace. Protože jsme testovali uspěšnost metody Sufix Tree a Itemsets jen na anglických textech (viz kapitola 4), použili jsme Porterův lemmatizátor (viz [7]) pro anglický jazyk, jehož programová realizace je dostupná z [9]. Slovník stopslov obsahoval 390 nejčastějších anglických slov převzatých z WAIS[3]. Ty je samozřejmě vhodné před počátkem trénování odstranit. Text jednotlivých dokumentů byl vždy zpracováván jako jeden dlouhý řetězec, na konce vět nebyl brán zřetel.

Z výsledné Sufix Tree struktury následně získáme jednotlivé fráze, jejichž maximální délka je rovna maximální hloubce stromu a na základě vzorce

$$F_{ohod} = \frac{F_{čet}}{N_{K,\max}} \cdot 100 \ [\%] \qquad (1)$$

určíme jejich ohodnocení. Jednotlivé symboly mají následující význam:

F_{ohod} = výsledné ohodnocení konkrétní fráze
$F_{čet}$ = četnost konkrétní fráze v trénovací kolekci
K = délka ST-fráze
$N_{K,max}$ = počet výskytů nejčastější fráze (délky K) v trénovací kolekci

Jak je ze vzorce (1) patrné, pro ST-fráze různých délek se mění hodnota ve jmenovateli sloužící jako normovací konstanta. Je to z toho důvodu, aby nebylo potlačeno

Tab. 3.1 Ohodnocené ST-fráze získané ze struktury na Obr. 3.1

ST-fráze	$F_{čet}$	F_{ohod} [%]	K (délka ST-fráze)	$N_{K,\max}$
často	3	100	*1*	*3*
bratr	2	67		
mýt	2	67		
auto	2	67		
sestra	2	67		
řídit	1	33		
bratr často	2	67	*2*	*2*
často myje	2	67		
často řídit	1	33		
mýt auto	1	33		
sestra bratr	1	33		
sestra často	1	33		
řídí auto	1	33		
bratr často mýt	2	67	*3*	*2*
často řídit auto	1	33		
často mýt auto	1	33		
sestra bratr často	1	33		
sestra často řídit	1	33		

[3] http://fog.bio.unipd.it/waishelp/stoplist.html

ohodnocení ST-frází větších délek. Použijeme-li totiž k normování jen nejčastěji se vyskytující frázi, bude se vždy jednak o frázi délky 1 a tím by došlo k znevýhodnění delších frází. Proto musíme normovat vždy hodnotou, která je rovna počtu výskytů nejčastější fráze délky odpovídající ohodnocované frázi.

Tabulka Tab. 3.1 obsahuje příklad již ohodnocených ST-frází délky 1,2 a 3 získaných ze Suffix Tree struktury zobrazené na Obr. 3.1, která byla vytvořena ze tří dokumentů uvedených na stejném obrázku.

3.2 Fáze klasifikace

Ve fázi klasifikace nejprve vybereme určitý počet ST-frází, které mají nejvyšší ohodnocení určené na základě vzorce (1). Vliv počtu a délky ST-frází na uspěšnost klasifikace diskutujeme v podkapitolách 4.1 a 4.2.

Testovaný dokument procházíme a hledáme výskyt jednotlivých ST-frází. Tento postup je obdobný jako u metody Itemsets, oproti které ovšem nehledáme současný výskyt jednotlivých termů K-itemsetu kdekoli v dokumentu. Jednotlivé termy ST-fráze délky větší než 1 se v dokumentu musí vyskytovat těsně za sebou ve stejném pořadí jako při svém natrénování. Při výskytu některé z ST-frází charakterizujících určitou třídu přičteme pro testovaný dokument této třídě celkovou váhu odpovídající ohodnocení nalezené ST-fráze. Po otestování výskytu všech vybraných ST-frází reprezentujících jednotlivé klasifikační třídy zařadíme testovaný dokument do třídy s nejvyšší vahou V_{max}. Dalším rozdílem je skutečnost, že metoda Itemsets uvažuje jen prostý výskyt jednotlivých itemsetů v dokumentu. Při klasifikaci Suffix Tree frázemi uvažujeme vícenásobný výskyt jednotlivých frází v testovaném dokumentu.

Samozřejmě můžeme chtít asociovat dokument s více třídami. V tom případě zařadíme klasifikovaný dokument do všech tříd, jejichž váha přesahuje zvolený práh z maximální dosažené váhy V_{max} (jedná se tedy o zvolené procento z V_{max}). V průběhu celého testování jsme dosahovali nejlepších výsledků s hodnotu prahu 75 % pro obě porovnávané metody.

4 Testování, dosažené výsledky

Pro porovnání vlastností obou metod jsme vytvořili dvě kolekce složené z krátkých a dlouhých dokumentů. Využili jsme k tomu dokumenty anglické kolekce Reuters Corpus Volume 1 z týdnů 1 až 5 zařazených pouze do kořenových tříd (Economics, Markets, Government/Social, Corporate/Industrial). Následující tabulka (Tab. 4) obsahuje charakteristiky těchto kolekcí.

Tab. 4 Charakteristiky použitých kolekcí

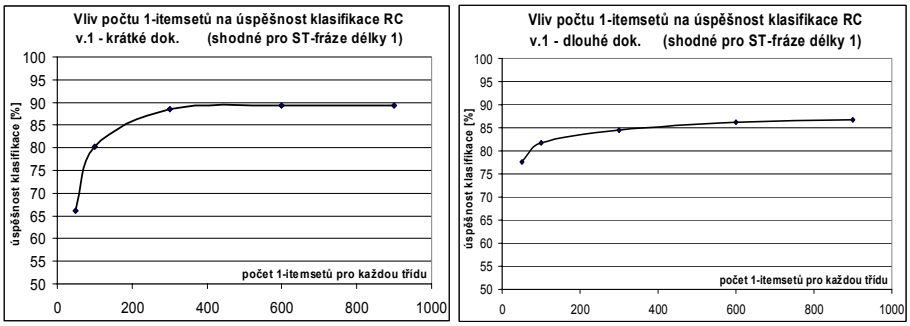
RC v.1 – KRÁTKÉ dokumenty		RC v.1 - DLOUHÉ dokumenty	
počet dokumentů celkem	9370	počet dokumentů celkem	10350
počet trénovacích dokumentů	2343	počet trénovacích dokumentů	2588
počet testovacích dokumentů	7028	počet testovacích dokumentů	7762
počet významových slov v dokumentech	**50 – 150 průměrně : 88**	počet významových slov v dokumentech	**300 – 3952 průměrně : 560**
počet všech slov	7244561	počet všech slov	10747726
počet významových slov	4306252	počet významových slov	6872532
počet různých významových slov	46653	počet různých významových slov	83900
počet nevýznamových slov	2938310	počet nevýznamových slov	3875194
průměrný počet tříd, do kterých je dokument klasifikován	1,5	průměrný počet tříd, do kterých je dokument klasifikován	1,8

4.1 Vliv počtu 1-itemsetů (ST-frází délky 1) na úspěšnost klasifikace

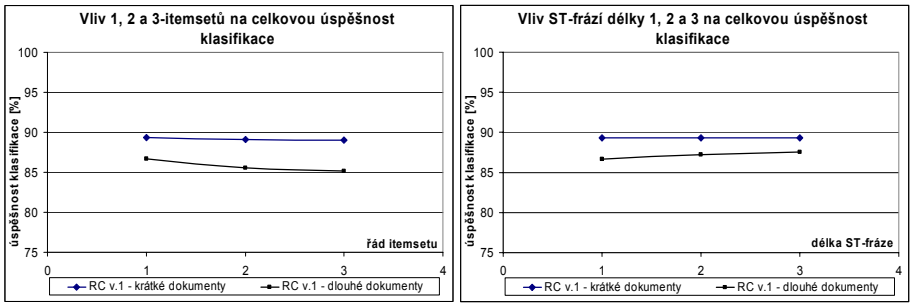
Při testování obou metod jsme se nejprve zaměřili na srovnání vlivu počtu 1-itemsetů na úspěšnost klasifikace. Míra celkové úspěšnosti byla v průběhu celého testování chápána jako průměr dosažených hodnot přesnosti a úplnosti. Protože natrénované 1-itemsety odpovídaly (včetně svého ohodnocení) natrénovaným ST-frázím délky 1 a protože princip fáze klasifikace je shodný, jsou výsledky z grafu 4.1 (kde je úspěšnost chápána jako průměr přesnosti a úplnosti) společné pro obě srovnávané metody.

Jak je z grafů patrné, u obou metod nastává pro krátké dokumenty od určitého počtu termů reprezentujících jednotlivé třídy "nasycení" a přidávání dalších termů již nemá větší vliv na úspěšnost klasifikace. Pro malý počet termů (50) samozřejmě docházelo k jevu, kdy v mnoha dokumentech nebyl nalezen žádný 1-itemset reprezentující některou z tříd. Z grafů je také zřejmé, že vhodným kompromisem mezi kvalitou klasifikace a složitostí procesu trénování bylo použití cca 400 termů.

Naopak pro kolekci delších dokumentů přidávání dalších termů reprezentujících jednotlivé třídy mělo smysl. S jejich vzrůstajícím počtem vzrůstala i úspěšnost klasifikace. Dochází zde však postupně k podobnému jevu jako v předchozím případě. Pro 800 termů a více se již projevuje vliv "nasycení". Počet 1-itemsetů (ST-frází délky 1) je tedy vhodné volit na základě předpokládané délky klasifikovaných dokumentů.



Graf 4.1 Porovnání vlivu počtu 1-itemsetů na celkovou úspěšnost klasifikace



Graf 4.2 Porovnání vlivu itemsetů a ST-frází vyšších řádů na úspěšnost klasifikace

Použití většího počtu by vedlo k zbytečnému zpomalení klasifikace dokumentů, použití nižšího počtu termů potom vede ke snížení celkové úspěšnosti klasifikace.

4.2 Vliv 1, 2 , 3-itemsetů a ST-frází délky 1, 2, 3

Dalším předmětem našeho testování byla klasifikace s použitím itemsetů a ST-frází vyšších řádů. Hodnoty z grafu 4.2 jen potvrzují co je již intuitivně zřejmé. U krátkých dokumentů nehrají delší ST-fráze významější roli. Zajímavé je však zjištění, že celková úspěšnost klasifikace u metody Itemsets je dána především 1-itemsety. Nemá tedy smysl generovat poměrně náročným Apriori algoritmem K-itemsety, kde K>1. Jak je z grafů patrné, úspěšnost se jejich použitím pro dlouhé dokumenty může zmenšovat. Je to pravděpodobně dáno skutečností, že s rostoucí délkou dokumentů se do procesu klasifikace zavlékají další obecné termy, které však pro svou obecnost nelze chápat jako charakteristické pro třídy, do kterých dokument skutečně patří. Větší délka dokumentů potom zvyšuje pravděpodobnost jejich případného současného výskytu a tím i možnost nesprávné klasifikace.

Pro krátké dokumenty již není vysvětlení mírného poklesu úspěšnosti klasifikace při použití itemsetů vyšších řádů tak jednoduché. Důvodem by mohl být stále příliš velký počet termů v testovacích dokumentech a skutečnost, že jednotlivé třídy jsou svým tématickým zaměřením poměrně široké. Natrénované 2 a 3-itemsety reprezentující jednotlivé třídy totiž měly výrazně menší váhu než itemsety získané v testu popsaném v podkapitole 2.1.

Tab. 4.2 Dosažené výsledky klasifikace pro jednotlivé metody

P = přesnost U = úplnost	**Naive Bayes**		**NBCI**		**TFIDF**		**Itemsets** (1,2,3)		**ST-fráze** (1,2,3)	
	P	U	P	U	P	U	P	U	P	U
RC v.1 – Krátké dokumenty	94,22	85,28	89,38	80,68	84,83	84,55	89,42	89,37	89,52	89,4
	89,75		***85,03***		***84,69***		***89,40***		***89,46***	

	Naive Bayes		**NBCI**		**TFIDF**		**Itemsets** (1,2,3)		**ST-fráze** (1,2,3)	
	P	U	P	U	P	U	P	U	P	U
RC v.1 – Dlouhé dokumenty	91,17	78,46	89,06	76,34	86,29	87,1	85,45	87,86	86,74	88,95
	84,82		***82,70***		***86,70***		***86,66***		***87,85***	

Naopak použitím delších ST-frází úspěšnost klasifikace roste, což se nejvíce projeví u delších dokumentů. Je to samozřejmě dáno jejich větší schopností rozlišit i velmi blízká klasifikační témata. Současně je oproti itemsetům výhodou menší pravděpodobnost jejich výskytu v dokumentech, které nepatří do třídy, kterou reprezentují.

4.3 Celkové výsledky klasifikace

V Tab. 4.2 jsou prezentovány konečné výsledky klasifikace na kolekci dlouhých a krátkých dokumentů. Hodnoty z této tabulky by neměly implikovat obecnou kvalitu jednotlivých klasifikačních metod, pro jiná vstupní data můžeme samozřejmě dostat jiné pořadí úspěšnosti prezentovaných metod.

Nejlepší hodnoty pro krátké dokumenty bylo dosaženo metodou Naive Bayes. Ta používá ke klasifikaci všech slov trénovací kolekce, což je v tomto případě nejlepší přístup. Nedochází zde totiž k zbytečné ztrátě informace vlivem hledání určitých častých položek, jako je tomu u ostatních metod. Již u krátkých dokumentů je patrný malý náskok v úspěšnosti klasifikace, který získávají ST-fráze oproti Itemsetům. Tento náskok se podle očekávání nejvíce projevil u kolekce delších dokumentů, kde již rozdíl mezi metodou Itemsets a ST-frázemi činil 1,2%. Delší dokumenty také podle očekávání lépe vyhovují metodě TFIDF, které umožnuje přítomnost většího počtu termů vytvořit relevantnější vektory reprezentující jednotlivé dokumenty a ve fázi klasifikace potom dochází k méně chybám. Popis méně známé metody NBCI je možné nalézt v [6].

Jak je z Tab. 4.2 patrné, obecně lepších výsledků jednotlivé metody dosáhly při klasifikaci kolekce krátkých dokumentů. Delší dokumenty pravděpodobně obsahují větší počet jednotlivých termů společných více třídám, což ztěžuje klasifikaci.

5 Závěr

Na testovacích kolekcích jsme ověřili, že výsledky klasifikace ST-frázemi jsou velmi dobré i v porovnání s jinými známými metodami. Použití delších ST-frází vede k lepším celkovým výsledkům než u metody Itemsets, což je nejvíce patrné u delších dokumentů. Použití 2 a 3-itemsetů nepřineslo žádné zlepšení. Naopak se celková úspěšnost klasifikace zhoršila.

Fáze trénování při klasifikaci ST-frázemi je algoritmicky méně náročná než u metody Itemsets. Složitost Apriori algoritmu totiž při hledání K-itemsetů výrazně roste se zvyšující se hodnotou K (viz [4]), zatímco vytváření Suffix Tree struktury má stále lineární složitost (viz [8]). Fáze klasifikace je u obou metod algoritmicky i časově srovnatelná.

Předmětem našeho dalšího zkoumání bude porovnání složitosti různých algoritmů pro získání ST-frází, testování možností dalšího zvyšování efektivity tohoto způsobu klasifikace a možnost kombinace ST-frází s dalším klasifikátorem založeným především na metodě Naive Bayes.

Reference

1. Grolmus P., Hynek J., Ježek K. User Profile Identification Based on Text Mining. Proc. of *6th Int. Conf. ISIM'03*, pp. 109-118, MARQ Ostrava, ISBN 80-85988-84-4
2. Grolmus P., Hynek J., Ježek K. Vyhledávání častých frází pro generování uživatelských profilů (in czech), *ITAT 2003,* pp.21-29, ISBN 80-7097-564-4
3. Hynek J., Jezek K. Document Classification Using Itemsets, *Proc.34th Int. Conf. MOSIS 2000, ISM 2000,* pp.97-102, ISBN 80-85988-45-3
4. Hynek J., Jezek K., Rohlik O. Short Document Categorization - Itemsets Method, *PKDD 4-th European Conference on Principles and Practice of Knowledge Discovery in Databases, Workshop Machine Learning and Textual Information Access.* Lyon, France, Sept.2000, pp.14-19
5. Hynek J., Ježek K.: Automatická klasifikace dokumentů do tříd metodou Itemsets, její modifikace a vyhodnocení (in czech), *Datakon 2001*, pp. 329-336, ISBN 80-227-1597-2
6. Kučera M., Ježek K., Hynek J. Text Categorization Using NBCI Method (in czech), *ZNALOSTI 2003*, Proc. pp.33-42, VŠB Technická univerzita Ostrava, ISBN80-248-0229-5
7. Porter, M.F., 1980, An algorithm for suffix stripping, Program, 14(3) : 130-137
8. Zamir O., Etzioni O. Web document clustering: A feasibility demonstration. In Proceedings of the *21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (*SIGIR '98*, Melbourne, Australia, Aug. 24–28), W. B. Croft, A. Moffat, C. J. van Rijsbergen, R., Wilkinson, and J. Zobel, Chairs. ACM Press, New York, NY, 46–54.
9. http://snowball.tartarus.org/

Annotation:

Suffix Tree phrases classification in comparison with Itemsets method

In this paper we present a text classification method using Suffix Tree (ST) phrases. We describe how to obtain ST-phrases from the training corpora, how to evaluate them and use them for text classification. Advantages and disadvantages of this approach are discussed and compared to the Itemsets method, which the Suffix Tree classification is based on. We also explain the way a threshold for multiclass classification is determined. We devote some time to examine the document length influence on classification effectiveness and also compare the impact of higher order Itemsets and ST-phrases in both methods. Of course, some comparison of the results obtained with other favourite text classification methods is provided at last.

Ontology Transformation Using Generalised Formalism

Petr Aubrecht, Monika Žáková, and Zdeněk Kouba

Department of Cybernetics Czech Technical University Prague, Czech Republic
{faubrech,zakovm1,koubag}@fel.cvut.cz

Abstract. This article proposes a framework for conversion of ontologies between different formalisms. The transformation is done using metamodels of both the source and target formalisms and optionally definition of mapping between them. The methodology provides two ways, how to do the conversion: with or without the mapping. The practical usage of the Generalised Ontology Formalism (GOF) is to transform ontologies repeatedly without human intervention. It is further shown how is the SumatraTT system used as a user interface to make such transformations easily.

Keywords: ontology, transformation, formalism, SumatraTT

1 Introduction

According to [1], one of the main purposes of ontologies is knowledge sharing and reuse. However there exist multiple ontology formalisms (and others are being developed) with specific advantages and drawbacks, supporting particular features or focusing on a simplification of some operation. Ontologies are created for a specific purpose in the most appropriate formalism satisfying needs of a narrow target community. The problem arises when an ontology is going to be used in another system supporting a different formalism. Although the basic idea of ontologies is to consolidate hierarchy of concepts, the formalisms/formats used are mutually incompatible. The biggest incompatibility we have met in the CIPHER project was between formalisms based on Lisp (Ontoligua etc.) and on RDF.

To solve the problem of ontology transformation between formalisms we proposed a Generalised Ontology Formalism (GOF). We analysed the existing formalisms to be able to translate all important features. The GOF formalism can be expressed as a graph with uniform vertices and six types of edges. The edge types represent the basic relations between concepts found in all considered formalisms and express the structural part of ontology. The design is focused mainly on frame-based formalisms and description logic, but it can also be used for other systems like lattices, topic maps, or even ER diagrams, Java class hierarchy, etc.

Different formalisms are handled by I/O gates, specific to the formalism. Each gate converts the ontology from the original form to the GOF and back. It also contains specific ontology of the formalism (FSO) defining a metamodel of the gate. The conversion is done either with or without connections to the FSO. Use of the FSOs leads to a more precise result, on the other hand it requires mappings for every pair of formalisms. For the GOF we also defined set operations for union and diff (mathematically $(O_1 \setminus O_2) \cup (O_2 \setminus O_1)$).

A user interface to the ontology transformation is provided by SumatraTT system [2]. SumatraTT is a universal data processing system, originally designed for data preprocessing for data mining. The presented framework contains methods for exporting both its gates and operations as SumatraTT modules. The graphical interface is used for easier setup of the transformation and for testing.

2 Known Approaches

According to [3], there are three main approaches to transformation between different knowledge representation formalisms:

Mapping Approach For every pair of formalisms a mapping is created which transforms expressions in the source formalism to expressions in the target formalism. It can be well adapted to the two specific formalisms and therefore involves lowest loss of information. However the number of transformations that have to be designed is relatively high and their properties must be checked individually. Thus this approach is feasible only for systems working with a relatively small and fixed set of formalisms.

Pivot Approach To avoid the necessity to create a large number of transformations one formalism is chosen as the pivot formalism. A transformation between two different formalisms is done via the pivot one. The pivot formalism has to be very expressive to enable lossless transformation of all other formalisms into it, especially if the system involves formalisms that are unlike each other. It has to be extended almost every time a new formalism has been added to the system and it tends to be biased towards one type of formalisms.

Layered Approach The third approach uses a layered architecture containing languages with increasing expressiveness [3]. There has been an attempt by W3C to provide a standard group of languages that would be layered on top of each other, using RDFS as the base. However other requirements on the higher ontology languages, especially decidability of reasoning, were more significant for their design than full backwards compatibility with RDFS. Therefore except for OWL-Full the ontological languages do not cover RDFS completely.

Family of Languages A new approach called the Family of languages approach is proposed in [4]. It is a generalization of layered and pivot language approach. The family of languages involves languages forming a semi-lattice with respect to some coverage relation. The coverage relation can be defined in a number of ways depending on the properties, which the transformation should preserve e.g. consequence-preserving. It can be proved that the properties of the transformation hold for all languages in the family.

3 Ontology Transformation

3.1 Formal Definitions

Ontology Definition For the investigation of transformations we need an exact mathematical definition of ontology that would allow us to draw conclusions about ontology transformations. Basically, ontology content can be divided into two basic parts: structural and procedural. The structural part is addressed far more often, for example by the Gruber's definition [5]. The structural part can be divided into a set of concepts and relations between concepts. The procedural part consists of restrictions and actions. Thus ontology is a 4-tuple:

$$Ontology = (Concepts, Relations, Restriction, Actions) \tag{1}$$

$$Concepts = concept \text{ set} \tag{2}$$

Concept is simply a node, represented by a unique literal. To simplify merging of concepts from different sources we use also a namespace as a part of the literal in the same way XML does. Finally, the namespace and a name of a *Concept* becomes an identification of the concept.

$$concept = string(\text{or } literal) \tag{3}$$

In our approach, the *Concepts* set is defined independently of formalism.

Formalism Definition Ontology formalism is defined by the triple *Relations, Restrictions, Actions*. A relation is an n-ary predicate $\text{rel}(x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n \in Concepts$. In the following text we use an abbreviation $Relation/n$ to express a set of relations with n arguments, for example $SubclassOf/2$ is a set of relations $subclassOf$ with two arguments:

$$SubclassOf/2 = \{rel(x_1, x_2) | rel = subclassOf,\ x_1, x_2 \in Concepts\} \tag{4}$$

We can say a formalism is a subset of another one, if all possible ontologies in one formalism are also valid in the other:

$$F_a \subset F_b \rightarrow Onto(F_a) \subset Onto(F_b) \tag{5}$$

Some examples of formalism definitions are shown below.

Taxonomy:

$$O_{taxonomy} = (Concepts, SubclassOf/2, \{\}, \{\}) \tag{6}$$

In OCML [6], like in all frame-based formalisms assignment (a specific slot is set to a particular value) has arity 3, as it connects the definition of the slot, the instance and the target value. Both the *Slot-type-validations* and *Actions* are Lisp functions connected to an event (assignment of a slot value).

OCML:

$$O_{OCML} = (\{c|c \in Class \cup Instance \cup Slot \cup Literal\}, \\ SubclassOf/2 \cup HasSlot/2 \cup Assignment/3 \cup \ldots, \\ Slot\text{-}type\text{-}validations, Actions) \quad (7)$$

Similar definitions can be provided for description logic based formalisms, topic maps, lattices, and even for non-ontology formalisms like ER diagrams or Java classes hierarchy.

3.2 Generalised Ontology Formalism

Generalised Ontology Formalism Definition The approach presented in this paper uses a metamodel language, consisting of concepts and relations expressed by arrows between them. The most important feature is the tendency to define an abstract language with only few symbols. Rich languages describing many details are restricted to these details and always become obsolete in a new situation. We decided to lower the limitations of the metamodel language in order to be able to describe a wide range of possible situations. This concept does not require extending the model language for every new supported formalism.

The idea is similar to RDF, but the relations are not concepts themselves and have no attributes. This allows a simple representation by means of a graph. RDF based formalisms are hard to draw – relations can be subclasses of other relations.

The Generalised Ontology Formalism O_{GOF} is defined as follows:

$$GOFArrowSet = \{\multimap, \blacktriangleright, \blacktriangleleft, \square, \prec, \rightarrow\} \quad (8)$$

$$GOFArrows = \{arrow(c_1, c_2) | arrow \in GOFArrowSet, c_1, c_2 \in Concepts\} \quad (9)$$

$$O_{GOF} = \{Concepts, GOFArrows, \{\}, \{\}\} \quad (10)$$

Semantics of GOF relations There exist six types of GOF relations, which are graphically expressed by arrows with six different arrow heads.

instanceOf ($\multimap$) is used for decreasing the abstractness of the concept. It corresponds to the is-a relation in frames.

subclassOf ($\blacktriangleright$) expresses the specialisation relation between a more general and a more specific concepts.

has-domain ($\blacktriangleleft$) "domain of a property," $\alpha \blacktriangleleft \mathcal{A}$ means that the class $\mathcal{A}$ is a domain of the α property

has-range ($\square$) "range of a property," $\alpha \square \mathcal{A}$ means that any instance of a class $\mathcal{A}$ is a possible value of property α.

propertyOf ($\prec$) "an assignment of a value to an instance of a property domain." The particular value(s) assigned through this assignment is defined by the $\rightarrow$ relation (defined below). More exactly:$\beta \prec \mathcal{I}_A \Rightarrow \exists \alpha, \beta \multimap \alpha \wedge \exists \mathcal{A}, \alpha \blacktriangleleft \mathcal{A} \wedge \mathcal{I}_A \multimap \mathcal{A}$. β is a property of an instance $\mathcal{I}_A$

has-value ($\rightarrow$) "a particular value of a property":
$\beta \rightarrow \mathcal{I}_B \Rightarrow \exists\alpha, \beta \multimap \alpha \wedge \exists\mathcal{B}, \alpha \multimap\!\!\Box\, \mathcal{B} \wedge \mathcal{I}_B \multimap \mathcal{B}$.

Gates A part of the whole generalised ontology formalism design is an idea of gates to particular formalisms. Every "gate" will provide two kinds of transformation of the formalism – an exact, lossless transformation with help of a formalism-specific ontology and a transformation to the pure GOF.

The formalism-specific ontology (FSO) is a formal description of the formalism, defining terms like Class, Subproperty, Restriction, etc. A transformation between the generalised ontology formalism with FSO and the original formalism has to be lossless to allow transformation of an ontology to GOF and back without loss of information. Formalism-specific ontologies for RDFS, OWL, OCML and some others have been already specified [7].

There may be a requirement to express an ontology in GOF without FSO.This may be the case of converting ontology to another formalism, where FSO does not exist (too different formalisms).

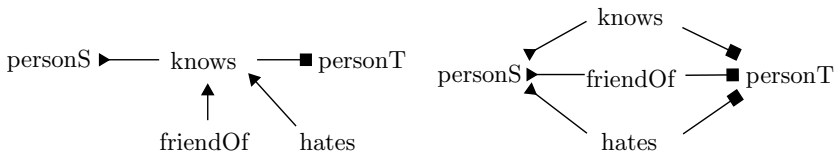
(Un)Informed Transformation There are three types of transformation between two ontologies:
Informed Transformation There exists a mapping of the source FSO to the target one. The transformation between formalisms uses FSOs and the mapping between them.
Informed Transformation with Mediators There exists a path of formalisms, where for every pair of succeeding formalisms exists a mapping of FSOs.
Uninformed Transformation If there is no way, how to use FSO, the transformation to the pure GOF is performed.

The informed transformation corresponds to the mapping approach mentioned in the section 2, while the uninformed transformation is generally similar to the pivot approach, but uses a simple language instead of the richest one.

Mapping Engine Each gate must be able to transform ontology in GOF to its specific formalism – to map GOF concepts to concepts in the FSO. To simplify the gate development, the framework provides a rule-based engine. Each rule defines, which combination of the FSO concepts can exist in the GOF. An advantage of the default engine is its ability to minimise information loss if the GOF form is invalid with respect to the set of rules. It identifies the smallest set of invalid relations, which are then ignored or fixed manually.

Example An example of a possible difference in capabilities of formalisms is the subproperty available in description logic. Frames do not provide any similar construct – it is not possible to specify a "subslot" as a subclass of a slot. Figure 1 shows an ontology with subproperty ($friendOf$ and $hates$ are subproperties of $knows$) represented in terms of the GOF. Frame-based formalisms are not able to handle this model directly. For this reason a transformation of the source model

will be carried out. Several kinds of transformation have been investigated and the most general way is shown in figure 1(b). All properties turn into direct properties of *personS*. Furthermore all instances have to be changed.



(a) Ontology with subproperty (b) Translated ontology

Fig. 1. Translation of ontology

Restriction Handling As stated above, the framework focuses on the structural part of the ontology. Therefore the GOF was not designed to cover completely the procedural part of the ontology such as axioms, slot facets in OCML and restrictions in OWL. However, in ontology languages based on description logic such as OWL and DAML+OIL, property restrictions play an important role in definition of classes. Therefore special attention was devoted to them. The example below shows the representation of an OWL property restriction in GOF. The example was taken from the Knowledge Acquisition ontology [8]. The representation in GOF using the FSO can be seen in the figure 2. The closest representation of restriction in frame-based languages is adding a slot with a defined default value to the class. In our example, it means adding a slot researchInterest with default value ResearchTopic to the class Researcher.

```
<owl:Class rdf:about="#Researcher">
  <rdfs:subClassOf><owl:Class rdf:about="#AcademicStaff"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#researchInterest"/>
      <owl:allValuesFrom><owl:Class rdf:about="#ResearchTopic"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

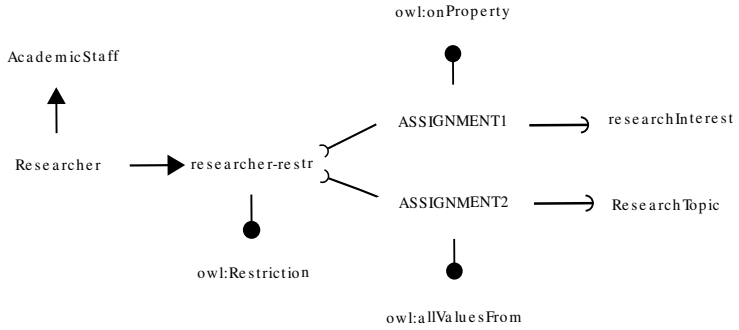


Fig. 2. OWL restriction model in GOF

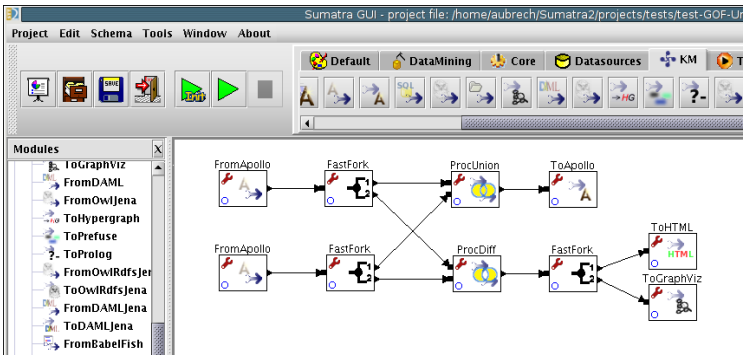


Fig. 3. Ontology processing in SumatraTT

4 SumatraTT

SumatraTT [2] is a universal data processing system, developed at the Department of Cybernetics, CTU in Prague. The primary purpose was to build a modular platform for development of modules providing specific tasks. The SumatraTT kernel is flexible enough to support problem-specific data formats. There has been developed a translation program generating SumatraTT modules automatically from GOF gates.

5 Results & Conclusion

The GOF provides a framework for research of ontology formalism conversion. This formalism is used as a dictionary in the process of information migration between different systems for both data conversion and information interchange between knowledge systems. The relative simplicity of GOF makes it suitable for

transformations to/from many formalisms and sharing ontologies from multiple sources in a uniform way. It also makes it easier to perform operations on ontologies based on either equal or different formalisms such as determining differences between ontologies, merging ontologies etc.

The the described approach has been tested on several ontologies available including SUMO, a from the DAML site and some ontologies created during the project CIPHER such as the historical architecture ontology. The tests focused mainly on transformation between OWL and Apollo in both directions. The size of ontologies varied from tens to 1434/1729 (SUMO) concepts/relations.

The structural parts of ontologies were translated successfully. There were differences between informed and uninformed transformation; in most cases due to different handling of operators with a higher arity in the OWL gate, see 3.2. Moreover the informed transformation was able to preserve literal data type, while the uninformed transformation handled it as a string.

The generalised ontology formalism is implemented as a part of a system SumatraTT, which provides a graphical interface for the transformations.

The presented research has been supported by IST RTD project Enabling Communities of Interest to Promote Heritage of European Regions "CIPHER" and the research grant No. 212300013 "Decision making and control for manufacturing" of the Czech Ministry of Education, Youth and Sports.

References

1. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N., Poli, R., eds.: Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, Kluwer Academic Publishers (1993)
2. Aubrecht, P., Mikšovský, P., Král, L.: SumatraTT: a Generic Data Pre-processing System. In: Database and Expert Systems Applications, Heidelberg, Springer (2003) pp 120–124
3. Stuckenschmidt, H.: Ontology-Based Information Sharing in Weakly Structured Environments (2002)
4. Euzenat, J., Stuckenschmidt, H.: Family of Languages' Approach to Semantic Interoperability (2001)
5. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Technical Report KSL-92-71 (Revised), Knowledge Systems Laboratory, Stanford University (1993)
6. Motta, E.: Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press (1999)
7. Aubrecht, P., Žáková, M.: Ontology Formalism Transformation. In Ježek, K., ed.: DATAKON 2004, Masaryk University in Brno (2004) 191–200
8. Horrocks, I.: Knowledge Acquisition ontology (2003) http://www.cs.man.ac.uk/-˜horrocks/OWL/Ontologies/ka.owl.

Java Library for Support of Text Mining and Retrieval

Peter Bednar, Peter Butka, and Jan Paralic

Dept. of Cybernetics and Artificial Intelligence, Technical University of Kosice,
Letna 9, 042 00 Kosice, Slovakia

{Peter.Bednar, Peter.Butka, Jan.Paralic}@tuke.sk

Abstract. This paper describes an original software system developed in Java for support of information retrieval and text mining. The system is being developed as open source with the intention to provide an easy extensible, modular framework for pre-processing, indexing and further exploration of large text collections. In paper, overall architecture of the system is provided, followed by three typical use case scenarios showing some possible applications of our system.

1 Introduction

Our research and education goals in the area of text mining and information retrieval with the emphasis of advanced knowledge technologies for the semantic web resulted in identification of the following requirements, which software system for our purposes should poses.

1. Be able to efficiently preprocess potentially large collections of text documents with flexible set of available preprocessing techniques.
2. Particular preprocessing techniques should be well adopted for various types and formats of text (e.g. plain text, HTML or XML).
3. Text collections in different languages were envisaged, e.g. English and Slovak, as very different sorts of languages require significantly different approaches in preprocessing phase.
4. Support for indexing and retrieval in these text collections (and experiments with various extended retrieval techniques).
5. Well-designed interface to knowledge structures such as ontologies, controlled vocabularies or WordNet.

The decision to design and implement a new tool, Java library for support of text mining and retrieval, was based on the detailed analysis of existing free software tools that could be used to support the abovementioned functionality requirements.

We found four different groups of tools:

- Text indexing and retrieval tools (such as e.g. Lucene 1),
- Tools for text processing (e.g. GATE 2, JavaNLP 3),
- Tools and APIs for support of the process of knowledge discovery in databases (Weka 4, KDD Package 5, JDM API 6),
- Frameworks for work with ontologies (e.g. KAON 7).

Features of all groups are summarized in the following table.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 162–169, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

Table 1. Some existing tools related to JBOWL.

Tools	Features					
	text analysis	NLP methods	vector representation	mining models	interface to ontologies	full-text search
Lucene	yes	no	no	no	no	yes
KDD Package	no	no	not optimized	yes	no	no
WEKA	no	no	not optimized	yes	no	no
JDM API	no	no	not optimized	yes	no	no
GATE	yes	yes	base support	text extraction task only	yes (text annotation)	no
JNLP	yes	yes	base support	no	no	no
KAON	yes	no	base support	no	yes	no

Each of the group covers very well one or two from the requirements stated above, but none of the tools is suitable for support of the other requirements and therefore are not very well suited for text mining and semantic retrieval.

Proposed Java library for support of text mining and retrieval provides an easy extensible and easy to learn modular framework for preprocessing and indexing of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models.

The rest of the paper is organized as follows. Next, section 2 presents the overall architecture of the designed system. Sections 3, 4 and 5 provide three different scenarios, how the system has been exploited for different purposes. Finally, Section 6 provides a brief summary of the paper as well as some sketches of our future work.

2 Architecture

JBOWL has the same architecture like standard Java Data Mining API (JSR 73 specification 6). This architecture has three base components that may be implemented as one executable or in a distributed environment.

- **application programming interface (API)** - The API is set of user-visible classes and interfaces that allows access to services provided by the *text mining engine* (TME). An application developer using JBOWL requires knowledge only of the API library, not of the other supporting components.
- **text mining engine (TME)** - A TME provides the infrastructure that offers a set of text mining services to its API clients. TME can be implemented as a local library or as a server of a client-server architecture.
- **mining object repository (MOR)** - The TME uses a mining object repository which serves to persist text mining objects.

TME manages execution of common text mining tasks, e.g. document analyzing, building a model, testing a model, applying a model on new data, computing statistics, and importing and exporting existing mining objects from and to MOR.

3 Text Categorisation

JBOWL provides set of common java classes and interfaces that enable integration of various classification methods. The design of JBOWL distinguishes classification *algorithms* (i.e. SVM, linear perceptron, etc.) and *classification models* (i.e. linear classifier, rule based classifier 8).

Note that users have many possibilities how to implement selected algorithm. For example, our implementation of SVM algorithm is simply wrapper around the SVMlib library. In this way, other packages like Weka can be integrated. Algorithms can even be implemented in different programming languages (i.e. C, C++) and integrated into JBOWL with Java Native Interface.

Following subsections describe implemented classification algorithms and some experiments that we have done in JBOWL environment.

Instance based learning - kNN

The kNN algorithm 9 is simple: given a new document, the system finds the k nearest neighbors among the training documents, and uses the categories of the neighbors to weight the category candidates.

Since JBOWL provide support for document vector model, implementation of kNN algorithm was very simple. In document analyzing task user can select various components for term frequency, inverse document frequency and normalization factor and in this way, it is possible to create document vector model with various weighting schemes (i.e. binary, tf, tf-idf weighting).

As settings for kNN algorithm user can select distance or similarity function (for convenience, the cosine value of two document vectors is used as a similarity function) and tresholding strategy used to binary category assignments.

One practical disadvantage of kNN classifier is that it requires large memory size to store all training documents. Using attribute selection and instance reduction methods can reduce these disadvantages.

For attribute selection, we have evaluated combination of three JBOWL attribute selection models, including ranking selection based on document frequency, information gain and mutual information. For instance reduction we have adopted algorithm called Decremental Reduction Optimalization Procedure 4 (DROP). More information about experiments was published in 10.

SVM

The support vector machine (linear) classifier 11 attempts to find, among all the surfaces Π_1, Π_2 ... in n-dimensional document space that separate the positive from the negative training examples, the Π_m that separates the positives from the negatives by the widest margin.

Since learning of SVM requires solving of quadratic optimization problem JBOWL implementation is based on SVMlib library 12 that is optimized for large sparse matrices.

Decision trees and decision rules

A decision tree classifier 13 is a tree in which internal nodes are labeled by attributes (words), branches departing from them are labeled by tests the weight that attribute has in the test document, and leafs are labeled by categories. Decision tree categorizes a test document by recursively testing for weights that the attribute labeling the internal nodes have in document vector, until a leaf reached.

Closely related to decision trees are rule-based algorithms, where each category is represented as a list of rules in disjunctive normal form (DNF) 13.

JBOWL implementation of decision trees learning algorithm is decomposed to attribute test evaluation function, search strategy and pruning method. In this way user can build algorithm that uses various combinations of methods for tree growing and pruning. For example user can implement algorithm, which will use information gain criterion for test evaluation as in C4.5 algorithm and cost-complexity pruning as in CART.

Algorithm for rule induction is decomposed in the similar way, so user can implement base algorithms such as FOIL, or complex algorithms like RIPPER which consists of growing and pruning phase of individual rule and global optimization of the final generated rule set.

Both implementations support learning on weighted training set, so they can be used directly as a base algorithms in boosting method.

4 Clustering of Texts

The goal of text clustering is to find some unseen categories (or clusters) in the set of analysed documents. Unlike text categorization, text clustering is the case of unsupervised learning task. JBOWL support for text clustering is based on self-organizing map architecture 14 that is suitable for high dimensional data produced in text mining. The self-organizing map method is a non-linear projection, which maps a high-dimensional data collection to an organized two-dimensional output map. One disadvantage of SOM maps is their static architecture. Related to SOM is GHSOM architecture (Growing Hierarchical SOM 15) that is designed to dispose of this problem. GHSOM grows the initial map (some rows or columns of neurons are added) and creates hierarchy of SOMs with possibly many layers.

Implementation of GHSOM integrated to JBOWL consists of clustering algorithm and visualization and evaluation method. We have implemented modified GHSOM algorithm 16, which avoids some problems with growing of the map and initialisation of the new layers. Compared to original method, modified version adds only single neuron in the growing phase that avoids appearance of un-initialised neurons when the whole row or column is added.

Currently output of the visualization and evaluation method is set of HTML pages generated for each layer. For each neuron, list of characteristic content bearing terms sorted according to their variability of occurrences in covered documents are extracted together with information about number of documents assigned to cluster 19.

The quality of the document cluster analysis depends on the text analysis task used for document pre-processing. To improve quality of clustering analysis of

documents in Slovak language, JBOWL provides NLP methods for morphological and syntactical analysis 20. Implemented morphological filter assigns tokens to grammatical categories (i.e. POS, case, number etc.) required for syntactical analysis and transform various word forms to base form (lemmatisation). Syntactical analysis can be used to recognize nominal phrases denoting one concept or proper name (i.e. "artificial intelligence", "European Union"). Note that all token filters are general and can be used for other text mining tasks.

5 Webocrat Scenario

The Webocracy[1] project addressed the problem of providing new types of communication flows and services from public institutions to citizens, and improving the access of citizens to public administration services and information. The new types of services increase the efficiency, transparency and accountability of public administration institutions and their policies toward citizens.

Within this project a *WEBOCRAT* system[2] has been designed and developed 18. *WEBOCRAT* system is a Web-based system comprising Web publishing, computer-mediated discussion, virtual communities, discussion forums, organizational memories, text data mining, and knowledge modeling. The *WEBOCRAT* system supports communication and discussion, publication of documents on the Internet, browsing and navigation, opinion polling on questions of public interest, intelligent retrieval, analytical tool, alerting services, and convenient access to information based on individual needs.

WEBOCRAT intelligent retrieval mechanism is built on top of JBOWL functionality. Document analysis, indexing, vector representation and API for text mining have been used as a basis. Three different text-mining tasks have been experimented for their possible exploitation within or for the *WEBOCRAT* system 17. *Clustering* and *association rules* mining are used to support development and maintenance of the ontology. But the most important is use of *text categorisation* support for semi-automatic annotation of newly added text resources as well as for automatic routing of users' informal submissions to particular department.

Moreover, *full-text retrieval* mechanism has also been added and tightly integrated with the concept-based retrieval. That means, if the user starts e.g. with full-text search, in addition to the results of the full-text query he/she will get also a list of relevant concepts and clicking to them, switch to concept-based retrieval is performed.

Support for design and maintenance of the ontology

Clustering does not fit the functionality of the *WEBOCRAT* system, because documents in *WEBOCRAT* system are primarily organized by their links to knowl-

[1] IST-1999-20364 Webocracy: "Web Technologies Supporting Direct Participation in Democratic Processes"

[2] http://www.webocrat.sk/

edge model so that primarily knowledge model is used for document retrieval and topic-oriented browsing. On the other hand, it could be useful to use techniques like GHSOM, because of its hierarchical structure that is tailored to the actual text data collection, *as a supporting tool within the initial phase, when the knowledge model of a local authority is being constructed*. This is true in such a case when local authority has a representative set of text documents in electronic form available for this purpose. But users must be aware of the fact, that GHSOM does not produce any ontology. It is just a hierarchical structure, where documents are organized in such a way that documents about similar topics should be topologically close to each other, and documents with different topics should be topologically far away from each other. Particular node in this hierarchical structure is labeled by (stemmed) words – terms, which occur most often in cluster of documents presented by this node. This list of terms can provide some idea about concept(s), which can be (possibly) represented in the designed knowledge model.

Association rules can be exploited e.g. for automatic improvements of the knowledge model in the following way. When we use as input attributes for association rules mining algorithm only concepts to which documents are linked that means that we are looking for frequently occurring linking patterns. These patterns can be confronted with the actual ontology. When e.g. the algorithm finds association between concepts X and Y and in the ontology no relation between concepts X and Y is presented, one can expect a missing relation between them.

Support for semi-automatic annotation of documents to concepts from ontology

In the *WEBOCRAT* system, ontology is used as a knowledge model of the domain, which is composed from concepts occurring in this domain and relationships between these concepts. Information is stored in the system in the form of mainly text documents, which are annotated by set of concepts relevant to the document content.

One strategy for document retrieval is based on concepts. User selects interesting concepts and asks for information related to them. The decision about document relevance to the user query is based on a similarity between set of query concepts and a set of concepts, which are annotated to the document.

Retrieval accuracy of concept-based model depends on the quality of documents annotation. Annotation of the new document is the classification task ***(text categorization task)*** when we need to make decision which concept (concept represents category) is relevant to the content of the document. To achieve required quality of annotation, *data mining methods can be useful to guide user at annotating new document*.

The system must propose relevant concepts for new document in real time, so important requirement to used algorithm is execution time efficiency. User can add or delete some associations between new document and concepts, and these changes can be immediately integrated into classifier. This requires ability of incremental learning. Relevance weighting of the concepts to the new document is better than simple binary decision. Concepts are ordered by weight of the relevance to the new document and user can search for additional relevant concept according to this ordering.

6 Conclusions and Future Work

In this paper a new research and educational toolkit has been described. Proposed Java library for support of text mining and retrieval provides an easy extensible and easy to learn modular framework for preprocessing and indexing of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models.

Different target user groups are expected to have interest to use this toolkit.

- Text mining researcher, who wish to develop and test new text mining methods;
- Application developers – Java programmers who wish to use text mining API for building of WEB or GUI applications;
- Component developers – that intend to implement framework extensions or integrate existing software with (part of) functionality of our framework;
- Students – Java knowledgeable students who have a basic understanding of the problems that text mining can solve.

We plan to provide our system for the research community for free and we expect the research community to use it for various experiments and extend it with new features (coordinating these efforts with our group). Currently, we are working on the following new modules – word sense disambiguation module; improved NLP methods for preprocessing of texts in Slovak language; vector representation of RDF sources with fuzzy retrieval and interface to OWL ontologies.

Acknowledgements

The work presented in the paper was supported by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the 1/1060/04 project "Document classification and annotation for the Semantic web".

References

1. Jakarta Lucene project, URL: http://jakarta.apache.org/lucene/docs/index.html, 2004
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, D.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.
3. Java NLP project: URL: http://www-nlp.stanford.edu/javanlp/, 2004
4. Witten, I. H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, San Francisco, 2000
5. Bednar, P., Paralic, J.: KDD Package. In Proc. of the Znalosti 2003 Conference (Vojtech Svatek Ed.), Ostrava, February 2003, Czech Republic, ISBN 80-248-0229-5, pp. 113 – 122
6. JSR 73: Data Mining API: URL: http://www.jcp.org/en/jsr/detail?id=73, 2004
7. Staab, S. Studer, R.: An extensible ontology software environment, In *Handbook on Ontologies*, chapter III, pp. 311-333. Springer, 2004.

8. Machová, K.: Strojové učenie. Princípy a algoritmy. ELFA s.r.o., 2002, Košice, pp. 117. ISBN 80 89066 51 8
9. Yang, Y. Chute, C. G.: An example-based mapping method for text categorization and retrieval. ACM Transaction on information systems, pp.12(3): 252-277, 1994
10. Bednar, P., Futej, T.: Reduction techniques for instance based learning. BASYS, 2004
11. Joachims, T.: Text categorization with support vector machine: Learning with many relevant features. European Conference on Machine Learning (ECML), pp. 137-142, Berlin, Springer. 1998.
12. Chang, C-C., Lin, C-J.: LIBSVM: a Library for Support Vector Machines. 2004
13. Apté, C., Damerau, F., Weiss, S.: Towards language independent automated learning of text categorization models. *Research and Development in Information Retrieval*, pp. 23-30, 1994
14. Kohonen, T.: Self-organizing maps. Springer-Verlag, Berlin, 1995
15. Dittenbach, M., Merkl, D., Rauber, A.: Using growing hierarchical self-organizing maps for document classification. In Proc. of European Symposium on Artificial Neural Networks – ESANN 2000, Bruges, April 2000, Belgium, ISBN 2-930307-00-5, pp. 7-12
16. Butka, P.: Clustering of textual documents (*in Slovak*). Master thesis. Department of Cybernetics and Artificial Intelligence, Technical University, Kosice, 2003
17. Paralič, J. – Bednár, P.: Text Mining for Documents Annotation and Ontology Support. Book chapter in: "Intelligent Systems at the Service of Mankind" (W. Elmenreich, T. Machado, I.J. Rudas eds.), Ubooks, Germany, 2003, pp. 237-248
18. Paralič, J. – Sabol, T. – Mach, M.: Knowledge Enhanced e-Government Portal. Proc. of the 4th IFIP International Working Conference on Knowledge Management in Electronic Government (KMGov 2003), Rhodes, Greece, May 2003, LNAI 2645, pp. 163 – 174
19. Rauber, A.: On the labeling of self-organizing maps. In Proc. of International Joint Conference on Neural Networks – IJCNN 1999, Washington DC, July 1999, USA
20. Blicha, M.: Spracovanie textových dokumentov v slovenskom jazyku s využitím nástrojov lingvistickej analýzy pre účely ich zhlukovania a kategorizácie. Master thesis. Department of Cybernetics and Artificial Intelligence, Technical University, Kosice, 2003

dRAP: Distribuované hledání prvořádových maximálních častých vzorů

Jan Blaťák

Laboratoř vyhledávání znalostí, Fakulta informatiky
Masarykova Univerzita v Brně, Česká republika

xblatak@fi.muni.cz

Abstrakt. Jednou ze základních úloh vyhledávání znalostí je dolování častých vzorů. V tomto článku popíšeme algoritmus dRAP-INDEPENDENT pro nezávislé distribuované hledání maximálních častých vzorů v logice prvního řádu. Na multirelační databázi chemických molekul ukážeme, že nový algoritmus nalezne více delších maximálních vzorů v kratším nebo stejném čase jako sekvenční algoritmus.

1 Úvod

Hledání častých vzorů [1, 9], tj. konjunkcí podmínek (literálů) platících alespoň pro daný počet záznamů z databáze, je jednou ze základních úloh dolování znalostí z databází. Maximálním častým vzorem pak rozumíme častý vzor, který nelze rozšířit o další literál. Časté vzory jsou používány v mnoha aplikacích, např. při hledání asociačních pravidel či pro tvorbu nových rysů. Proto je stále věnováno značné úsilí návrhu efektivních metod pro jejich vyhledávání, a to jak v datech jednoduché struktury (*atributově-hodnotová* nebo také *propoziční* data), tak v datech složitější struktury (jedná se např. o data uložena v relační databázi, která budeme dále označovat jako data *multirelační*). Největší důraz je přitom kladen na snížení výpočetní náročnosti, zejména neúměrného využívání paměti, a zrychlení výpočtu pokrytí testovaných vzorů. Ani ty nejlepší sekvenční algoritmy však nestačí pro zpracování příliš velkých dat a navíc nemohou být přímo použity pro dolování v distribuovaných databázích (lokálních záznamech poboček firem apod).

Při dolování v datech složité struktury, např. v chemických molekulách, řetězcích DNA či v textu, se pak často setkáváme s problémem, že nejsme schopni v rozumném čase zpracovat ani poměrně malá data. Nejznámější systém pro hledání častých vzorů v multirelačních datech WARMR [6], není možné efektivně použít pro hledání dlouhých, případně maximálních vzorů v hustých datech, přestože je založen na algoritmu *Apriori* [3] a využívá řadu vylepšení. Některé nedostatky WARMRu se snaží překlenout systém FARMR [10] a také systém RAP [4], který jsme navrhli a implementovali v naši předchozí práci. Zpracování velmi rozsáhlých dat však může být stále příliš náročné, protože oba uvedené systémy, mohou vyčerpat dostupnou paměť či čas vyhrazený pro výpočet.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 170–177, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

Pozornost je proto v současnosti věnována návrhu technik pro distribuované hledání častých vzorů. V tomto článku popíšeme první verzi systému RAP pro nezávislé distribuované hledání maximálních častých vzorů v logice prvního řádu. Distribuovaným systémem přitom budeme rozumět paralelní systém bez sdílených prostředků (paměti, sběrnice apod.).

2 Distribuované hledání častých vzorů

Pro distribuované hledání častých vzorů v propozičních datech již byla vyvinuta celá řada systémů (viz studie [8]). Uvedeme proto pouze krátký přehled základních metod a podrobněji popíšeme pouze algoritmus Partition [11], ze kterého jsme vycházeli při návrhu systému dRAP. Krátce se zmíníme také o systému PolyFARM [5], který je jako jediný z distribuovaných systémů schopný zpracovávat multirelační data.

Systémy lze rozdělit do několika kategorií podle toho zda *distribuují* data, prostor hypotéz, nebo obojí. Důležitou roli hraje také způsob komunikace uzlů během výpočtu. Systémy, jejichž uzly nekomunikují vůbec, většinou dosahují velmi dobrých výsledků. Je však velmi těžké rovnoměrně rozdělit práci mezi jednotlivé uzly. Častěji se proto vyvažuje zátěž během výpočtu zasíláním zpráv jednotlivých uzlům.

Při analýze sekvenčních algoritmů zjistíme, že časově nejnáročnější část výpočtu spočívá v testování kandidátních vzorů na datech, tedy výpočtu pokrytí. *Pokrytí*, nebo také *podporu* vzoru Q na databázi $\mathbf{r}$ definujeme jako počet záznamů z $\mathbf{r}$ splňujících podmínku danou vzorem Q. Dále budeme používat značení $freq(Q, \mathbf{r})$.

Problém lze řešit distribuováním dat, jako např. v algoritmu *Count Distribution* (CD) [2], který je také založen na algoritmu Apriori [3]. V n-tém kroku jsou vytvořeny kandidátní vzory délky n, které jsou rozeslány všem výpočetním uzlům. Ty spočítají pokrytí na svém vzorku dat a výsledek vrátí řídicímu uzlu, který vybere časté vzory a pokračuje ve výpočtu. Protože však každý uzel udržuje v paměti všechny kandidátní vzory, může dojít k vyčerpáni paměti.

Tento problém se snaží řešit techniky pro distribuování prostoru řešení. Každý uzel má kompletní data, avšak testuje na nich pouze část množiny kandidátních vzorů. Rovnoměrné rozdělení zamezí vyčerpání paměti a řeší tedy problém, se kterým se potýká řada sekvenčních, ale také distribuovaných algoritmů založených na horizontálním dělení dat. Vyvažování práce je však velmi náročné a často jej nelze realizovat efektivně. Zástupcem těchto algoritmů je např. *Intelligent Data Distribution* (IDD) [7].

Výhody obou uvedených přístupů kombinují tzv. hybridní systémy. Při výpočtu dochází jak k dělení dat mezi uzly, tak k paralelnímu zpracování množiny kandidátů. Tento přístup je použit např. v algoritmu *Hybrid Distribution* (HD) [7], který přidělí stejný vzorek dat skupině uzlů. Mezi nimi jsou poté distribuovány kandidátní vzory.

Algoritmus Partition. Algoritmus Partition [11] je určen pro vyhledávání všech častých vzorů v horizontálně fragmentovaných datech. Každému uzlu (i),

podílejícímu se na výpočtu, je přiřazena část databáze ($\mathbf{r}_i$). Následně je na uzlu a fragmentu databáze spuštěn některý sekvenční algoritmus pro nalezení všech častých vzorů s prahem minimálního pokrytí nastaveným na hodnotu $minfreq_i$, vypočítanou na základě počtu záznamů ($|\mathbf{r}_i|$) ve vzorku i a v celé databázi ($|\mathbf{r}|$) tak, že

$$minfreq_i = minfreq \cdot \frac{|\mathbf{r}_i|}{|\mathbf{r}|}$$

Po skončení výpočtu jednotlivých uzlů jsou shromážděny všechny nalezené lokálně časté vzory a je spočítáno jejich pokrytí na celé databázi. Základní myšlenka, na které je algoritmus postaven, spočívá v tom, že každý častý vzor musí být lokálně častý alespoň na jednom výpočetním uzlu. Výhodou algoritmu je pak zejména to, že nevyžaduje žádnou komunikaci během první fáze výpočtu.

PolyFARM. Pro distribuované dolování častých vzorů v horizontálně fragmentovaných multirelačních datech byl prvním a dosud jediným systémem PolyFARM[1] [5]. Je založen na algoritmu *Count Distribution*, jeho struktura je však modulární. Výpočet probíhá ve třech fázích, které jsou řešeny samostatnými moduly. Kandidátní vzory jsou vytvářeny modulem *Farmer*, výpočet pokrytí na vzorku dat řeší modul *Worker* a modul *Merger* slučuje informace o pokrytí od jednotlivých uzlů a následně předává nalezené vzory zpět modulu *Farmer*. Zajímavostí je pak zejména to, že celý systém je napsán ve funkcionálním programovacím jazyce Haskell, přičemž data a nalezené vzory jsou zapsány v jazyce logiky prvního řádu.

3 Distribuovaný RAP

V naší předchozí práci [4] jsme navrhli a implementovali RAP, systém pro hledání prvořádových maximálních častých vzorů v multirelačních datech. RAP využívá několik metod prohledávání, poskytuje několik heuristik pro výběr vzorů z množiny všech kandidátů a metod prořezávání stavového prostoru. Zaměřili jsme se na řešení problémů spojených s časovou náročností výpočtu pokrytí vzorů a navrhli algoritmus dRAP-INDEPENDENT pro vyhledávání maximálních vzorů v distribuovaných datech.

3.1 Algoritmus dRAP-Independent

Výpočet tohoto algoritmu sestává ze tří kroků. První krok výpočtu – rozdělení databáze – a druhý krok – vyhledání lokálně častých vzorů – přitom vyžaduje jen velmi malé úpravy oproti algoritmu Partition. K nalezení konečné množiny globálně maximálních častých vzorů, třetí krok výpočtu, však použít algoritmus Partition nelze. Každý nalezený vzor je sice častý vzhledem k celé databázi, nemusí ale nutně být maximální. Naopak vzory, které byly sice lokálně časté, ale nejsou časté globálně, je nutné nejprve zobecnit a posléze zpracovat stejně jako

[1] `http://www.aber.ac.uk/compsci/Research/bio/dss/polyfarm/`

ostatní lokálně časté maximální vzory (specializovat). Algoritmus dRAP-INDEPENDENT tedy vypadá následovně.

Algoritmus: dRAP-INDEPENDENT
Vstup: Databáze $\mathbf{r}$, práh minimálního pokrytí *minfreq* a počet výpočetních uzlů n
Výstup: množina globálně maximálních častých vzorů *GFMP*

```
1: rozděl databázi r na n částí (r_1, ..., r_n)
2: for i = 1 to n do
3:    na uzlu i spusť RAP se vstupem r_i a minfreq_i
4: end for
5: for každý nalezený lokálně častý maximální vzor Q do
6:    while freq(Q, r) ≤ minfreq do
7:       zobecni Q
8:    end while
9:    if Q ∉ GFMP then
10:      vytvoř RAPem specializaci Q_s vzoru Q
11:      přidej specializaci Q_s do GFMP
12:   end if
13: end for
```

Zbytek této podkapitoly věnujeme bližšímu popisu jednotlivých fází výpočtu. Nastíníme také možná zlepšení a další směry vývoje.

Dělení dat. Dělení multirelační databáze není triviální. Každý příklad totiž může být definován pomocí mnoha faktů z různých tabulek. Situaci pak navíc stěžuje použití doménové znalosti. Fragmentování se proto často provádí pouze na množině jednoznačných identifikátorů příkladů. Identifikátorem může být např. primární klíč, jedná-li se o RDBS. Nevýhodou je pak to, že systém načítá i data, která k výpočtu nepotřebuje.

Stávající verze algoritmu dRAP-INDEPENDENT využívá právě tento přístup. Příklady jsou jednotlivým uzlům přidělovány náhodně, přičemž je ve všech částech zachován stejný poměr tříd jako na datech původních. To však znemožňuje rozumně vyvažovat práci mezi uzly. Lze totiž vytvořit vzorek příliš *řídkých* nebo naopak *hustých* dat, tj. dat, ve kterých lze vyhledat málo resp. velmi mnoho dlouhých častých vzorů. Řešení problému s vyvažováním zátěže je jedním z cílů naší další práce.

Lokálně časté maximální vzory. K hlavním výhodám modelu nezávislého počítání patří možnost použití libovolného systému pro výpočet lokálně častých vzorů. Stávající verze algoritmu využívá této vlastnosti a druhý krok výpočtu realizuje sekvenční verzi systému RAP.

Globálně časté maximální vzory. Zpracování nalezených lokálně častých maximálních vzorů je stěžejním krokem výpočtu algoritmu dRAP-INDEPENDENT. Nejprve jsou zobecněny vzory, které nejsou lokálně časté. Poté jsou sekvenční verzí algoritmu RAP nalezeny jejich specializace. Z každého vstupního vzoru je přitom vytvořen právě jeden vzor maximální.

Celý proces je ve stávající verzi vykonáván na jednom výpočetním uzlu. Proto je zvláště druhá část, specializace vzorů, poměrně časově náročná. Cílem další práce je proto distribuovat také tuto fázi výpočtu.

3.2 Řízení výpočtu

Při nevhodném rozdělení dat se může stát, že výpočet celého systému uvázne při čekání na jediný uzel, zatímco ostatní již svůj běh ukončily. Tento jev nastává v případě kdy jeden uzel zpracovává příliš hustá data a musí generovat příliš dlouhé vzory. Přitom nalezené maximální vzory většinou nebývají globálně časté.

Algoritmus dRAP-INDEPENDENT řeší tyto problémy pomoci uživatelem definovaných omezení, jako jsou maximální počet hledaných vzorů, maximální délka vzorů, časové omezení pro vyhledávání lokálně častých maximálních vzorů a minimální počet pracujících uzlů. Pro uživatele je pak zajímavá zvláště poslední hodnota, kterou může říci, že pokud již výpočet probíhá na menším počtu uzlů, než je stanovený práh, má být celý výpočet ukončen. Experimentálně jsme ověřili, že násilné ukončení ve většině případů zamezí vytváření příliš dlouhých vzorů, které nejsou globálně časté.

4 Experimenty

Algoritmus dRAP-INDEPENDENT jsme testovali na databázi chemických sloučenin *Mutagenesis* [12]. Experimenty jsme prováděli na 2, 4, 8 a 16 výpočetních uzlech vybavených procesory AMD Athlon$^{\text{TM}}$XP 1600+ a 512 MB operační paměti. Testovali jsme také vliv dodatečných omezení, konkrétně nastavení maximálního času pro vyhledávání lokálně častých maximálních vzorů. Hodnotu jsme přitom volili jako 10, 20, ..., 100 % času spotřebovaného sekvenčním algoritmem, kterou jsme získali spuštěním sekvenčního RAPu na celých datech.

Pro měření výkonu systému jsem použili míru, která zohledňuje nejen čas výpočtu, ale také počet nalezených vzorů a je definována jako

$$s = \frac{t_s}{t_d} \cdot \frac{c_d}{c_s},$$

kde t_s je čas sekvenčního algoritmu t_d čas distribuovaného algoritmu a hodnoty c_s a c_d udávají počet maximálních vzorů nalezených sekvenčním a distribuovaným algoritmem.

Sledovali jsme zrychlení druhého kroku výpočtu (vyhledávání lokálně častých maximálních vzorů) a celého algoritmu dRAP-INDEPENDENT. Zkoumali jsme také, jaké vzory jsou distribuovaným algoritmem nalezeny, zejména jejich délku, strukturu, či chování hodnoty heuristické míry na celých a rozdělených datech.

Analýza dat Mutagenesis. Pro dolování jsme využili doménovou znalost $\mathcal{B}_4$ [12], která obsahuje informace o atomech, vazbách mezi atomy, seznam složitějších struktur (benzenové kruhy a podobně) a hodnoty $LogP$ a ϵ_{LUMO}. Práh minimální frekvence byl nastaven na hodnotu 25 % (tj. 43 molekul ze 169), prostor řešení byl procházen heuristickým prohledáváním a prořezávány byly ty vzory, které byly prefixem nějakého již nalezeného vzoru. Spojité hodnoty byly diskretizovány do tří nových podintervalů.

Sekvenční algoritmus RAP v celých datech nalezl celkem devět maximálních vzorů (3 délky 4, 5 délky 5 a 1 dlouhý 7 literálů). K tomu bylo potřeba 155

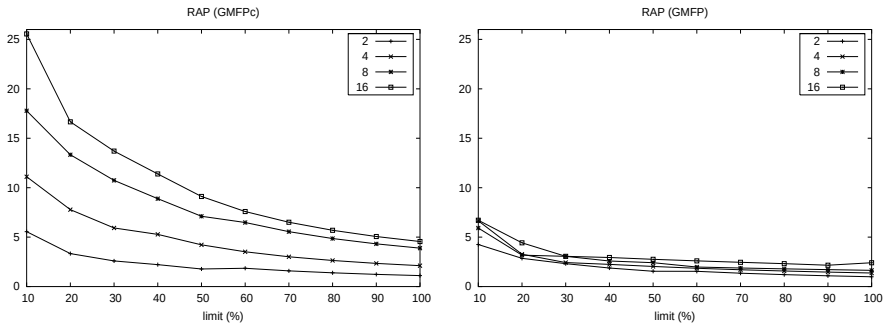
krát diskretizovat spojité hodnoty, což spotřebovalo zhruba 20 % celkového času. Celých 76 % času spotřeboval výpočet heuristiky pro výběr kandidátních vzorů,

Tabulka 1. Výsledky experimentů pro osm výpočetních uzlů.

	Předčasně	Počet vzorů				Zrychlení	
Limit	ukončeno	LFMP	GFMPc	GFFPc	GFMP	gLFMP	gGFMP
10 %	8	30	16 (13)	6	1	18	6.7
20 %	8	43	24 (20)	7	1	13	4.4
30 %	8	54	29 (25)	9	1	11	3.1
40–50 %	6	62	32 (27)	11	1	9–7	2.9–2.8
60–100 %	2	73	35 (29)	12	1	6–4	2.6–2.2

jejichž vytváření zabralo pouhá 2 % času. Provedení zbylých kroků výpočtu, načítání dat a testování zda je vzor známý, nebo není častý, zabralo přibližně 1.2 % celkového času. Výpočet trval přibližně 53 sekund.

Tabulka 1 obsahuje srovnání výsledků dosažených při provádění výpočtu na osmi uzlech. Na každém řádku je uvedeno, na kolika uzlech bylo předčasně ukončeno hledání lokálně maximálních vzorů a počet všech lokálně (LFMP) a globálně (GFMPc) častých vzorů (v závorce je uvedeno kolik vzorů bylo zároveň globálně maximálních). Zobecňováním bylo vytvořeno GFFPc nových kandidát-

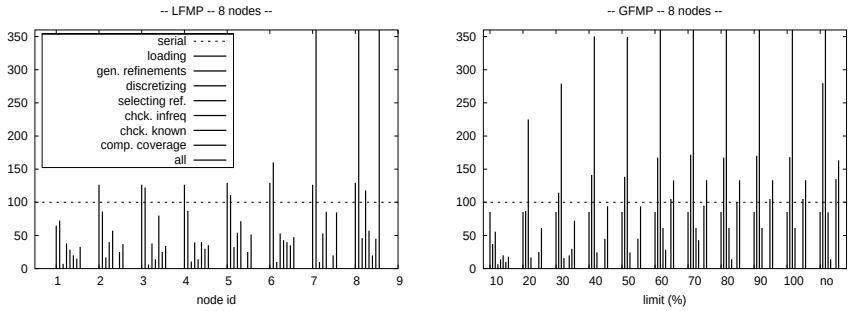


Obr. 1. Závislost míry zrychlení s na hodnotě omezení času výpočtu.

ních vzorů. Hodnota GFMP udává počet shodných vzorů nalezených sekvenčním i distribuovaným systémem. Pouze jeden společný vzor pak byl nalezen vlivem diskretizace spojitých hodnot. Míru zrychlení vytváření lokálně maximálních vzorů a celého výpočtu zobrazují hodnoty gLFMP a gGFMP.

Rozdělení dat vede k rychlejšímu nalezení většího počtu vzorů již pro 10 % limit. Nalezených GFMPc vzorů je téměř dvakrát více než při sekvenčním hledání. To odpovídá osmnáctinásobnému zrychlení první fáze výpočtu. Zrychlení klesá s rostoucím časovým limitem. To je způsobeno použitým prořezáváním a

vytvářením delších vzorů (8 literálů). Závislost gLFMP a gGFMP na přiděleném čase pro všechny experimenty zobrazují grafy na obrázku 1. Relativně malé



Obr. 2. Čas spotřebovaný jednotlivými kroky algoritmu RAP při vyhledávání lokálně častých maximálních vzorů na osmi výpočetních uzlech (levý graf) a specializaci vzorů nalezených v daném čase (vpravo).

zrychlení celého výpočtu je důsledkem sekvenční specializace nalezených vzorů. V další verzi systému bude také tento krok počítán distribuovaně.

Levý graf na obr. 2 zobrazuje relativní čas (počítáno vzhledem k hodnotám naměřeným při sekvenčním výpočtu) jednotlivých kroků při vyhledávání lokálně častých maximálních vzorů na osmi výpočetních uzlech bez omezení výpočetního času. Vidíme, že nejnáročnější kroky jsou provedeny podstatně rychleji a také to, že osmému uzlu byla přiřazena velmi hustá data. S rostoucí hodnotou přiděleného času, potažmo počtu lokálně častých vzorů, pak roste také doba specializování. Tato fáze pak trvá stejně dlouho jako sekvenční výpočet, což je však částečně způsobeno zpracováním delších vzorů.

Výsledky lze tedy shrnout tak, že pro vyšší počet výpočetních uzlů a poměrně striktní omezení času vyhledávání lokálně častých maximálních vzorů, dosahuje stávající verze algoritmu dRAP-Independent nejlepších výsledků.

5 Závěr

Prezentovali jsme dRAP-Independent, první systém pro distribuované hledání maximálních vzorů v multirelačních horizontálně fragmentovaných datech. Na testovací databázi jsme ukázali, že distribuované vyhledávání vede k rychlejšímu nalezení většího počtu delších maximálních vzorů.

V další práci se chceme zabývat odstraněním nedostatků navrženého algoritmu, zejména neefektivního sekvenčního specializování vzorů. Plánujeme do systému implementovat metody využívající komunikaci pro řízení výpočtu a metody pro efektivní dělení prostoru dat i kandidátů. Systém budeme testovat na dalších datech, zejména na datech z oblasti zpracování přirozeného jazyka či biologie.

Poděkování. Rád bych poděkoval anonymním recenzentům a L. Popelínskému za cenné připomínky. Tato práce byla částečně podporována z grantů MŠMT 143300003 a 0021622418.

Reference

1. Agrawal R., Imielinski T., and Swami A. N. Mining association rules between sets of items in large databases. *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C.,*, pp. 207–216. ACM Press, 1993.
2. Agrawal R. and Shafer J. C. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, 1996.
3. Agrawal R. and Srikant R. Fast algorithms for mining association rules in large databases. *Proc. of 20th Int. Conf. on Very Large Data Bases*, pp. 487–499. Morgan Kaufmann, 1994.
4. Blaťák J. and Popelínský L. Mining first-order maximal frequent patterns. *Neural Network World*, 5:381–390, UIVT AV ČR, 2004.
5. Clare A. and King R. D. Data mining the yeast genome in a lazy functional language. *Proc. of the 5th Int. Symposium on Practical Aspects of Declarative Languages*, pp. 19–36. Springer-Verlag, 2003.
6. Dehaspe L. and Toivonen H. Discovery of Relational Association Rules. *Relational Data Mining*, pp. 189–212. Springer-Verlag, 2001.
7. Han E.-H., Karypis G., and Kumar V. Scalable parallel data mining for association rules. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 277–288. ACM Press, 1997.
8. Joshi M. V., Han E.-H., Karypis G., and Kumar V. Efficient parallel algorithms for mining associations. *Large-Scale Parallel Data Mining*, pp. 83–126, 1999.
9. Mannila H. and Toivonen H. An algorithm for finding all interesting sentences. *Proc. of the 6th Internation Conf. on Database Theory*, pp. 215–229, 1996.
10. Nijssen S. and Kok J. N. Efficient frequent query discovery in farmer. *7th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pp. 350–362. Springer, 2003.
11. Savasere A., Omiecinski E., and Navathe S. B. An efficient algorithm for mining association rules in large databases. *The VLDB Journal*, pp. 432–444, 1995.
12. Srinivasan A., Muggleton S., King R., and Sternberg M. Theories for mutagenecity: A study of first-order and feature based induciton. Technical report, PRG-TR-8-95 Oxford University Computing Laboratory, 1995.

Annotation:

dRAP: Distributed mining first-order maximal frequent patterns

Frequent pattern discovery is one of the most important data mining tasks. In this paper we describe dRAP-INDEPENDENT, algorithm for independent distributed maximal frequent pattern mining in first-order logic. We show that the new algorithm finds more and longer maximal frequent patterns in the same time as the serial RAP algorithm in the database of chemical compounds.

Ontológia, používateľský pohľad

Karol Furdík

InterSoft, a.s., Floriánova 19, 040 01, Košice, Slovensko

Karol.Furdik@intersoft.sk

Abstrakt. Príspevok sa zaoberá vizualizáciou ontológie v systémoch využívajúcich znalostné modely. Prezentuje sa spôsob zjednodušenia vnútornej štruktúry ontológie, ktorý dovoľuje pomocou ontologických sietí štruktúrovať, archivovať, usporadúvať a vyhľadávať informácie vo vzájomnom kontexte, pritom však pre používateľa transparentne a zrozumiteľne. V príspevku sú tiež predstavené mechanizmy jazykovej analýzy textu, slúžiace ako podporné nástroje pre budovanie a správu doménového modelu, t.j. na abstrahovanie znalostí z textových dokumentov. Tento prístup je demonštrovaný na príklade konkrétneho webovského systému na publikovanie a inteligentné vyhľadávanie informácií, vyvíjaného na základe výsledkov medzinárodných projektov KnowWeb ESPRIT No. 29065 a Webocracy IST-1999-20364.

Kľúčové slová: ontológia, manažment znalostí, doménový model, jazyková analýza, konceptualizácia

1 Úvod

Používanie znalostných technológií v informačných systémoch, napriek nesporným výhodám explicitnej reprezentácie znalostí, stále nedosahuje možnosti a potenciál, ktorý tieto technológie ponúkajú. Výhodou znalostných systémov je najmä oddelenie štruktúry údajov od ich informačného obsahu, poskytovanie informácií v kontexte, presnositeľnosť a zdieľanie informácií, nezávislosť na pevnej architektúre databázy či iného spôsobu uloženia údajov. Z hľadiska prístupu k informáciám je veľkou výhodou odvodzovanie, čiže tvorba kvalitatívne nových sémantických súvislostí a ich využitie na získanie kontextovo prepojených informácií podľa ich obsahu.

Napriek tomu uplatnenie znalostných technológií v praxi zaostáva za svojimi možnosťami. Jedným z dôvodov je zložitosť a náročnosť tvorby znalostných modelov, vyžadujúca si podrobnú analýzu doménovej oblasti a zdĺhavú činnosť experta, znalostného inžiniera. Náročné sú aj procesy konceptualizácie, prepojenia doménového modelu s informačným obsahom, najmä vzhľadom na konzistenciu konceptuálnych popisov. Zmeny doménového modelu, vyžadované zväčša práve počas konceptualizácie, tiež vyžadujú zásah znalostného inžiniera. Sumarizujúc uvedené, problémom pri nasadení znalostných technológií v praxi je tvorba a administrácia doménového modelu, dôvodom je najmä zložitosť štruktúr znalostného modelu, jeho malá transparentnosť a zrozumiteľnosť pre používateľov.

V príspevku predstavíme spôsoby, ktoré by mohli čiastočne odstrániť problémy spojené s komplexnosťou znalostného modelu, ktoré by súčasne čo najviac zachovali

pôvodné výhody, najmä nezávislosť údajov od informačného obsahu a schopnosť odvodzovať nové fakty. Modelovým príkladom bude webovský informačný a publikačný systém, obsahujúci informácie vo forme textových a multimediálnych dokumentov. Na reprezentáciu sémantického kontextu dokumentov bude slúžiť formalizmus ontológie, pomocou ktorého je vytvorený znalostný doménový model.

2 Tvorba a administrácia doménového modelu

Ontológie sú vďaka flexibilite a vysokej vyjadrovacej schopnosti [1] v súčasnosti populárnym, úspešným a efektívnym nástrojom na modelovanie znalostí. Vytvárajú vhodnú platformu pre jednoznačnú sémantickú reprezentáciu pojmov a vzťahov danej oblasti (domény), umožňujú zdieľanie, znovupoužiteľnosť a modifikovateľnosť doménových znalostí. Základné jednotky ontológie, tzv. *koncepty* (resp. *pojmy*), sú abstraktné konštrukty, pomocou ktorých sa buduje model relevantnej časti sveta. Rozoznáva sa viacero typov konceptov: *triedy*, *relácie*, *funkcie*, *procedúry*, *objekty*, *premenné, konštanty*, a iné. Tvoria zväčša zložitú sieťovú štruktúru – *doménový model* danej problémovej oblasti. Proces tvorby doménového modelu sa označuje ako *pojmové*, resp. *doménové modelovanie*. Vzhľadom na rozsah a zložitosť výsledného modelu v reálnych podmienkach tento proces nie je triviálnou úlohou. Zložitosť sa dá znížiť napríklad redukciou typov konceptov a definíciou dodatočných ohraničení spresňujúcich pomerne voľnú štruktúru všeobecnej ontológie. Možnosti ohraničenia ontológie pomocou objektového prístupu sme v rokoch 1999-2001 skúmali a testovali pri návrhu, vývoji a implementácii systému *KnowWeb* – systému na modelovanie znalostí a na podporu rozhodovania v organizácii [5]. Tento systém bol výsledkom medzinárodného projektu *Web in Support of Knowledge Management in Company*, *ESPRIT Project No. 29065* (http://knowweb.fei.tuke.sk). Ohraničenia sa riadili požiadavkami priemyselných partnerov zúčastnených na projekte [4]. Testovanie v reálnych podmienkach ukázalo, že navrhnuté obmedzenia nemajú výrazný vplyv na vyjadrovaciu schopnosť ontológie, rozhodujúcou mierou však prispievajú k jednoduchšiemu a používateľsky akceptovateľnému procesu tvorby a administrácie doménových modelov [4]. Boli prijaté nasledujúce ohraničenia:

- Štruktúra ontológie bude *strom*, a nie sieť. Každý uzol (koncept) bude teda mať maximálne jeden rodičovský uzol.
- Typy konceptov: a) *Inštancie*, konkrétne objekty, b) *Triedy*, abstraktné pojmy. Inštancia musí mať práve jednu rodičovskú triedu, zároveň nesmie byť rodičom iného uzla.
- Typy relácií: a) orientovaná relácia „*Subclass_of*“ medzi dvoma triedami, b) orientovaná relácia „*Instance_of*“ medzi triedou a inštanciou.
- Uzly (koncepty) môžu byť ohodnotené jedným alebo viacerými *atribútmi* – údajovými štruktúrami odvodenými od existujúcich tried a inštancií a dedenými v hierarchii ontológie.

Tieto „striktne objektové” pravidlá boli doplnené o nasledujúce:

- Existuje päť elementárnych tried. Sú to {*reťazec, číslo, dátum, čas*, *mena*}.
- Atribút, ktorého typ je elementárny, má hodnotu zodpovedajúcu tomuto elementárnemu typu.

Systém KnowWeb bol pokusne nasadený a testovaný v prostredí komerčných organizácií a univerzít na Slovensku, vo Fínsku a vo Veľkej Británii. Z výsledkov testov je zrejmé, že zvolený prístup je sľubný a riešenie je prínosom pre riadenie a monitorovanie informačných tokov v zúčastnených organizáciách [6].

3 Používateľské rozhranie a konceptualizácia

Použitie zjednodušeného modelu ontológie vo webovskom publikačnom systéme ukázalo nevyhnutnosť ďalších modifikácií, a to najmä smerom k prehľadnému a pre používateľa transparentnému riešeniu administrátorského a publikačného rozhrania. Systém, ktorý sme v úvode definovali ako modelový príklad tohoto príspevku, totiž kladie ďalšie požiadavky na možnosti a spôsob modifikácie doménového modelu: jednoduchosť a zrozumiteľnosť rozhrania, transparentnosť administrátorských zásahov, transformácia zmien v znalostnom modeli do webovského rozhrania, a iné. Tieto požiadavky postupne vyplynuli z použitia zjednodušeného modelu ontológie v systéme *Webocrat*, vyvíjanom v rokoch 2000-2003 v rámci projektu *Webocracy* IST-1999-20364 (*Web in Supporting Participation in Democratic Processes*). Tento systém, primárne určený pre oblasť elektronickej verejnej správy (*eGovernment*), poskytuje občanom cez WWW rozhranie integrované dynamické služby – diskusie, hlasovania a prieskumy verejnej mienky, inteligentný prístup k dôležitým dokumentom, podporu verejného obstarávania, portál na odosielanie rôznych podaní, a podobne. Integrujúcim prvkom pre tieto služby je ontologický znalostný model úradu verejnej správy [8]. Tento model tvorí reprezentáciu sémantického kontextu publikovaných dokumentov a objektov, čím vlastne vytvára štruktúru WWW stránky – definuje štruktúru menu a sieť hypertextových liniek spájajúcich publikované dokumenty. Podieľa sa aj na procese inteligentného získavania informácií, kde sa fulltextové vyhľadávanie kombinuje s konceptuálnym vyhľadávaním.

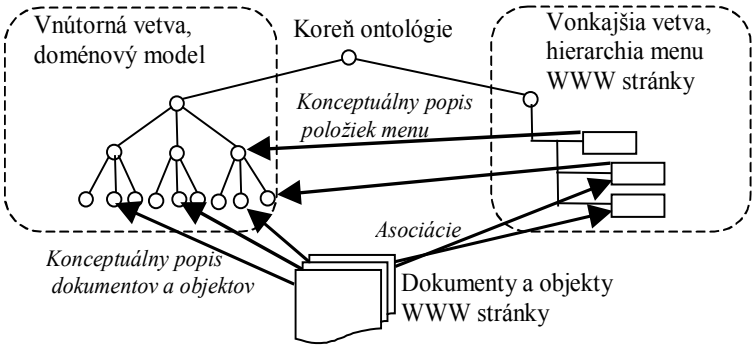
Informácie publikujú zamestnanci organizácie v pridelenej časti web priestoru pomocou špecializovaného rozhrania so zabezpečeným prístupom podľa používateľských skupín. Publikovať dokument pritom znamená nielen napísať text, ale aj zaradiť ho do štruktúry stránky, čiže nalinkovať ho na pojmy modelu. Pri tomto procese je často potrebné modifikovať samotný model – zmeniť či vytvoriť nové koncepty, atribúty a relácie. Publikujúci zamestnanci však nie sú znalostní inžinieri, preto spôsob konceptualizácie a úpravy doménového modelu musí zodpovedať ich schopnostiam a potrebám. Rozhodne nie je možné použiť štandardné nástroje na tvorbu ontológií, nevyhovujúce prílišnou zložitosťou. Publikačný nástroj nemá vyžadovať špeciálne znalosti, konceptualizácia má byť jednoduchá, transparentná a podľa možnosti aj čiastočne automatizovaná, aby zmeny doménového modelu bolo možné vykonávať intuitívne a aby sa tieto zmeny hneď prejavili na štruktúre stránky.

Z týchto praktických požiadaviek vyplýva potreba ďalšej modifikácie štruktúry ontológie, a to najmä vzhľadom na skutočnosť, že doménový model (respektíve jeho časť) definuje vlastnú povrchovú štruktúru a organizáciu WWW stránky. Naviac sa ukázalo nevyhnutné upraviť ontológiu tak, že sa nerozlišujú typy konceptov, teda že triedy a inštancie v konečnom dôsledku splývajú do jedného typu abstraktného konceptu (ktorý má v skutočnosti vlastnosti triedy).

K ohraničeniam z predchádzajúcej časti pribudli tieto spresnenia:

- Bude existovať uzol (koncept) iba jedného typu, a to typu *Trieda*.
- *Koreňový uzol* celej ontológie je abstraktný, nepoužíva sa na konceptuálny popis, slúži iba na prepojenie tzv. vnútornej a tzv. vonkajšej vetvy ontológie.
- Prvú úroveň tvoria práve dva uzly (koncepty): a) koreň vnútornej vetvy, b) koreň vonkajšej vetvy ontológie.
- *Vonkajšia vetva ontológie* je objektovou a údajovou reprezentáciou stromovej štruktúry WWW stránky. Koncepty vonkajšej vetvy majú povinné atribúty {*názov*, *popis*, *text*}, zodpovedajúce priamo vizualizácii týchto konceptov ako položiek menu vo WWW stránke.
- *Vnútorná vetva ontológie* tvorí samotnú znalostnú bázu systému. Slúži na tvorbu konceptuálneho popisu asociovaných dokumentov, avšak zároveň aj na konceptuálny popis uzlov vonkajšej vrstvy.

Dokumenty a dynamické objekty WWW stránky nie sú súčasťou znalostného modelu, ich obsah je však sémanticky ohodnotený konceptuálnym popisom pojmov vnútornej aj vonkajšej vetvy. Symetrické väzby (linky) dokumentov s konceptami vonkajšej vetvy, tzv. *asociácie*, sa premietajú priamo do štruktúry stránky. Majú preto vyššiu prioritu ako väzby s konceptami vnútornej vetvy. Celá štruktúra vnútornej organizácie ontológie, dokumentov a objektov v systéme je zobrazená na obrázku 1.



Obr. 1. Štruktúra ontológie a asociačných vzťahov.

Z používateľského hľadiska kompetentní zamestnanci vykonávajú pomocou špecializovaného administrátorského nástroja nasledujúce akcie:

Tvoria a modifikujú štruktúru menu WWW stránky. Každá položka menu má názov, stručný popis a samotný HTML text. Jej reprezentáciou je uzol vo vonkajšej vetve ontológie, avšak pre používateľov je proces vytvárania a zmeny týchto uzlov transparentný a okamžite sa premieta do grafickej, vizuálnej formy WWW stránky.

Vkladajú do systému dokumenty a dynamické objekty a asociujú ich s položkami menu. Systém dovoľuje publikovať súbory rôznych formátov, diskusie, hlasovania, novinky, časovo ohraničené udalosti, textové HTML dokumenty a ďalšie špecializované objekty. Jeden objekt alebo dokument môže byť asociovaný s viacerými (prípadne so žiadnym, alebo aj so všetkými) položkami menu. Administrátorský nástroj ponúkne celý strom menu, v ktorom používateľ označí tie položky, s ktorými má byť daný dokument asociovaný.

Konceptuálne popisujú položky menu, dokumenty a objekty, prípadne aj dopĺňajú doménový model. V tomto prípade existujú tri možné prístupy, odstupňované v závislosti od miery zrozumiteľnosti vnútornej vetvy doménového modelu pre zamestnancov – používateľov:

1. *Manuálna konceptualizácia*, predpokladá dobrú organizáciu doménového modelu a jeho zrozumiteľnosť pre používateľov. Vnútorná vetva ontológie sa zobrazuje ako hierarchia pojmov. Výberom niektorých z týchto pojmov používateľ vytvorí konceptuálny popis pre položku menu, pre publikovaný objekt alebo dokument.
2. *Čiastočne automatizovaná konceptualizácia*. Systém vykoná *jazykovú analýzu* textu položky menu, objektu alebo dokumentu. Na základe analýzy systém navrhne zoznam konceptov, ktoré by mali tvoriť konceptuálny popis. Používateľ môže návrh akceptovať alebo výber manuálnym spôsobom upraviť.
3. *Automatická konceptualizácia*. Celá vnútorná vetva ontológie je pre používateľov skrytá, konceptuálne popisy sa tvoria a modifikujú automaticky, s použitím mechanizmov jazykovej analýzy textu.

Výsledkom je dynamická WWW stránka, ktorej hierarchické menu zodpovedá vonkajšej vetve ontológie. Po kliknutí na položku menu sa zobrazí jej text spolu so zoznamom asociovaných dokumentov a objektov. Súčasne sa na základe konceptuálneho popisu položky menu vyhľadá a zobrazí zoznam obsahovo súvisiacich dokumentov (stupeň obsahovej blízkosti je vyjadrený číselnou váhou).

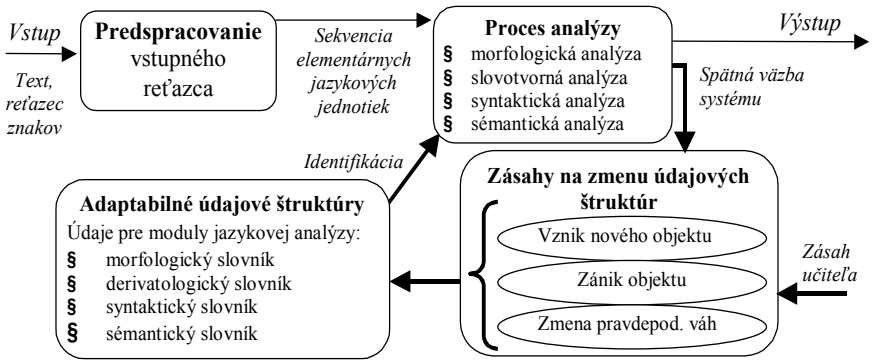
Konceptuálny popis sa tiež používa pri vyhľadávaní, v kombinácii s fulltextovým indexom. Primárne sa vykoná fulltextové vyhľadávanie, následne sa k získaným dokumentom na základe konceptuálnych popisov vyhľadajú obsahovo súvisiace dokumenty. Výsledkom je štruktúra primárne zoradená podľa fulltextového indexu, avšak sémanticky rozšírená o obsahovo blízke dokumenty.

4 Jazyková analýza. Aktualizácia doménového modelu

Úlohou jazykovej analýzy je vhodným formalizmom vyjadriť sémantický obsah textu. Keďže v popisovanom systéme sú nositeľmi sémantickej informácie iba koncepty znalostného modelu, jazyková analýza vykonáva transformáciu písaného textu do podmnožiny konceptov vnútornej vetvy ontológie. Táto podmnožina následne tvorí konceptuálny popis dokumentu, ktorému prislúcha analyzovaný text.

Text je tvorený textovými jednotkami ako sú morfémy, slová, frazémy, vety a vetné celky, je nositeľom významu a má teda istý obsah. Analýza textu spočíva v identifikácii lingvistických jednotiek a ich vzájomných vzťahov. Text sa člení na obyčajne hierarchicky organizované, súvislé a obsahovo ucelené časti, lingvistické jednotky, z ktorých sa následne abstrahujú ich príslušné konceptuálne popisy. Tieto sa porovnávajú s vnútornou vetvou doménového modelu. Ak sa v nej identifikované koncepty nachádzajú, potom sa priradia príslušnej časti textu ako jej popis. Môže sa však stať, že analýza identifikuje nové, v doménovom modeli doteraz nezaradené koncepty. Vtedy je potrebné najprv aktualizovať doménový model, doplniť nové koncepty, a až následne nimi reprezentovať obsah danej časti textu. Týmto spôsobom môže jazyková analýza buď automaticky, alebo v spolupráci so administrátorom – znalostným inžinierom, aktualizovať doménový model [3].

Jazykovú analýzu možno riešiť buď štatistickými metódami, alebo kombináciou lingvistických a pravdepodobnostných metód. Pre flektívny jazyk, akým je slovenčina, je výhodnejší a presnejší druhý spôsob, t.j. využitie pravdepodobnostných metód na identifikáciu neznámych jazykových jednotiek v texte a ich následné zaradenie do lingvistických kategórií a vzťahov. Na rozdiel od čisto lingvistického prístupu je tu splnená požiadavka otvorenosti a adaptability – systém by totiž nemal byť ohraničený akýmkoľvek pevne definovaným slovníkom, mal by byť schopný v texte rozoznať aj nové, doteraz v systéme nezaradené jazykové jednotky [3]. Je to dôležité najmä pri informačných a publikačných systémoch pracujúcich s reálnymi textami, kde napríklad vlastné mená, skratky a ustálené slovné spojenia nie je možné vyčerpávajúco v slovníku zachytiť. Pritom sú práve tieto často hlavnými nositeľmi sémantickej informácie. Je preto vhodné, aby sa modul jazykovej analýzy koncipoval ako *učiaci sa systém*, schopný na základe trénovacej množiny dokumentov, resp. iných apriórnych vedomostí, indexovať aj texty obsahujúce nové jazykové jednotky – slová, frázy a syntaktické konštrukcie. Zodpovedá to tiež kognitívnej funkcii jazyka, ktorú možno považovať v procese abstrahovania obsahu z textov za kľúčovú [7].



Obr. 2. Schéma jazykovej analýzy ako učiaceho sa systému.

Na obrázku 2 je schéma jazykovej analýzy, ktorej údajové štruktúry sa dopĺňajú o nové poznatky získané buď spätnou väzbou, alebo zásahmi učiteľa, administrátora. Moduly jazykovej analýzy pracujú v paralelnom režime, vzájomne si poskytujú informácie o medzivýsledkoch a súčinnosťou eliminujú nejednoznačnosti.

Systém jazykovej analýzy pozostáva z týchto piatich modulov:

1. *Predspracovanie vstupu*. Bezkontextovou gramatikou sa text člení na postupnosť tzv. *elementárnych jazykových jednotiek* – najmenších, ďalej nedeliteľných segmentov textu, ako sú čísla, reťazce alfabetických znakov, interpunkcia, a iné.
2. *Morfologická analýza* skúma jazykové javy na úrovni slov. Spája elementárne jazykové jednotky do slov a určuje ich morfologické charakteristiky. Sú to najmä základný tvar slova (lemma), slovný druh a príslušné gramatické kategórie. Práve tu možno efektívne využiť kombináciu lingvistických a pravdepodobnostných metód. Paralelným využitím dynamického morfologického slovníka, heuristickej bezkontextovej gramatiky na separáciu koreňov a pravdepodobnostného modelu tvarovej podobnosti [7] sa výrazne zníži neurčitosť, skvalitní sa morfologická identifikácia a zachová sa otvorenosť voči neznámym slovám.

3. *Slovotvorná analýza* pomáha určiť sémantický kontext slovotvorne motivovaných slov a identifikovať slovotvorné hniezda – skupiny v určitom zmysle sémanticky blízkych slov. Využíva sa slovník pozostávajúci zo zoznamu typov slovotvornej motivácie, tzv. *onomaziologických kategórií* [2]. Vstupom je lemma, určená počas morfologickej analýzy. V lemme sa hľadajú slovotvorné predpony a prípony, ktoré zodpovedajú slovotvorným formantom uloženým v slovotvornom slovníku, a ktoré sú zároveň nositeľmi parciálnych významov. Slovotvorná a morfolgická analýza najviac prispievajú k identifikácii potenciálne nových konceptov, pomocou ktorých sa následne aktualizuje doménový model.
4. *Syntaktická analýza* skúma jazykové javy na úrovni viet a závislostné vzťahy medzi slovami. Používa sa mechanizmus rozšírených prechodových sietí (ATN), pomocou ktorého sa formálne definujú vetné syntaktické modely [3]. Vetné modely koncepčne vychádzajú z *teórie slovesnej valencie*, čiže z centrálneho postavenia slovesa vo vete, vyžadujúceho vo svojom bezprostrednom okolí vetné participanty [9]. Sloveso so zoznamom participantov sa nazýva *valenčný rámec* a formalizované do tvaru ATN siete slúži na identifikáciu viet a vetných členov, na jednoznačné určenie morfologických dvojtvarov a na sémantické ohodnotenie slov a frazém podľa ich postavenia vo vete.
5. *Sémantická analýza*. Sémantický slovník zaznamenáva významové a sémantické charakteristiky určené v predchádzajúcich moduloch, môže tiež obsahovať explicitné sémantické pravidlá. Tento slovník slúži ako zdroj údajov pre aktualizáciu doménového modelu.

Moduly jazykovej analýzy boli (v rámci doktorandskej dizertačnej práce autora, [3]) implementované do funkčného celku a systém bol ako prototyp testovaný na kolekcii textov v publicistickom štýle, najmä novinových článkov obsahujúcich politické, ekonomické, športové a kultúrne informácie. Testovaciu kolekciu tvorilo 280 dokumentov, jeden text mal priemernú dĺžku 154 slov. Výsledky testov potvrdili, že hodnoty presnosti a úplnosti systému využívajúceho jazykovú analýzu v spojení s konceptuálnou reprezentáciou sú v priemere o 26 % lepšie ako pri štatistických prístupoch založených na metóde kľúčových slov a na fulltextovom vyhľadávaní [3].

5 Záver

Prezentovaný spôsob zjednodušenia vnútornej štruktúry ontológie, jej prepojenia s logickou štruktúrou WWW stránky a podporné moduly jazykovej analýzy tvoria jednu z možných ciest integrácie znalostných technológií do webovských informačných a publikačných systémov. Svedčia o tom jednak výsledky testovania prototypov jednotlivých súčastí, ako aj implementácia a nasadenie systému ako celku v reálnom prostredí viacerých organizácií.

Systém Webocrat s pôvodnou štruktúrou ontológie podľa popisu uvedeného v časti 2 je v prevádzke na úradoch mestských častí Košice – Dargovských hrdinov (http://www.kosice-dh.sk) a Košice – Ťahanovce (http://www.tahanovce.sk/mutah), a tiež pre úrad mesta Wolverhampton vo Veľkej Británii (http://www.wolforum.org). Ďalší vývoj, spočívajúci najmä v modifikácii znalostného modelu a algoritmov jazykovej analýzy podľa zásad z častí 3 a 4, vyústil do implementácie webovského

publikačného systému, ktorý je v súčasnosti v testovacej prevádzke vo viacerých organizáciách samosprávy, verejnej správy, školstva a komerčnej sféry. Predbežné výsledky a odozvy od používateľov nasvedčujú, že zvolený prístup je sľubný a má perspektívu ako spôsob integrácie znalostných systémov do praktických aplikácií webovských publikačných a informačných systémov.

Referencie

1. Chandrasekaran B., Josephson J.R., Benjamins V.R. What Are Ontologies and Why Do We Need Them. *IEEE Intelligent Systems 14/1999.* ISBN 3-7908-1257-9.
2. Furdík J. *Slovenská slovotvorba*. Náuka, Prešov 2004. ISBN 80-89038-28-X.
3. Furdík K. *Získavanie informácií v prirodzenom jazyku s použitím hypertextových štruktúr. Doktorandská dizertačná práca*. Katedra kybernetiky a umelej inteligencie FEI, Technická univerzita v Košiciach, Košice 2003.
4. Furdík K., Džbor M. *Web-OrgMem Module's Specification. ID2.1. Project KnowWeb, Technical Report.* IFBL Slovakia s.r.o. / Technická univerzita v Košiciach, Košice 1999.
5. Kende R. et.al. An Information System for support of Knowledge Management in Compan*y*. *Proc. of the Jubilee Int. Conference*, Hungary, Budapest: Banki Donat, 1999.
6. Kováč J., Furdík K. *IFBL Pilot Application – Electronic Commerce. D7.3, Project KnowWeb, Technical Report*. IFBL Slovakia s.r.o., Košice 2001.
7. Kostelník P., Furdík K. Pravdepodobnostný model tvarovej podobnosti pre flektívne jazyky. VARIA X, zborník konferencie *Kolokvium mladých jazykovedcov 2000*. Bratislava 2003. ISBN 80-89037-04-6.
8. Mach M., Furdík K. *Webocrat system architecture and functionality. R2.4, Project Webocracy, Technical Report.* Technická univerzita v Košiciach / Juvier s.r.o., Košice 2001.
9. Nižníková J. *Vetné modely v slovenčine*. Filozofická fakulta Prešovskej univerzity, Prešov 2001. ISBN 80-8068-052-3.

Annotation:

Ontology, user-driven approach

This paper treats of the visualisation of ontologies in knowledge-based systems. Emphasis is given on the process of such administration tasks as domain modelling and conceptualisation from the perspective of users. A method of simplification of the inner ontology structure is described, which enables to manipulate with the conceptual structure of information in a transparent and understandable way for the users. A module of linguistic analysis as a supporting tool for creation and modification of ontology-based domain model is also presented. This approach is demonstrated on the example of real web-based publishing and information retrieval system, designed and implemented using the results of two international projects: KnowWeb ESPRIT No. 29065 and Webocracy IST-1999-20364.

Vyhledávání souvislých komponent a redukce grafu Webu

Petr Gajdoš, Jan Kožusznik, Eliška Ochodková, Václav Snášel

Fakulta elektrotechniky a informatiky
VŠB Technická universita Ostrava
tr. 17. listopadu 15, 708 33 OstravaPoruba
Ceská Republika

{Petr.Gajdos, Jan.Kozusnik}@vsb.cz
{Eliska.Ochodkova, Vaclav.Snasel}@vsb.cz

Abstrakt. Současná velikost Webu s sebou přináší řadu problémů týkajících se především vyhledávání konkrétních stránek. V tomto článku prezentujeme metodu umožňující pomoc s navigací při vyhledávání. Tato metoda je založena na hledání 2-souvislých komponent. Budou-li internetové stránky představovat vrcholy grafu a jednotlivé webové odkazy hrany tohoto grafu, můžeme využít několik známých metod z teorie grafů k nalezení komponent, které budou představovat shluky souvisejících stránek.

Klíčová slova: Web, graf, 2-souvislá komponenta.

1 Úvod

Studium různých typů sítí (informačních, sociálních, biologických) bylo a je stále velmi populární. Inspirována empirickými studiemi byla vyvinuta celá řada různých modelů a postupů, které nám umožňují lépe porozumět vlastnostem těchto sítí. Nejdůležitějším reprezentantem informačních sítí je pro nás pochopitelně World Wide Web. Na model Webu se můžeme dívat jako na obrovský graf, kde vrcholy grafu představují webové stránky a linky mezi nimi jsou chápány jako hrany tohoto grafu.

Současná velikost Webu s sebou přináší řadu problémů týkajících se především vyhledávání konkrétních stránek. V tomto článku se zabýváme možností usnadnit navigaci při vyhledávání prostřednictvím hledání více souvislých komponent (konkrétně 2-souvislých komponent), které neformálně řečeno představují jistá místa s častějšími vazbami. Lze předpokládat, že takto propojené stránky mají do určité míry podobný obsah, viz [2]. To znamená, že tato metoda může doplnit vyhledávací metody tím způsobem, že poskytne uživateli seznam dalších stránek souvisejících se stránkou vyhledanou.

Vlastnosti grafu Webu byly a jsou intenzivně zkoumány. Jedním z nejznámějších experimentů, který byl později matematicky upřesněn v [11], byla demon-

strace tzv. „small-world“ efektu - faktu, že mnoho dvojic vrcholů v různých sítích je spojeno relativně „krátkou“ cestou. Například v [12] je uvedena průměrná délka 16 orientované cesty mezi dvěma vrcholy a průměrná délka neorientované cesty 6 (pokud tyto existují). Dále je v téže publikaci uveden průměr 28 grafu Webu. Tato hodnota však platí pro vrcholy ležící v jakémsi jeho „jádru“ - silně souvislé komponentě. Průměr celého grafu Webu je více než 500.

Pro zkoumání různých vlastností grafu Webu je velmi zajímavá distribuce stupňů (zejména vstupních stupňů v případě orientovaného grafu) vrcholů grafu. Je zajímavé, že Web splňuje tzv. „power-law“, tj. platí, že pravděpodobnost, že vrchol má stupeň i, je proporcionální

$$1/i^x \tag{1}$$

pro nějaké $x > 1$. V [12] je uvedeno, že při pokusech pro zde uvedený graf Webu, bylo zjištěno, že část Webu s i vstupními linky (zajímá nás tedy vstupní stupeň vrcholu) je proporcionální $1/i^{2.1}$. Při hledání souvislých komponent v experimentálním grafu Webu v [12] byla nalezena jedna velká souvislá komponenta (neorientovaná) s téměř 91% všech vrcholů a bylo zjištěno, že exponent x je 2.5.

Protože pracujeme s grafem Webu, můžeme využít metody známé z teorie grafů k nalezení komponent, které budou představovat shluky tematicky souvisejících stránek. Základní myšlenkou je převést graf Webu na jednodušší „redukovaný“ graf, strom, a tak podstatně zpřehlednit a urychlit práci s Webem. Vycházíme z předpokladu, že přehlednější a jednodušší struktura Webu umožní efektivnější vyhledávání a práci s internetovými stránkami. Výše uvedené experimenty s grafem Webu zkoumaly pouze 1-komponenty nebo silně souvislé komponenty. Nás zajímá existence více souvislých komponent v grafu Webu, konkrétně 2-souvislých a 3-souvislých komponent, pro jejichž hledání existují algoritmické postupy. V prvním kroku redukce je proto použit algoritmus pro vyhledávání 2-souvislých komponent. Nalezené 2-komponenty lze dále zkoumat a hledat v nich další „více souvislé“ komponenty. Ty pak budou reprezentovat obsahově bližší stránky.

První kapitola obsahuje definice potřebných pojmů a jsou zde uvedena důležitá tvrzení charakterizující souvislost grafu. V další kapitole je zmíněn postup redukce grafu Webu na strom, který podložíme experimenty na kolekci WebTrec. Na závěr ukážeme možné směry dalšího výzkumu.

2 Důležité pojmy z teorie grafů

V této kapitole budou shrnuty nebo zavedeny potřebné pojmy, viz [8]. Předpokládejme, že pracujeme s neorientovaným grafem. Pokud budeme hovořit o orientovaném grafu, bude to explicitně uvedeno.

Definice 1. *Počet hran, se kterými je daný vrchol v incidentní, se nazývá* stupeň vrcholu *v a označuje se d_v.*

Definice 2. *Jsou-li u a v vrcholy grafu G, říkáme, že vrchol v je* dosažitelný *z vrcholu u, jestliže v grafu G existuje cesta z vrcholu u do vrcholu v.*

Definice 3. Průměr *diam(G) grafu G je maximální ze všech délek nejkratších cest mezi libovolnými dvěma vrcholy grafu G.*

Definice 4. *Graf, ve kterém jsou z libovolného vrcholu dosažitelné všechny ostatní vrcholy, se nazývá* souvislý. *Jinak se nazývá* nesouvislý. Silně souvislý graf *je takový graf, ve kterém pro každou dvojici vrcholů x, y existuje orientovaná cesta z x do y a také zpět z y do x.*

Definice 5. Komponenta *H grafu G je maximální souvislý podgraf H grafu G. Maximální silně souvislý podgraf daného grafu G se nazývá* silně souvislá komponenta *grafu G.*

Definice 6. *Vrchol x grafu G se nazývá* artikulace *grafu G, jestliže existují hrany $\{x, y_1\}$ a $\{x, y_2\}$, které nepatří současně téže kružnici grafu G.*

Definice 7. *Množinu vrcholů, jejichž vynecháním se graf stane nesouvislým nebo triviálním, nazýváme* vrcholovým řezem *nebo jen* řezem *grafu G.*

Definice 8. Vrcholová souvislost *nebo jen* souvislost $\kappa(G)$ *grafu G je minimální počet vrcholů, jejichž vynecháním dostaneme buď nesouvislý graf nebo izolovaný vrchol.*

Definice 9. *Graf G nazýváme* vrcholově k-souvislým *nebo jen* k-souvislým, *jestliže je $\kappa(G) \geq k, k \geq 1$.*

Analogické pojmy lze definovat také pro hranovou souvislost (most, hranový řez, hranová souvislost, hranová k-souvislost $\kappa'(G)$).

Definice 10. *Netriviální souvislý graf bez artikulací se nazývá* neseparabilní *(nerozložitelný) graf.*

Definice 11. Blok *grafu G je maximální neseparabilní podgraf grafu G.*

Z uvedených definic vyplývá, že graf G je 1-souvislý jestliže G je netriviální a souvislý, dále graf G 2-souvislý jestliže G je neseparabilní graf řádu 3 a více. Obecně graf G je k-souvislý jestliže odstraněním méně než k vrcholů nevznikne ani nesouvislý ani triviální graf.

Věta 1. *Pro každý graf G platí*

$$\kappa(G) \leq \kappa'(G) \leq \delta(G), \tag{2}$$

kde $\delta(G)$ kde je minimální stupeň grafu G, tj. $\delta(G) = min\{d_x | x \in U(G)\}$.
Důkaz. Viz [8]. □

Čím vyšší jsou stupně vrcholů grafu tím je pravděpodobnější, že graf má „vysokou“ souvislost.

Věta 2. *Nechť G je graf řádu ≥ 2 a nechť k je celé číslo takové, že $1 \leq k \leq n-1$. Jestliže*

$$d_x \geq \lceil (n + k\ -2)/2 \rceil \tag{3}$$

pro každý vrchol $x \in G$, potom graf G je k-souvislý.
Důkaz. Viz [8]. □

Definice 12. *Nechť G je graf. Pak* k-souvislou komponentou *grafu G nazveme maximální k-souvislý podgraf H grafu G.*

Speciálně nás zajímají 2-souvislé komponenty grafu G, což jsou maximální 2-souvislé podgrafy grafu G, viz příklad 1.

Definice 13. *Buď G souvislý graf, $b_1, ..., b_r$ všechny jeho bloky, $a_1, ..., a_s$ všechny jeho artikulace. Graf $B(G)$, definovaný předpisem*

$$U(B(G)) = a_1, ..., a_s, b_1, ..., b_r, \tag{4}$$

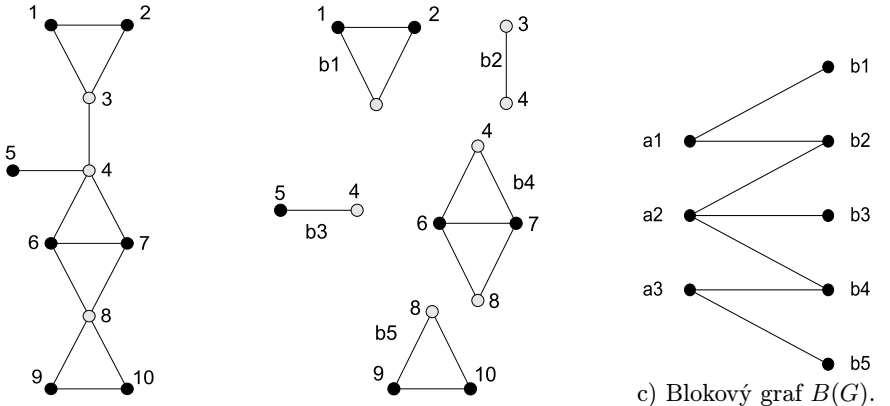
$$H(B(G)) = \{\{a, b\} | \exists i, j : a\ = a_i, b = b_j, a_i \in U(b_j)\}, \tag{5}$$

se nazývá blokový graf *grafu G.*

Zřejmá, leč důležitá vlastnost blokového grafu je popsána v následujícím tvrzení.

Věta 3. *Pro každý souvislý graf G je blokový graf $B(G)$ stromem.*
Důkaz. Viz [8]. □

Proto dvě k-komponenty grafu G které sdílejí vrcholový řez s nejvýše k vrcholy jsou v grafu $B(G)$ spojeny cestou délky 2.



a) Graf G je 1-souvislý

b) Graf G rozdělený na 2-souvislé komponenty

c) Blokový graf $B(G)$.

Obr. 1. Příklad rozdělení grafu na souvislé komponenty pomocí 1-vrcholových řezů.

3 Hledání 2-souvislých komponent

Pro nalezení 2-souvislých komponent se používá algoritmus *prohledávání do hloubky*, dále jen *DFS* podle anglického názvu Depht-First Search. Tento algoritmus je s drobnými úpravami (viz [13]) aplikovatelný jak na neorientovaný, tak na orientovaný graf.

Modifikace algoritmu. Vzhledem k velikosti grafu a velké složitosti DFS-algoritmu hledáme jiné rychlejší k nalezení 2-souvislých komponent. S využitím věty 2 a vzhledem ke statistickým vlastnostem grafu je možno modifikovat algoritmus pro hledání 2-souvislých komponent. Formálně toto můžeme formulovat tak, že definujeme práh n_0, pomocí kterého omezíme maximální velikost 2-souvislé komponenty (počet vrcholů). Je totiž málo pravděpodobné, že stránky reprezentované uzly v rozsáhlé 2-souvislé komponentě budou mít navzájem velkou podobnost. Další možností, jak urychlit hledání 2-komponent, je eliminace kružnic. Opět můžeme tvrdit, že dva nejvzdálenější vrcholy ve velké kružnici jsou si málo podobné. Jinými slovy, bude-li v průběhu hledání nalezena 2-souvislá komponenta, pro kterou platí, že $\frac{|H|}{|V|}$ se blíží 2, potom můžeme říct, že se tato komponenta podobá kružnici. Bude-li pak $|V| \geq$ zvolený práh, nemusíme se touto komponentou dále zabývat.

4 Experimenty

4.1 Popis testovací kolekce Webtrec

Pro potřeby otestování uvedených přístupů, tj. rozkladu grafové struktury na 2-souvislé komponenty, jsme použili kolekci Webtrec. Tato testovací kolekce byla (do konce roku 2004) poskytována organizací CSIRO (Commonwealth Scientific and Industrial Research Organisation), viz [4]. Vytvořena byla postupným stahováním webových stránek v doméně *.gov* na počátku roku 2002. Dokumenty byly oříznuty na velikost $100kb$, protože původně velikost testovací kolekce byla $35.5G$. Nyní je velikost kolekce $18.1G$. Další informace jsou uvedeny v tabulce 1.

4.2 Výsledky experimentů

Prováděli jsme experimenty pro hledání 2-souvislých komponent nad několika různými neorientovanými grafy, které reprezentovaly určitou část testovací kolekce Webtrecu:

- Grafy G_1 a G_2 reprezentovaly celý Webtrec. Před tvorbou grafu G_2 jsme ale z kolekce odstranili všechny stránky typu .pdf, .gif, .txt.
- Grafy G_3, G_4 a G_5 vznikly postupným procházením webových stránek Webtrecu od určité kořenové stránky. U grafu G_3 byla omezena hloubka zanoření na 10 a u grafu G_4 resp. G_5 byl naopak omezen počet stránek na cca 200000 resp. 100000.

Tabulka 1. Parametry testovací kolekce Webtrec

Dokumentů	1247753 (1.25 miliónů)
text/html	1053372
application/pdf	131333
text/plain	43754
application/msword	13842
application/postscript	5673
ostatní	44
Svazků	4613
Celková velikost	19455030550 = 18.1G
Průměrná velikost svazku	4217435 = 4.0M
Průměrná velikost dokumentů	15592 = 15.2k
Velikost oříznutí souborů	100kb
Dokumenty, které neobsahovaly slova	55

Parametry jednotlivých grafů jsou uvedeny v tabulce 2.

Uvedené grafy jsme analyzovali pomocí dvou nezávislých produktů. Prvním byl program *Pajek*, viz [3]. Přehled výsledků dosažených programem *Pajek* je v tabulce 2. Výsledky testů jsou si do značné míry podobné. Ve všech případech většina nalezených komponent obsahovala 2 vrcholy. Zbývající komponenty měly do 15 vrcholů, s výjimkou jedné komponenty (viz tabulka 3), která měla pouze o něco málo méně vrcholů nežli celý graf.

Tabulka 2. Parametry testovacích grafů a výsledky hledání 2-komponent.

Graf	Vrcholů	Hran	Počet komponent	Počet komponent s více než 2 vrcholy
G_1	1 247 753	11 164 829	158 920	272
G_2	1 053 372	10 606 653	100 642	280
G_3	179 571	700 267	96 215	47
G_4	200 010	708 469	96 924	62
G_5	100 006	256 231	57 769	24

Tento výsledek vyplývá z charakteru testovací kolekce Webtrecu. Stránky byly totiž získány z jedné společné domény .gov, kde je provázanost asi tak vysoká, že vzniká jediná 2-souvislá komponenta. Existence spousty malých komponent je naproti tomu způsobena spoustou odkazů z hlavní velké skupiny stránek někam, kde již nebylo dál prováděno stahování do Webtrecu.

Narozdíl od [7] (kde se pracovalo s grafem o pouhých 171 vrcholech) se ovšem nepotvrdilo, že velké 2-souvislé komponenty se podobají kružnicím, což vyplývá z tabulky 3, ve které je uveden vypočítaný poměr $\frac{|H|}{|V|}$ (kde V je množina vrcholů a H je množina hran). Proto jsme dále zjišťovali, jaké stupně mají vrcholy v jednotlivých komponentách a zjistili jsme, že se v těchto 2-komponentách nalézají

Tabulka 3. Vlastnosti největší nalezené 2-komponenty.

Graf	Počet vrcholů v největší komponentě	Počet hran v největší komponentě	Poměr $\frac{\lvert H\rvert}{\lvert V\rvert}$
G_1	1 067 700	10 995 025	8,948
G_2	940 332	10 494 707	11,161
G_3	83 278	602 940	7,240
G_4	102 974	608 011	5,904
G_5	42 188	196 565	4,659

vrcholy s vysokým stupněm. Výsledky lze nalézt v tabulce 4. Pro ověření výsledků a nalezení 2-souvislých komponent byla dále použita komerční knihovna LEDA, viz [9]. Nalezený výsledek byl ale naprosto identický s výsledkem programu *Pajek*.

Tabulka 4. Stupně vrcholů v největší nalezené 2-komponentě.

Graf	Nejnižší stupeň	Nejvyšší stupeň	Průměr stupňů všech vrcholů komponenty
G_1	2	44628	20.596
G_3	2	2884	14,480
G_4	2	1964	11.809
G_5	2	644	9.318

5 Závěr

Zdá se, že se nacházíme teprve na počátku procesu, na jehož konci bude porozumění vlastnostem grafu Webu (ale i dalších sítí), že používané postupy jsou souborem různých často nesouvisejících technik. Provedené experimenty ukazují, že extrakce navigační struktury z grafu Webu je potřeba provádět v lokálním měřítku, tj. hledat v okolí vyhledané stránky.

Nejdůležitějším výsledkem výzkumu je zjištění, že pro práci s grafem Webu je vhodné zavést podobný pojem jako je stop slovo v Information Retrieval, tj. stop vrchol. Stop vrchol je vrchol grafu Webu s vysokým stupněm. Tyto vrcholy je třeba z dalšího zpracování vyloučit. Naše další práce bude pokračovat následujícími kroky:

1. Budeme opakovat experimenty s grafy, ze kterých vyloučíme vrcholy s nejvyššími stupni.
2. Provedeme tytéž experimenty s orientovanými grafy a dále budeme hledat 3-souvislé komponenty.
3. Chceme se zaměřit na využití náhodných projekcí, viz [14] pro řešení některých grafových úloh souvisejících právě s k-souvislostí grafu.

4. Dále se pokusíme nalézt odpovědi na následující otázky:
   - Pro jaké k a jaký typ vlastností webových stránek obsahují k-komponenty více něž p procent vrcholů, které mají danou vlastnost? Například jestliže zkoumanou vlastností S_i je vlastnost „prodat výrobek i“, můžeme zjistit, že pro $k = 2$ (tedy pro 2-souvislé komponenty) má např. průměrně 40% stránek vlastnost S_i pro nějaké i, ale pro $k = 3$ se procento stránek s touto vlastností zvýší na 60%.
   - Budeme zkoumat průměr k-souvislých komponent a porovnávat jak souvisí s p pro dané k.

Příspěvek vznikl za podpory GAČR grant 201/03/1318.

Reference

1. Baeza-Yates R., Castillo, C. Relating Web Characteristics with Link Based Web Page Raking. *Proceedings of SPIRE 2001, pp. 21-32.* Chile 2001.
2. Chakrabarti, S. *Mining Web.* Morgan Kaufnann Publishers, 2003.
3. Batagelj V., Mrvar A. http://vlado.fmf.uni-lj.si/pub/networks/pajek/. Pajek.
4. CSIRO. http://www.csiro.au/.
5. Henzinger M. Algorithmic Challenges in Web Search Engines. *Internet Mathematics vol. 1, No. 1: 115-126.* 2003.
6. Hung-Yu K., Shian-Hua L., Jan-Ming H., Ming-Syan Ch. Mining Web Informative Structures and Contents Based on Entropy Analysis. *IEEE Transactions on Knowledge and Data Engineering vol. 16, No. 1.* 2004.
7. Húsek D., Řezanková H., Snášel V. Vyhledávání a grafová struktura Webu. *Datakon 2004, v tisku.*
8. Chatrand G., Lesniak L. *Graphs & Digraphs.* Chapmann & Hall/CRC, 1996.
9. Naeher L., Mehlhorn K. http://www.mpi-sb.mpg.de/LEDA/. LEDA.
10. Newman M. The Structure and Function of Complex Networks. *SIAM Review, vol. 45 (2003), 167-256.* 2000.
11. Pool I., Kochen M. Contacts and influence. *Social Networks, 1 (1978), pp. 148.* 1978.
12. Rajagopalan S. et all. Graph structure in the web. *Computer Networks, 33 (2000), pp. 309-320.* 2000.
13. Tarjan R. Testing Graph Connectivity. *Sixth Annual ACM Symposium on Theory of Computing, Assoc. Comput. Mach. (1974), 185-193.* New York 1974.
14. Vempala S. The Random Projection Method. *DIMACS, 2004, vol. 65.* 2004.

Annotation:

Searching of connected components and reduction of Web graph

Present Web dimension causes many problems, especially problems that are involved in searching of particular pages. There is presented in this paper technique, that makes navigation during searching possible. This technique is based on searching of 2-connected component. If Web pages resp. individual Web links are represented as graph nodes resp. as graph edges, then we can use some known graph methods for component searching. These components represents clusters of related pages.

Dynamic search of relevant information

Peter Gurský and Tomáš Horváth

Institute of Computer science, University of P. J. Šafárik, Jesenná 5, 041 54 Košice, Slovak Republic

gursky@upjs.sk, thorvath@science.upjs.sk

Abstract. In this paper we introduce the method for the integration of ranked distributed data with use of dynamically learned monotone aggregation function. We use the method of monotone graded classification to learn the new aggregation functions in dependency of user preferences on returned objects. This method helps user to specify his/her requirements without any knowledge about the values of attributes of the objects and retrieve more relevant object consecutively. We show our method on an illustrative example.

Keywords: integration of distributed information, middleware, aggregation, generalized annotated programs

1 Introduction

Today is becoming important for Semantic web, browsers, relational and object database systems to be able to access multifeature data, such as video, images, audio, text and another objects with many different attributes. Such attributes are typically fuzzy. There are several algorithms in this area, which solve this problem. Fagin [2] introduced "Fagin's algorithm" which solve this problem first time. Fagin et al. [3] presented "threshold" algorithm. Nepal et al. [4] defined algorithm that is equivalent to threshold algorithm. Güntzer et al. [1] define "quick-combine" algorithm with heuristic rule that determines which sorted list should be preferred. None of them do support any learning of the aggregation function, which is used to combine the attributes.

In this paper we use the modification of the *x/Δx switch algorithm*, which was presented by Gurský et al. [10]. To learn the aggregation function, which depends on user preferences, we use the method of monotone graded classification presented by Horváth et al. [9]. For better understanding of our motivation, we introduce the following example.

This example concerns with hotel reservation system in Košice, Slovakia. We suppose the following query:

[1] This work was supported by the grant VEGA 1/0385/03 and 'Štátna úloha výskumu a vývoja „Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov" prierezového štátneho programu „Budovanie informačnej spoločnosti".'

```
Find hotels in Košice that are far from the airport,
their cost is reasonable and their building is fine.
```
(1)

In order to process this query we need to specify a user's notions of being near to, cost reasonable and building is fine by the fuzzy sets for example. To retrieve and order hotels we also need an aggregation function, which compute the overall score of hotels according to values of all three attributes. It is possible, that each attribute is supported by the different server (web service).

Our idea of the middleware system, which supports these requirements, is as follows. The user specifies his/her fuzzy constants by the fuzzy sets and sent them to the web services. With the use of the fuzzy set, the web service can compute the rank of the supported attribute of each object and create the list of the objects ordered by the ranks. It is also possible that the web service does not support user rankings and use only its own (possibly hidden) ranking. In this case the web service has the only one ordered list. Now the user can specify, which attribute is more important for him/her and which is not. For example the user can preference the hotels that are cheap before the new ones. This is possible to do it by creating an aggregation function (i.e. a weighted mean). Otherwise, we can use the simple arithmetic mean. Now the system computes the top *k* objects using the *x/*Δ*x switch algorithm*. In the ideal case the user becomes exactly the object, which he/she was searching for. Otherwise the user can estimate each retrieved object using some scale (commonly 0…10 (from very bad to very good)). Based on these values the system computes the new aggregation function and retrieves new top *k* objects, which are more relevant than the previous ones.

The other idea of the middleware does not allow the user to now, which or how many attributes are used by the system. In the first step the user receives some objects from the domain. After that he/she can evaluate each object using the scale, than the system computes an aggregation function and retrieve the top *k* objects. In this case, the query from our example can look like: "*Find hotels in Košice that are good for me*".

The last idea can be the combination of the previous ones. The user could use, during the specification of his/her own aggregation function, only some of the known attributes. For example the user's query can be: "*Find hotels in Košice with the reasonable cost*".

The other areas in which the learning of aggregation function can be useful are information retrieval or multimedia databases.

In this paper we present such a system that can solve these problems.

In chapter 2 we describe the *x/*Δ*x switch algorithm* and its modification. In chapter 3 we discuss, which learning method can be suitable and using the *method of monotone graded classification*. Chapter 4 concludes our paper.

2 Aggregation

Assume we have a finite set of objects. Cardinality of this set is *N*. Every object *x* has *m* attributes $x_1, ..., x_m$. All objects are in lists $L_1, ... L_m$, each of length *N*. Objects in list L_i are ordered descending by value of object in attribute x_i. We can define two

functions to access objects in lists. Let x be an object. Then $s_i(x)$ is the grade (or score, rank) of object x in list L_i and $r_i(j)$ is object in list L_i in j-th position. The lists can be accessed in two ways. First one is *sorted* (sequential) *access*. Using this type of access the grades of objects are obtained by proceeding through the list sequential from the top. The second mode of access is *random access*. Here, we request the grade of object x in list L_i and obtain it in one random access.

We have also monotone aggregation function F, which combine grades of object x from lists L_1, ..., L_m. The overall value of object x we denote as $S(x)$ and it is computed as $F(s_1(x), ... s_m(x))$.

Our task is to find top k objects with highest overall grades. We also want to minimize time and space. That means we want to use as low sorted and random accesses as possible.

There are many algorithms in this area [1, 2, 3, 4, 10]. Gurský et al. [10] experimentally showed, that the most powerful is the *x/Δx switch algorithm*.

2.1 x/Δx switch algorithm

For each list L_i, let $u_i = s_i(r_i(z_i))$ be the grade of the attribute of the last object seen under sorted access. Define the *threshold value* τ to be $F(u_1, \ldots, u_m)$. Because we assume that we have a monotone aggregation function F and the lists are sorted descend by their values, the threshold is the value, which none of still unseen objects can reach [2]. This enforces that when all objects in the top k have their values greater or equal to the threshold, then this top k list is final and there is none unseen object with greater value. This property is very important to have the algorithm correct.

Let $z = (z_1,...z_m)$ be a vector, which assigns each $i = 1, \ldots m$ the position in list L_i last seen under sorted access. Let H be a heuristics that decides which list (or lists) should be accessed next under sorted access. Moreover, assume that H is such, that for all $j \leq m$ we have $H(z_j) = z_j$ or $H(z_j) = z_j+1$ and there is at least one $i \leq m$ such that $H(z_i) = z_i+1$. The set $\{i \leq m: H(z)_i = z_i+1\}$ we call the set of candidate lists for the next sorted access.

In this algorithm we use three types of heuristics.

First type of heuristics (denote $\boldsymbol{H_1}$) do the parallel sorted access in each list. It means that for each $i \leq m$ holds $H(z_i) = z_i+1$. This heuristics was firstly presented by Fagin et al. [3] in "threshold" algorithm. This kind of heuristics is used only in the first phase of computation to retrieve the beginnings of the lists. Next two heuristics are used in the rest of computation.

The use of the $((\partial F/\partial x)*\Delta x)$ heuristics ($\boldsymbol{H_2}$) was first presented by Güntzer et al. [1] as a part of "quick-combine" algorithm. Let us look at the next unequation. For each object x in the final top k list must hold:

$$S(x) = F(s_1(x), .. s_m(x)) \geq \tau \qquad (2)$$

Hence, when we can say, that this unequation holds, we have the final top k list. Obviously, there are two ways to make (2) hold: to increase the left side or to decrease the right side. Heuristics H_2 tries to decrease τ as fast as possible. As a criterion which list is suitable in the next sorted access, we use the next equation:

$$\Delta_i = \left(\frac{\partial F}{\partial x_i}(s_1(r_1(z_1)),\ldots,s_m(r_m(z_m))) \right)^{-} * (s_i(r_i(z_i - p)) - s_i(r_i(z_i)) \qquad (3)$$

The constant p is some suitable (small) natural number. Hence Δ_i is the multiplication of the partial derivative of aggregation function F from the left in the point $(s_1(r_1(z_1)),\ldots, s_m(r_m(z_m)))$ and the expected change of values in p steps (Δx factor) of the i-th list. When we have Δ_i for each $i \le m$, we can set the value of $H(z_i)$. Heuristics H_2 sets $H(z_i)=z_i+1$ if $\Delta_i = \max\{\Delta_j;\ j \le m\}$. Otherwise $H(z_i)=z_i$. The only necessary condition we required from F is the continuity from the left.

The *((∂F/∂x)*x)* heuristics (***H₃***) is a variation of the last one. This heuristics was presented by Gursky et al.[11] at the first time. Instead of following the Δx factor, H_3 chooses an x-factor, which is the last seen value in the i-th list. The criterion for this heuristics is:

$$c_i = \left(\frac{\partial F}{\partial x_i}(s_1(r_1(z_1)),\ldots,s_m(r_m(z_m))) \right)^{-} * (s_i(r_i(z_i))) \qquad (4)$$

The criterion (4) computes the partial derivation of F from the left in the point $(s_1(r_1(z_1)),\ldots, s_m(r_m(z_m)))$ and multiply it with value in the point $s_i(r_i(z_i))$ in L_i. This heuristics sets $H(z_i)=z_i+1$ if $\chi_i = \max\{\chi_j;\ j \le m\}$. Otherwise $H(z_i)=z_i$. We need the continuity from the left for the function F again.

When the aggregation function is the simple weighted mean, the derivation used by these heuristics is a constant, more precise it is the weight of the attribute.

Now we can describe the *x/Δx switch algorithm*:

0. $z:=(0,\ldots,0)$, set the suitable small natural p.
1. if any $z_i<p$ then $H:=H_1$; otherwise if $H=H_1$ then $H:=H_2$; if $H=H_2$ then $H:=H_3$; and if $H=H_3$ then $H:=H_2$.
2. Do sorted access in parallel to each of the sorted lists to all positions where $H(z)_i = z_i+1$. Put $z_i = H(z)_i$.
3. First control: Compute the *threshold value* τ. As soon as at least k objects have been seen whose grade is at least equal to τ , then go to step 6.
4. With the object x that was seen under sorted access in some list, do random access to the other lists to find the grade $s_i(x)$ of object in every list. Then compute the grade $S(x) = F(s_1(x), .. s_m(x))$ of object x. If this grade is one of the k highest ones we have seen, then remember object x and its grade $S(x)$ (ties are broken arbitrarily, so that only k objects and their grades need to be remembered at any time).
5. Second control: As soon as at least k objects have been seen whose grade is at least equal to τ , then go to step 6, otherwise go to step 1.
6. Let Y be a set containing the k objects that have been seen with the highest grades. The output is then the graded set $\{(x, S(x)) : x \in Y\}$.

Algorithm 1. x/Δx switch algorithm

The main idea of this algorithm is to try to keep the left hand side of (2) big (using the heuristic H_3) and to decrease the right hand side of (2) (heuristic H_2) simultaneously.

3 Combining aggregation and classification

There are some problems in case of changing the aggregation function during the search. One of them is that the implementation of the computed aggregation function to the algorithm is complicated (the evaluation of the partial derivation). The second is that sometimes it is hard to compute the aggregation function. That is why we use instead of computing of aggregation function the results of classification methods. Its advantage is that the results are interpreted as a table, so it is easy to use.

Now we have little problems how to change the *x/Δx switch algorithm* to compute the overall score of each object when we do not have any aggregation function but the classification of objects in dependence of the attribute values. We need to solve two things. Firstly we need to simulate the computation of aggregation function and secondly we need to define how to compute the criteria of heuristics H_2 and H_3.

The meaning of any classification rule is as follows: When the attributes of object x have the values greater or equal to relevant values on the right side of the rule then the overall value of x is at least the same as on the left side of the rule. Hence during the simulation of computation of aggregation function we can simply test the validity of requirements of the rules from the strongest one to weaker rules. When we find the rule that holds, we can say that the overall value of the object is the value on the left side of the rule. Since we test the sorted rules, we always rank the object with the highest possible value.

The problem of computing the criteria Δ_i and χ_i of heuristics H_2 and H_3 is at first sight in the interpretation of the partial derivation. In the case of classifications we do not need any computation. We must to remind the first motivation of both heuristics. H_2 tries to decrease the threshold value and H_3 tries to keep the values of new objects high. Hence in the first case we can do the sorted access to the list, in which the last seen value is the nearest to the lower bound of the class (values on the right side of the rules). The second heuristic prefers the list of which the last seen value is the most far from all lower bounds of class. The H_2 heuristic can refined by parameter p when we look back in each attribute to analyze the speed of decreasing to the lower bounds and overall criterion for this heuristic can be some combination of distances and the speed. The problem of this combination is the object of our next research.

In figure 1 we can see the user ranking of some concrete hotels. The result of classification method can be interpreted as following rules:

```
y(A,0.9) :-  x1(A,0.7), x2(A,0.8).
y(A,0.8) :-  x1(A,0.9), x2(A,0.6).
y(A,0.8) :-  x1(A,0.6), x2(A,0.7).
y(A,0.8) :-             x2(A,0.9).
y(A,0.7) :-             x2(A,0.8).
y(A,0.7) :-  x1(A,0.6), x2(A,0.6).
y(A,0.7) :-  x1(A,0.3), x2(A,0.7).
y(A, 0.7) :- x1(A,0.7), x2(A,0.5).
y(A, 0.6) :- x1(A,0.6), x2(A,0.3).
y(A, 0.6) :- x1(A,0.3), x2(A,0.6).
y(A, 0.6) :- x1(A,0.4), x2(A,0.5).
y(A, 0.6) :-            x2(A,0.7).
y(A, 0.5) :- x1(A,0.1), x2(A,0.5).
y(A, 0.5) :-            x2(A,0.6).
y(A, 0.5) :- x1(A,0.3), x2(A,0.3).
y(A, 0.5) :- x1(A,0.4), x2(A,0.2).
y(A, 0.4) :- x1(A,0.2), x2(A,0.2).
y(A, 0.4) :- x1(A,0.4), x2(A,0.1).
y(A, 0.4) :-            x2(A,0.4).
y(A, 0.3) :- x1(A,0.3).
y(A, 0.3) :-            x2(A,0.1).
```

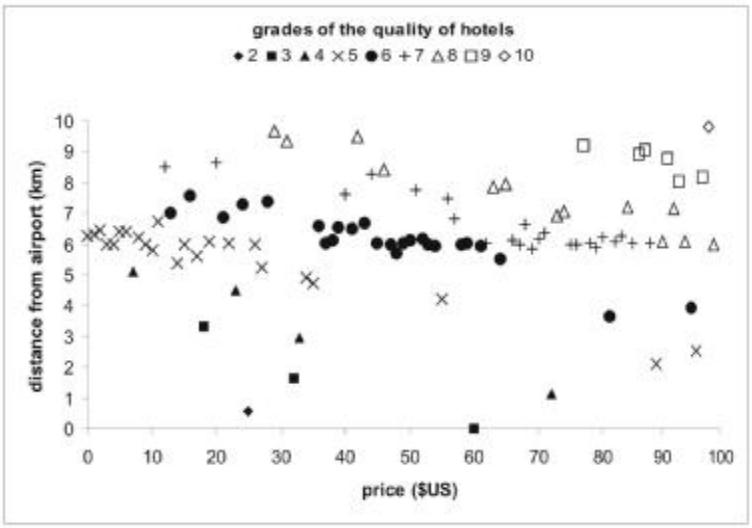


Figure 1. Illustrative example of users overall preference of (100) hotels. The grades 2, 3,..., 10 means the classes of "the worst", "bad", ..., "the best" hotels according to users criterion. User did not rank any hotel by the grades 0 or 1.

These rules are graphically represented in figure 2. It can be easily seen that the computation of the overall value or computing the heuristics is easy to solve. Using this classification we are able to retrieve more relevant objects to user requirements.

The use of some classification method in the *x/Δx switch algorithm* is conditioned only by monotone classification, but selected method is suitable.

Main interest is to learn the (user dependent) classification (ranking, grading, …), which gives and overal score to a graded fulfilment of all user's requirements. This appears also in mulcriterial and/or multiuser decision making and also in graded classification (where the monotonicity of dependence can be more problematic).

For our purposes it is convenient to induce rules of the form

$$\text{IF body\_attribute}_1 \geq (\leq) \text{ grade}_1 \text{ AND } \ldots \text{ AND body\_attribute}_n \geq (\leq) \text{ grade}_n \quad \text{THEN head\_attribute} \geq (\leq) \text{ grade}_H \tag{5}$$

where $grade_H = @(grade_1, ..., grade_n)$, @ is an n-ary aggregation operator.

These types of rules preserve the monotonicity (both in the directions down and up depending on the directions of the monotonicity of attributes).

Our method of learning the (monotone graded) classification [8, 9] is based on multiple use of classical Inductive logic programming system ALEPH [6, 7] with additional monotonicity axioms in the background knowledge.

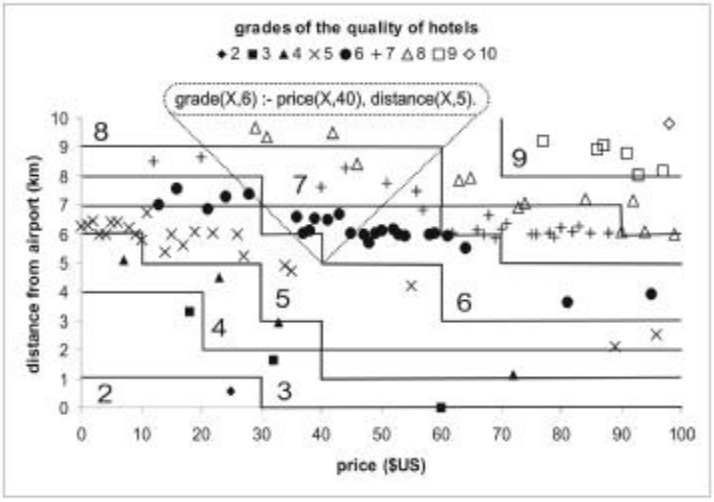


Figure 2. Result of the graded classification method with illustration of one rule

An example of an induced rule: *grade(X,6) :- price(X, 40), distance(x,5).* It means: IF price ≥ 40 $ AND distance ≥ 5 km THEN grade of the hotel ≥ 6.

These rules have the expected form and they preserve the monotonicity (the lowest class contains all hotels, the better just a few of them).

4 Conclusion

In this paper we proposed the method of searching of relevant information from distributed sources. The user is able to do his classification of retrieved objects to specify his/her requirements easily.

We integrated the modification of the *x/Δx switch algorithm* and the *method of monotone graded classification*. First of them use the classification from the second one instead of the aggregation function.

In the future work we want to implement this method and test it on the real data.

References

1. Güntzer, U., Balke, W., Kiessling, W.: Optimizing Multi-Feature Queries for Image Databases, Proceedings of the 26th VLDB Conference, Cairo, Egypt, 2000

2. Fagin, R.: Combining fuzzy information from multiple systems. J. Comput. System Sci., 58:83-99, 1999
3. Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware. In Proc. 20th ACM Symposium on Principles of Database Systems, pages 102-113, 2001
4. Nepal, S., Ramakrishna, M. V.: Query processing issues in image (multimedia) databases. In Proc. 15th International Conference on Data Engineering, pages 22-29, 1999
5. Natsev, A., Chang, Y-C., Smith, J.R., Li, C-S., Vitter, J.S.: Supporting incremental join queries on ranked inputs. In Proc. 27th Very Large Databases Conference, pages 281-290, 2001
6. Džecroski, S., Lavrač, N.: An introduction to inductive logic programming. Relational data mining, S. Džeroski, N. Lavrač eds. Springer, 48-73, 2001
7. Srinavasan, A.: The Aleph Manual. Tech. Rep. Comp. Lab. Oxford Univ. 2000 at http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html
8. Vojtáš, P., Horváth, T., Krajči, S., Lencses, R.: An ILP model for a monotone graded classification problem. Kybernetika 40, 317-332, 2004
9. T. Horváth, S. Krajči: Integration of methods of Clustering via Conceptual lattices and Inductive logic programming for a graded classification problem. In: ZNALOSTI '04, Brno 2003: FEI VŠB-TU Ostrava, 2004, ISBN 80-248-0456-5, pp. 297 – 308.
10. Gurský, P., Lencses, R.: Aspects of integration of ranked distributed data. In proc. Datakon, 2004
11. Gurský, P., Lencses, R., Vojtáš, P.: Algorithms for user dependent integration of ranked distributed information, technical report, 2004
12. Naito, E., Ozawa, J., Hayashi, I., Wakami, N.: A proposal of a fuzzy connective with learning function and query networks for fuzzy retrieval systems. In Fuzziness in database management systems. P. Bosc and J. Kacprzyk eds. Physica Verlag, 345-364, 1995

Fuzzy ILP: prístupy a problémy

Tomáš Horváth

Ústav informatiky, Prírodovedecká fakulta, Univerzita P. J. Šafárika
Jesenná 5, 041 54 Košice

horvath@ics.upjs.sk

Abstrakt. Induktívne logické programovanie (ILP) je efektívnym nástrojom dolovania znalostí v multirelačných dátach. Má rozvinutú teóriu s niekoľkými metódami a rôznymi stratégiami prehľadávania priestoru možných riešení - v dvojhodnotovom (crisp) logickom programovaní. Výsledkom ILP je logický program. Pri indukovaní fuzzy logických programov sa však stretávame s rôznymi problémami, ako napr. veľké množstvo neznámych operátorov, atď. Tento článok sa zaoberá fuzzy induktívnym logickým programovaným (FILP). Popíšu sa rôzne problémy pri hľadaní riešenia a uvedie sa formálny model fuzzy logických programov (FLP) a FILP. Popíšu a porovnajú sa rôzne prístupy k riešeniu úlohy FILP.

Kľúčové slová: Induktívne logické programovanie, fuzzy logické programovanie, dolovanie dát.

1 Úvod

Jednou metódou dolovania dát je induktívne logické programovanie ILP, ktorá sa úspešne používa pri dolovaní multirelačných dát použitím predikátovej logiky. Pri tejto metóde sú dané množiny kladných a záporných príkladov (objekty). Úlohou je popísať kladnú množinu objektov nejakými pravidlami pomocou tzv. základnej znalosti o týchto objektoch. Pomocou týchto pravidiel a základnej znalosti by sme mali vedieť odvodiť (pokryť) všetky kladné objekty pričom sa nesmie pokryť ani jeden záporný objekt. Poznáme niekoľko metód ILP s rôznymi stratégiami prehľadávania priestoru možných riešení v prípade klasickej (crisp) predikátovej logiky [2].

V klasickej logike však môžeme reprezentovať len fakty, ktoré sú pravdivé absolútne. Preto tento formalizmus nie je vhodný na reprezentáciu informácií, ktoré sú neisté, neurčité a/alebo vágne. Toto je dôležitý nedostatok vo vyjadrovacej sile a veľká prekážka pri použití v mnohých aplikáciách. Neurčitosti alebo vágnosti sa v reálnom svete nemôžeme vyhnúť: Naše informácie sú často nepresné a neúplné a len zopár „pravidiel“ ktoré používame pri usudzovaní sú všeobecne pravdivé.

Kvôli týmto obmedzeniam, ktoré sú nezanedbateľné v mnohých oblastiach reálneho života (medicína, ekonómia, umenie...) sa používajú rôzne modeli uvažovania v ILP ako napr. pravdepodobnostné, viachodnotové, atď. Aplikácia ILP na tieto modely je pomerne mladá disciplína, ktorá sa v terajšej dobe rýchlo rozširuje.

* S podporou grantu VEGA 1/0385/03

Súčasný výskum sa najviac sústreďuje na indukovanie logických programov s pravdepodobnosťou [1], ktoré sú založené medzi inými na teórii Bayesových sietí, niektorých štatistických modelov a stochastických gramatík. Medzi tieto typy programov patria stochastické, probabilistické a bayesové logické programy SLP, PLP, BLP.

Nasledujúci príklad znázorní rozdiely medzi rôznymi modelmi. Podľa nasledujúceho (klasického) logického programu je Jozef ohrozeným človekom na infarkt pretože fajčí a má stresové zamestnanie:

```
fajciar(jozef).
zamestnanie(manazer,jozef).
stres(manazer).
ohrozeny(X):-fajciar(X),zamestnanie(Y,X),stres(Y).
```

Tento zápis je absolútny a nie vždy platí pre všetky situácie z reality – Jozef bude ohrozený len s určitou pravdepodobnosťou. Táto skutočnosť sa môže lepšie vystihnúť pravidlom PLP (SLP)

```
0.6: ohrozeny(X):-fajciar(X),zamestnanie(Y,X),stres(Y).
```

kde sa pravidlu priradí nejaké číslo (0.6) čo vyjadruje pravdepodobnosť, ktorou hlava pravidla bude platiť v prípade platnosti tela. Faktom sa tiež môžu priradiť pravdepodobnostné hodnoty. Daná situácia sa môže modelovať aj pravidlom BLP

```
ohrozeny(X) | fajciar(X),zamestnanie(Y,X),stres(Y).
```

čo vyjadruje podmienenú pravdepodobnosť hlavy od tela pravidla.

Medzi rôznymi metódami sa môžu spomenúť logické programy s anotovanými disjunkciami LPAD [7], kde hlava pravidla je disjunkcia rôznych anotovaných atómov podľa pravdepodobnosti pravdivosti daného atómu v závislosti tela.

Tieto programy reprezentujú reálny svet lepšie ako klasické programy avšak majú niekoľko nedostatkov. Napr. človek môže byť silným alebo „príležitostným“ fajčiarom, čo má vplyv na výsledok ako aj to, že práca manažéra je stresovejšia vo veľkej medzinárodnej spoločnosti ako v malej firme. Nakoniec, nie každá vlastnosť je rovnocenná – stresová práca môže mať väčší vplyv na infarkt ako fajčenie. Táto skutočnosť sa ťažko modeluje klasickými spojkami (konjunkcie, disjunkcie) a/alebo použitím podmienenej pravdepodobnosti.

Tieto nedostatky sa dajú odstrániť použitím fuzzy logických programov FLP [8], kde faktom aj pravidlám sa priradí pravdivostná hodnota a v pravidlách sa môžu použiť rôzne spojky – podľa typu úlohy. Predchádzajúca situácia vo FLP sa môže reprezentovať ako

```
fajciar(jozef).0.6
zamestnanie(manazer,jozef).
stres(manazer).0.8
ohrozeny(X):-
      @(fajciar(X),zamestnanie(Y,X),stres(Y)).0.7
```

čo vyjadruje napr., že Jozef je „celkom silný fajčiar“, práca manažéra je „veľmi stresujúca“, pravidlo platí s určitou pravdivostnou hodnotou a agregačná funkcia @ môže byť napr. @(x,y,z)=(x+2*z)/3 podľa čoho zamestnanie má väčší vplyv na infarkt ako fajčenie. Ako vidíme, fuzzy logika je účinná aj pri používaní pojmov prirodzenej reči (veľký, silný, ...) avšak indukcia FLP je zložitá úloha, napr. kvôli veľkému množstvu vopred neznámych typov @, atď. Asi aj kvôli tomu je počet vedeckých prác popisujúcich problematiku fuzzy ILP pomerne malý.

2 Induktívne logické programovanie ILP

Indukcia znamená nájdenie všeobecných pravidiel popisujúcich dáta, o ktorých máme veľa dielčích informácií v príkladoch. Je to opak dedukcie, pri ktorej z danej teórie vieme odvodiť vlastnosti jednotlivých objektov. V ILP [2] príkladovú množinu zvyčajne delíme do dvoch tried (kladné a záporné príklady - E^+ a E^- - dané ako množiny atómov). Naviac je daná základná znalosť B - množina faktov alebo/a pravidiel. Úlohou je nájdenie hypotézy H - množiny pravidiel – takej, že platí

$$\begin{aligned} &\forall e \in E^+ : (H \cup B) \models e \quad \textit{(kompletnosť)} \\ &\forall e \in E^- : (H \cup B) \not\models e \quad \textit{(konzistentnosť)} \end{aligned} \tag{1}$$

Hypotéza, ktorá je kompletná aj konzistentná sa nazýva *korektná*. Pre programy bez negácie je sémantika daná minimálnym Herbrandovým modelom $M_{B\cup H}$ (viď [?5]) a podmienka (1) sa dá prepísať nasledovne:

$$E^+ \subseteq M_{B\cup H} \quad \textit{(kompletnosť)} \qquad\qquad M_{B\cup H} \cap E^- = \varnothing \quad \textit{(konzistentnosť)} \tag{2}$$

Príklad.
E^+ = {dcera(mary,ann), dcera(eve,tom)}
E^- = {dcera(tom,ann), dcera(eve,ann)}
B = {rodic(ann,mary), rodic(tom,eve), zena(mary), zena(eve)}
Úlohou je zostrojiť algoritmus ktorý nájde H = {dcera(X,Y) :- zena(X), rodic(Y,X)}.

Priestor možných hypotéz - hornových klauzúl - je zväz usporiadaný podľa subsumpcie (existuje substitúcia ktorá jednu klauzulu špecializuje na podmnožinu druhej). To je základom ILP metód a algoritmov.

3 Fuzzy logické programovanie FLP

3.1 Fuzzy logika

Jazyk £ má dva typy syntaktických objektov: logické a kvantitatívne. Logické pozostávajú z viacsortového predikátového počtu bez funkčných symbolov.

Kvantitatívne sú racionálne čísla z jednotkového intervalu $[0,1] \cap Q$. Jazyk £ má viacero spojok z každého typu – vo viachodnotovej logike je to potrebné ak chceme aby jazyk bol schopný popísať všetky typy interakcií medzi rôznymi predikátmi – niekoľko viachodnotových konjunkcií $\&_1$, $\&_2$, ..., $\&_k$, disjunkcií $\vee_1$, $\vee_2$, ..., $\vee_l$, implikácií $\rightarrow_1$, $\rightarrow_2$, ..., $\rightarrow_m$ a agregácií $@_1$, $@_2$, ..., $@_n$ (každé zloženie týchto spojok je opäť agregácia [8]). Hodnoty týchto spojok označujeme $^\bullet$, napr. $\rightarrow^\bullet$, $\&^\bullet$, $\vee^\bullet$, $@^\bullet$.

Hlavný syntaktický objekt jazyka sú ohodnotené formule $(\varphi.\beta)$, kde φ je formula a β je racionálne číslo z intervalu $[0,1] \cap Q$. *Interpretácia* fuzzy jazyka I je funkcia ktorá zobrazuje konštantné atómy (prípadne elementárne výroky) do intervalu [0,1]. Táto logika je extenzionálna, t.j. pravdivostná hodnota formule sa počíta pozdĺž vytvárajúcej postupnosti formule s použitím pravdivostných funkcií spojok (to je aj hlavný rozdiel s pravdepodobnostným prístupom). Takto možno I rozšíriť na všetky formule. Interpretácia I je *modelom* ohodnotenej formule $(\varphi.\beta)$ ak $I(\varphi) \geq \beta$. Pre pravidlo $(A \leftarrow @((B_1,...,B_n).\beta)$ to znamená, že $I(A \leftarrow @((B_1,...,B_n)) = \leftarrow^\bullet(I(A), @^\bullet(I(B_1),...,I(B_n))) \geq \beta$.

Definícia 1. Interpretácia I je modelom programu P: Formule $\rightarrow [0,1] \cap Q$ ak pre každú formulu $\varphi \in P$ platí $I(\varphi) \geq P(\varphi)$.

3.2 Dedukcia vo fuzzy logike a fixpointová sémantika

Keď sa pracuje vo fuzzy logike s viacerými spojkami, často sa stáva že $A \leftarrow B$ nie je ekvivalentné s $A \vee \neg B$. Keďže neplatí zákon vylúčeného tretieho, nemôžeme automaticky použiť SLD rezolúciu [3]. Deduktívna sémantika je založená na *fuzzy modus ponense* ktorý v reči ohodnotených formúl vyzerá nasledovne

$$\frac{(B.\beta),\ (H \leftarrow_i B.\rho)}{(H.C_i(\beta, \rho))} \qquad (3)$$

Toto pravidlo je korektné a dáva najlepšiu možnú odpoveď ak C_i je reziduálny konjunktor vzhľadom k implikátoru $\rightarrow_i$ (viď [8]). Fuzzy logický program je množina ohodnotených implikácií a atómov.

Pre definíciu fuzzy ILP problému potrebujeme pojem fuzzy minimálneho modelu. Ten je pevným bodom nasledujúceho operátora na zväze všetkých fuzzy interpretácií jazyka. Príslušný *produkčný operátor* je definovaný následovne: pre fuzzy interpretáciu I, pre konštantný atóm A a fuzzy logický program P sa položí hodnota operátora rovnej

$$T_P(I)(A)=\max\{\ \sup\{C_i(I(B),\rho): (A\leftarrow_i B.\rho) \text{ je konštantná inštancia pravidla v programe P}\},\ \sup\{\beta: (B.\beta) \text{ je základná inštancia atómu v programe P}\}\ \} \qquad (4)$$

V [8] je dokázané, že pevný bod operátora T_P je minimálnym modelom M_P fuzzy logického programu P. Operátor T_P je zväzovo spojitý ak pravdivostné funkcie všetkých spojok (okrem implikácií) a reziduálnych konjunktorov C_i sú zľava spojité.

To znamená, že pevný bod operátora T_P sa dá dosiahnuť iteráciou v nanajvýš spočítateľne mnoho krokoch a teda pre každý konkrétny atóm je jeho hodnota vypočítateľná. Deduktívna sémantika spĺňa korektnosť a úplnosť ([8]).

Definícia 2. Dvojica (Θ,x), kde Θ je substitúcia a $x \in [0,1]$ sa nazýva správna odpoveď na otázku A (?-A.) vzhľadom k programu P, ak pre každú interpretáciu I, ktorá je modelom programu P, platí $I(\forall(A\Theta)) \geq x$.

4 Fuzzy ILP

V klasickom ILP sa príkladová množina skladá z dvoch podmnožín – kladných a záporných príkladov. V tomto prípade sa každý príklad môže reprezentovať jeho pravdivostnou hodnotou (1 – kladný, 0 – záporný). Vo fuzzy ILP príkladová množina je množina ohodnotených atómov (fuzzy množina), v ktorom každý prvok má určitú pravdivostnú hodnotu (stupeň náležania do tejto množiny). Teda príklady sa nerozdeľujú na kladné a záporné ale na viac (alebo menej) kladné (záporné) príklady. Treba poznamenať, že to je určité rozšírenie klasickej príkladovej množiny.

Základná znalosť je množina formúl, ktoré môžu byť buď ohodnotené atómy alebo fuzzy pravidlá (tu treba povedať, že niektoré ILP systémy nemôžu pracovať so základnou znalosťou obsahujúcou pravidlá. Pre nadefinovanie všeobecnej formulácie fuzzy ILP by trebalo povoliť aj použitie pravidiel v základnej znalosti).

Hypotéza fuzzy ILP je množina fuzzy pravidiel. V klasickom prípade sú jednoznačne určené podmienky pre hypotézu (1), (2). Tieto však nemôžeme rovno aplikovať vo fuzzy prípade. Môžu sa však tieto podmienky nejako preformulovať – zovšeobecniť – tak, aby platili aj vo fuzzy prípade. Najprv sa to ukáže pre crisp ILP.

Príkladová množina sa môže chápať ako zobrazenie E: príklady $\rightarrow \{0,1\}$. Výsledná hypotéza H spolu so základnou znalosťou B popisujú určitú podmnožinu kladnej časti príkladovej množiny. Teda $H \cup B$ tvoria (crisp) logický program. Ak sa zadá konštantná otázka „?- e.“ vzhľadom k programu $H \cup B$ výsledok bude odpoveď „yes“ alebo „no“, podľa toho či daný príklad patrí alebo nepatrí do minimálneho modelu $H \cup B$ (fixpointový operátor pre logické programy [3]). Pre konštantné otázky (príklady sú konštantné atómy) sa môže $H \cup B$ chápať ako zobrazenie $H \cup B$: príklady $\rightarrow \{0,1\}$.

Podľa toho, aby hypotéza pokrývala všetky kladné príklady a nepokryla žiaden záporný príklad, stačí aby platilo $\forall e \in E: (H \cup B)(e)=E(e)$ (korektnosť). Často sa stáva, že hypotéza nepokrýva všetky kladné príklady, t.j. nie je splnená podmienka kompletnosti. Vtedy sa podmienka korektnosti zoslabí a bude platiť

$$\forall e \in E: (H \cup B)(e) \leq E(e). \qquad (5)$$

V tomto prípade je tiež splnená podmienka konzistentnosti.

Aplikujme tento vzťah vo fuzzy logickom programovaní. Treba všimnúť, že ak E považujeme za interpretáciu programu $H \cup B$, tak podľa definície 1. a vzťahu (5) E bude modelom programu $H \cup B$. Ďalej podľa definície 2. bude $(H \cup B)(e)$ správnou odpoveďou na otázku ?-e vzhľadom k programu $H \cup B$. Vždy bude platiť, že

$\forall e \in E$: $0 \leq E(e)$, t.j. 0-odpoveď bude správna. Tomu sa môže vyhnúť tak, že sa zavedie nejaká hranica (ε), o koľko môže byť $(H \cup B)(e)$ menšie ako $E(e)$.

Definícia 3. (Fuzzy ILP úloha) Je daná množina E ohodnotených atómov a množina B ohodnotených atómov a fuzzy pravidiel. Ďalej je dané $\varepsilon \in [0,1]$. Úlohou je nájsť konečnú množinu fuzzy pravidiel H, pre ktorú platí

$$\forall e \in E: \; E(e) - \varepsilon \leq (H \cup B)(e) \leq E(e). \qquad (6)$$

5 Fuzzy ILP metódy

V tejto kapitole sa popíšu niektoré prístupy k riešeniu fuzzy ILP úlohy. Keďže v týchto metódach sa definície fuzzy ILP úlohy odlišujú, a tým pádom aj ich riešenia, uvedú sa aj rôzne definície tejto úlohy.

5.1 FCI [6]

V tomto prístupe pod fuzzy logickým programom sa rozumie fuzzy množina (crisp) pravidiel. Cieľom systému FCI je nájsť fuzzy logický program H, ktorý spĺňa podmienky $\forall e \in E^+$: $B \cup H \models e^+ / \mu^+$ a $\forall e \in E^-$: $B \cup H \not\models e^- / \mu^-$, kde B je fuzzy logický program, E^+, E^- sú množiny (crisp) príkladov a $\mu^+, \mu^- \in [0,1]$ sú konštanty. Pravdivostná hodnota klauzule založený na Lukasiewiczovej implikácii a konjunkcii je $I(A \leftarrow B_1 \wedge ... \wedge B_n) = \min\{n + I(A) - I(B_1) - \ldots - I(B_n), 1\}$. Algoritmus FCI hľadá pravidlá, ktoré nepokrývajú záporné príklady v stupni viac ako μ^- pokiaľ každý kladný príklad je pokrytý v stupni viac ako μ^+. Fuzzy množina pokrytých príkladov crisp pravidlom c je $covered(c)=\{e/\mu \mid e \in E^+ \cup E^-, \mu = M_{B \cup \{c/1\}}(e)\}$. Obmedzenie FCI je aj to, že každé pravidlo v H má pravdivostnú hodnotu 1. FCI je implementovaný v algoritme FOIL [2].

5.2 Rôzne typy fuzzy pravidiel [4]

Tento prístup je zaujímavý tým, že sa v ňom definujú rôzne typy fuzzy pravidiel. V tomto prístupe fuzzy predikát sa chápe ako množina obyčajných predikátov, ktorých charakteristické funkcie sú stupne (hladiny) rezov $\mu_{F\alpha}$ asociované fuzzy členským funkciám μ_F, t.j. $\mu_{F\alpha} = 1$ akk $\mu(F) \geq \alpha$ inak $\mu_{F\alpha} = 0$. Fuzzy pravidlo $C(x) \leftarrow A(x)$ asociujeme crisp pravidlami $C_\beta(x) \leftarrow A_\alpha(x)$. Ak $A_\beta(x)$ platí, tak $A_\alpha(x)$ tiež platí pre $\alpha \leq \beta$, tým pádom sa uvažujú len crisp aproximácie $C_\alpha(x) \leftarrow A_\alpha(x)$, kde x označuje vektor atribútov. Pri tejto metóde sa uvažuje o troch typoch pravidiel: flexibilné pravidlo $\forall x, \exists\alpha \; C_\alpha(x) \leftarrow A_\alpha(x)$, graduálne pravidlo $\forall x, \forall\alpha \; C_\alpha(x) \leftarrow A_\alpha(x)$ a istotné (certainty) pravidlo $\forall x, \forall\alpha \; C_{1-\alpha}(x) \leftarrow A_\alpha(x)$. Pre každý typ pravidla je definovaný iný faktor istoty (confidence factor [?1], ktorý je akýmsi rozšírením $cf(C \leftarrow A) = P(A|C)/P(A)$) a iná pokrytosť cover. Práca tohto systému spočíva v tom,

že pre každý stupeň α sa hľadajú rôzne typy pravidiel pomocou ILP systému FOIL, v ktorom sa miesto iniciálnych cf a cover použije cf a cover pre daný typ hľadaného pravidla.

5.3 Implikačný operátor [5]

Táto metóda je tiež rozšírením algoritmu FOIL. Tento výpočet je zložitejší, ale výsledná hypotéza je flexibilnejšia, ako v predchádzajúcom prípade. Najprv sa vypočíta pravdivostná hodnota tela pravidla $\mu_T(A(t))$ podľa nejakej T-normy kde t je vektor atribútov a A je konjunkcia atomických formúl v tele pravidla. Množiny $L_A = \{0=\sigma_1 < \sigma_2 < \ldots < \sigma_r=1\}$a $L_C = \{0=\gamma_1 < \gamma_2 < \ldots < \gamma_s=1\}$ sú množiny pravdivostných hodnôt pre telo a hlavu pravidla. Implikačný operátor I pre pravidlo je reprezentovaný pomocou matice $r \times s$, kde $I(i,j) = I(\sigma_i,\gamma_j)$. Membership map M je matica $r \times s$, kde M(i,j) = počet rôznych ohodnotení t takých, že $\mu_T(A(t)) = \sigma_i$ a $\mu(C(t)) = \gamma_j$, kde C je hlava pravidla. Pomocou algoritmu FOIL a predefinovaných faktorov istoty cf sa určuje M, podľa ktorého sa prepočítava I, ktorej nájdenie je cieľom tejto.

5.4 IGAP [9]

Jadrom tohto prístupu je ekvivalencia FLP a generalizovaných anotovaných programov GAP [9], podľa čoho pravidlo FLP $A\leftarrow_i @(B_1,...,B_n).r$ sa pretransformuje na ekvivalentné pravidlo GAP $A:C_i(@(x_1,...,x_n),r) \leftarrow B_1:x_1,...,B_n:x_n$ (opačne pravidlo GAP $A:\rho \leftarrow B_1:\mu_1$ & ...& $B_k:\mu_k$ sa prevedie na pravidlo FLP $A\leftarrow \rho(B_1,...,B_k).1$, kde ρ je n-árny agregačný operátor). Ďalej sa využije veta, podľa ktorej interpretácia je modelom FLP ak je modelom ekvivalentného GAP a naopak. Kým vo FLP sú rôzne typy operátorov konjunkcie, implikácie GAP obsahuje len crisp spojky, čo pomôže zvládnuť problém vopred nepoznaných operátorov v pravidlách. Vo FLP sa pravdivostná hodnota vypočítava pozdĺž vytvárajúcej postupnosti formule s použitím pravdivostných funkcií týchto spojok. GAP priraďuje pravdivostnú hodnotu každej zložky formule, kým FLP priraďuje pravdivostnú hodnotu implikácii.
Táto metóda indukuje GAP pravidlá pomocou viacnásobného použitia klasického ILP systému ALEPH [2] s pridanými axiómami monotónnosti v základnej znalosti. Pracuje s fuzzy základnou znalosťou a fuzzy príkladovou množinou.

6 Záver

V tejto práci sa uviedol formálny popis fuzzy logických programov FLP a induktívneho logického programovania ILP, ďalej formalizmus fuzzy ILP problému a niekoľko metód na jeho riešenie. Tento model sa na začiatku porovnal s rôznymi pravdepodobnostnými modelmi na ilustračnom príklade.

Nedostatkom FCI je, že príkladová množina je dvojhodnotová, ktorá sa popíše nejakým fuzzy pravidlom, ktorý má vždy pravdivostnú hodnotu 1. Ďalej nedostatkom je tiež, že sa tie príklady „orezávajú“ vždy v tom istom stupni μ.

Nedostatok druhej metódy - s viacerými typmi pravidiel- je, že pravdivostné hodnoty hlavy a tela pravidla sú rovnaké (α), resp. je medzi nimi silná závislosť (α a 1-α). Pri hľadaní rôznych typov pravidiel sa počítajú rôzne výsledky – niekedy môže byť ťažké rozhodnúť, ktorý výsledok je najlepší.

Metóda implikačného operátora môže počítať pravidlá s odlišnými pravdivostnými hodnotami v tele aj v hlave pravidla. Nedostatkom však je, že na začiatku sa musí definovať príslušná T-norma, teda vhodný výber T-normy má veľký vplyv na výsledok. Nájdenie fuzzy pravidiel však znamená tiež nájdenie vzťahov medzi predikátmi v telách, teda nájdenie tejto T-normy, čo však táto metóda pokladá za určitú. Tým pádom jej výsledok nemusí byť vždy relevantný k („vopred neznámym“) dátam.

Metóda IGAP je schopná indukovať pravidlá s rôznymi pravdivostnými hodnotami v tele aj v hlave pravidla. Aj v tele výsledného pravidla sa môžu vyskytovať atómy s rôznymi pravdivostnými hodnotami, teda je schopná sa naučiť agregačnú funkciu. Nedostatkom tejto metódy je, že použitie axióm monotónnosti a príliš jemná diskretizácia vedie k zvýšeniu zložitosti tohto algoritmu.

Reference

1. Application of Probabilistic Inductive Logic Programming (APRIL II). Specific Targeted Research Project" funded by the European Commission under the "Sixth Framework Programme (2002-2006); Information Society Technologies"
2. S. Džeroski, N. Lavrač. *Relational data mining*, S. Džeroski, N. Lavrač eds. Springer 2001, 48-73.
3. J. W. Lloyd . Foundation of logic programming. Springer 1987
4. H. Prade, G. Richard and Mathieu Serrurier. Enriching relational Learning with fuzzy predicates. In. Proceedings of PKDD 2003, N. Lavrač et al. (eds), Springer 2003, ISBN 3-540-20085-1, 399-410.
5. H. Prade, G. Richard, D. Dubois, T. Sudkamp and Mathieu Serrurier. Learning first order fuzzy rules with their implication operator. In. Proceedings of IPMU 2004.
6. D. Shibata et al. An induction algorithm based on fuzzy logic programming. In Proc. PAKDD’99, Ning Zhong, Lizhu Zhou (eds.), LNCS 1574, Springer 1999, 268-273.
7. F. Riguzzi. Learning Logic Programs with Annotated Disjunctions. In. Proceedings of ILP 2004, R. Camacho et al. (eds), Springer 2004, ISBN 3-540-22941-8, 270-287.
8. P. Vojtáš. Fuzzy logic programming. Fuzzy sets and Systems *124 (2001),* 361-370
9. Vojtáš P, Horváth T, Krajči S, Lencses R (2004) An ILP model for a monotone graded classification problem. Kybernetika 40, (2004), AV ČR, 317 – 332.

Effects of Selected Basic Parameters and Data Features on Text Categorization by SVM

Tomáš Hudík and Jan Žižka

Faculty of Informatics, Department of Information Technologies
Masaryk University, Botanicka 68a, 602 00 Brno, Czech Republic
{xhudik,zizka}@informatics.muni.cz

Abstract. This paper describes results acquired from testing influences of selected important parameters of Support Vector Machines (SVM) applied to text categorization. The main object was to verify whether results obtained with standard datasets could be applied to real medical text documents. The research also focused on features as document similarity, category balance, presence of common words (stop-words), and data volume. The experimental results demonstrated that there could be typical problems with setting up parameters for some real data. As a result, SVM could not always find sufficiently good separating hyperplanes as it mostly did for 'trouble-free' datasets like Reuters or 20Newsgroups.

1 Introduction

The aim of the research was to experimentally reveal effects of selected basic algorithm parameters and data features on text categorization by Support Vector Machines (SVM). The experiments with unstructured text documents from various Internet resources tried to find out what could be the best possible way to classify and categorize large amounts of text articles as well as which results can be expected from the trained classifiers. Generally, the results were quite acceptable, however, there were cases when even such an effective algorithm as SVM could not provide acceptable outcomes. Obviously, not only the SVM parameters are important; attributes of text data play a significant role, too.

2 Text Categorization by Support Vector Machines

The categorization process of text documents is a function $\Phi : D \times C \rightarrow \{T, F\}$, where Φ assigns T (true) or F (false) to every document $d_i \in D$ and every category $c_i \in C$—if d_i belongs to c_j the result of Φ is T else F. The sets D and C can be defined in various ways, e.g., the document d_j can belong to exactly one or more categories C, etc. One of the text categorization problems is converting documents into an n-dimensional space needed for their further processing by the *Support Vector Machines* (SVM) algorithm [1]. More possibilities exist, see [6], but in most cases the *Bag Of Words* (BOW) method is usually employed. BOW represents the input texts in an n-dimensional space, where n

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 210–217, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

stands for the number of *attributes* in all text documents, i.e., *words* or their *stems* that help avoid redundant word differences, e.g., *word, words, wording, wordy, wordily, wordiness, wordlessly*, and so like, to decrease the dimensionality. In addition, it is sometimes useful to eliminate stop-words[1]. The BOW's output can be a set of vectors where every member represents one document. The SVM algorithm [8] is relatively fast and resistent to overfitting and local extremes, and does not suffer from problems with a large number of dimensions. SVM is a general classifier that tries to find a *hyperplane* between two distinct classes of objects. The main problem with SVM is often its linearity because most of real domains are inherently nonlinear. This problem solve *kernel functions* that increase the number of dimensions of a particular task, thus supporting finding a linear solution. SVM is in details described in, e.g., [1] and [9].

3 Datasets Used in Experiments

The experiments employed three various datasets. The first one was 20Newsgroups [3, 10]—a set of 20 Internet topics where each newsgroup includes 1000 contributions[2]. The second, very special medical dataset [7, 11], was collected by an expert physician. These documents—representing a difficult real classification case—are divided into two groups: interesting and uninteresting from the physician's viewpoint. Positive instances ($\oplus$) include 191 interesting articles while the negative ones ($\ominus$) contain 40 uninteresting documents. The third area of experiments employed the Reuters-21578 dataset [3, 12]. The original Reuters documents are divided by topics into 23 categories. The number of articles in each particular category is different; the articles have generally various numbers of words. Here, the experiments worked with only 10 categories having the highest number of articles. Reuters and 20Newsgroups (and their various modifications) often serve as standard benchmarks [6].

Each topic in 20Newsgroups was separated to make an individual set. Then, a dictionary for each of the 20 topics was generated using stems of words. For the binary classification, it was necessary to join each pair of dictionaries (190 topic combinations). Feature spaces were generated using selected kernel functions with pairs of topic-dictionaries and stop-words. Every attribute space had 2000 articles which were divided into a training (1500 articles) and testing set (500 articles). The training examples were used for learning by the program SVM^{light} [5] (see `http://www.joachims.org`). Only the linear kernel was employed because other kernels took much more time and it was generally not clear which kernel function could be the best one for the given task. Then, the testing process provided accuracy, precision, and recall [6] on the appropriate test sets.

For the much smaller medical data with two categories, the dictionary and the attribute space were created similarly like for 20Newsgroups. The attribute space was divided 50% for the training and the rest for the testing phase.

[1] mostly common words like $a, an, the, of, in, me, you, to, is, it, on, \ldots$, or words which are used just once in all the text documents, and so like

[2] except for *soc.religion.christian* with 997 contributions

In the case of Reuters-21578, each topic category was tested in comparison with each other category by binary classification again similarly like for 20Newsgroups. Unlike 20Newsgroups, Reuters-21578 does not have the same number of articles in each category, so it was necessary to choose appropriate settings of numbers of examples for training and testing. The number of training examples was set as one half of articles taken from a category which had fewer articles and (randomly) the same number of articles from the other category. The same number of examples from both categories was used for the testing phase. Additional tests were carried out to reveal how the classification would behave when a particular category was compared with all others. Successively, each particular category was used like the positive one and other groups as the negative ones.

4 Results of Experiments

20Newsgroups The training and testing processes were carried out by the 20-times random selection for each pair of newsgroups (190 pairs). In addition, to get even better prediction of classifiers, experiments with 200-times random selection were made as well. Full results are out of the scope of this paper but it can be found in [4]. Here, only the most interesting results are shown. The worst accuracy of classification provided the group *talk.religion.misc* against *alt.atheism*—just 85.82% because the topics were rather similar which usually decreases the classifier efficiency. The best accuracy had *comp.sys.mac.hardware* against *rec.sport.baseball*; neither had its accuracy worse than 100% for all of the 200 random selections because the topics were quite different.

Medical documents 20-times random selections provided seemingly very good results: the accuracy was 82.76%. However, the new problem was that SVM could not recognize the negative examples. The reason was that 50% of the real data examples were used from interesting articles and the rest from uninteresting ones: 95 $\oplus$ and 20 $\ominus$. When trying to make the same number of $\oplus$ and $\ominus$ (e.g., 20 $\oplus$ and 20 $\ominus$), the accuracy was 52.98%, but it could recognize both the $\ominus$ and $\oplus$ class. The same effect was obtained by setting up the weight of positive examples[3]. Then, various ratios between both classes were tested. If the ratio was 33 $\oplus$ to 30 $\ominus$ using 10-times random selection, in 8 tests the recognition of $\ominus$ was again very low (between 10%-40%). Consequently, here it is possible to see that SVM is sensitive to the ratio between the classification classes. To increase the accuracy, two ways are possible. The first one is using stop-words. An appropriate selection of stop-words can sometimes dramatically influence the accuracy: changing the number of stop-words means changing the number of irrelevant features. The second one, which is often used for decreasing the number of irrelevant features, is the following: a word is a feature only if its occurrence in a particular article is at least n. The results of experiments are demonstrated in Table 1 for 20 random selections. Here, 300 stop-words and $n = 1$ provided the

[3] the weight parameter gives the ratio between $\oplus$ and $\ominus$; in our case it was 0.2 because the number of $\ominus$ was around 20% of $\oplus$

Table 1. Text classification with various numbers of stop-words and word occurrences in a particular article. The upper index means *in how many tests the accuracy of one of the classes was lower than 50%*

stopword #	word occurrence # (n)			winner
	1	2	3	
0	$61.1\%^{7}$	$59.02\%^{12}$	$52.5\%^{11}$	$\mathbf{61.1\%^{7}}$
100	$74.78\%^{15}$	$71.84\%^{16}$	$60.13\%^{12}$	$60.13\%^{12}$
200	$71.72\%^{14}$	$72.5\%^{15}$	$60.37\%^{10}$	$60.37\%^{10}$
300	$74.13\%^{14}$	$73.04\%^{16}$	$54.4\%^{13}$	$74.13\%^{14}$

highest accuracy, however, in 14 tests from 20 the classifier could not recognize one of the classification classes. Therefore, this solution was quite poor—the recognition of both classes must be as good as possible, otherwise the results would not be representative enough and they would never be better than the baseline value[4]. However, it is also necessary to take into consideration that the available medical data suffered from a relatively small volume (191 $\oplus$ and 40 $\ominus$) plus a rather high word-similarity between $\oplus$ and $\ominus$, which negatively influenced the results. The table also shows that the results were rather unstable, with only one typical trend: the higher n the lower the accuracy. Interestingly, only one case provided a number of wrong recognitions lower than 10 (for the stop-word number 0 and $n = 1$). This value was taken as the solution. The table also depicts that the feature reduction had no significant impact on the accuracy.

Because of not very good results obtained from the standard experiments with the linear kernel, additional experiments were carried out to see if employing other kernel functions could bring better outcomes. The work was based on the described solutions, using no stop-words and $n = 1$. Table 2 shows the main results. In the first two attempts, a *linear kernel* with a different parameter c (a trade-off between a training error and a margin) was used. Changing values of c increased the accuracy, however, the number of tests in which SVM could not correctly recognize one of the classes increased up to 18. The other attempts tried a *polynomial kernel* that was not good as well. It was not possible to find out the appropriate settings of SVM to recognize the positive articles (17.39% actually means that SVM recognized all $\ominus$ and no $\oplus$). Twenty wrong tests means that in all iterations SVM could find no correct class. The following kernel was a *radial basic function* with only one adjustable power parameter γ (g in Table 2). Firstly, the number of bad tests was maximum (20), but after decreasing g the number of wrong tests was lower. Then the second parameter c was added, which improved the results up to 70.1% with only 4 wrong tests. Other settings of the kernel did not improve the results. For this kernel, it is possible to see how important is a good setting: with the implicit settings, it recognized just negative articles, but after a certain parameter modification, it gave quite acceptable solutions.

[4] the ratio between the largest class and all the examples; in our case it was 82.7% because $\frac{191}{(191+40)} = 0.827$

Table 2. Looking for the most appropriate kernel. The last column is the number of iterations (from 20) when one of both classes was not recognized correctly (i.e., less than 50% of examples were classified correctly)

kernel	parameters	accuracy %	# of wrong tests
linear	-c 0	60.45	7
linear	-c 0.1	79.56	18
polynomial		17.39	20
polynomial	-d 2	17.39	20
polynomial	-d 2.5	55.07	20
polynomial	-d 2.5 -r 1 -s 2	48.5	15
RBF		17.39	20
RBF	-g 2	17.39	20
RBF	-g 0.02	35.2	20
RBF	-g 0.000002	59.3	8
RBF	-g 0.00000001	62.42	5
RBF	-g 0.0001	61.47	6
RBF	-g 0.00001 -c 0	62.6	7
RBF	-g 0.00001 -c 100	73.9	18
RBF	**-g 0.00001 -c 60**	**70.1**	**4**
RBF	-g 0.00001 -c 50	65.4	4
RBF	-g 0.00001 -c 40	55.76	8
sigmoidal	*-j 0.2*	17.39	20 $\oplus$
sigmoidal	*-j 0.21*	82.62	20 $\ominus$
sigmoidal	-r 1 -s 1	17.39	20
sigmoidal	-r 9 -s 1000	17.39	20

The last kernel was a *sigmoidal function*. This kernel was completely unsuitable; it was not possible to find out any setting of the kernel so that the number of bad tests would be lower than the maximum (20). In one interesting experiment (the first *sigmoidal* line in the table), the weight j of positive examples was set to 0.2. In this case SVM recognized only the negative examples (wrong 20 $\oplus$). In the second line, j was set to 0.21 and SVM recognized only positive examples (wrong 20 $\ominus$). This is an evidence that SVM could be sometimes very unstable.

Reuters-21578 The experimental results are demonstrated in Table 3. Generally, these results are good. Only in two[5] cases SVM had problems with recognizing the classes. However, a closer look at the categories—*wheat*, *corn* against *grain*—explains it: the categories are rather similar overlapping topics and it is also not easy for people to decide to which category belongs a particular article. In this table, the categories are organized by the number of articles, i.e., *earn* had the highest number of articles and *corn* the lowest number. Generally, in the top left corner of the table, there are better results than in the bottom right corner.

[5] in Table 3 there are four cases because it is a symmetrical matrix

Table 3. The accuracy of the binary classification for the Reuters dataset in 10 categories. The emphasized number means that SVM could not recognize at least one of the classes. The upper index says *in how many tests (from 30) this classification accuracy was very low*

%	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn
earn		97.42	98.89	99.09	97.91	98.69	99	96.77	98.67	97.59
acq	97.42		98.96	98.29	95.76	97.73	99.23	95.03	99.04	97.47
money-fx	98.89	98.96		99.25	98.81	91.98	73.16	98.19	99.26	98.88
grain	99.09	98.29	99.25		98.24	93.78	99.24	90.96	*60.16*^{14}	*58.41*^{24}
crude	97.91	95.76	98.81	98.24		97.66	98.86	85.59	98.78	97.95
trade	98.69	98.14	91.98	93.78	97.66		96.23	95.13	95.47	94.3
interest	99	99.23	73.16	99.24	98.86	96.23		98.61	99.2	98.79
ship	96.77	97.73	98.19	90.96	85.59	95.13	98.61		95.75	93.88
wheat	98.67	99.04	99.26	*60.16*^{14}	98.78	95.47	99.2	95.75		75.3
corn	97.59	97.47	98.88	*58.41*^{24}	97.95	94.3	98.79	93.88	75.3	

One category against others The random selection used 100 iterations because the number of Reuters $\ominus$ was always very high in comparison with $\oplus$. The results of testing are in Fig. 1 and Fig. 2. Firstly, the experiments tried to explore if the *length* of an article was important. The categories were arranged by their average lengths of articles as Fig. 1 shows. Obviously, the number of words had no impact on the Reuters classification task because there was no significant trend (e.g., a better accuracy for longer articles). Secondly, the following attempts tried to explore the impact of the number of articles on the accuracy. The topics were arranged by the number of the articles. Fig. 2 shows the results: this criterion is obviously important. The higher is the number of articles in a particular topic group the higher is also the classification accuracy and the extent of diffusion is lower. Five groups (*money-fx*, *grain*, *crude*, *trade*, and *interest*) had similar counts of articles and relatively similar accuracies. The set *interest* had the highest accuracy because there was no other group with a similar topic. On the other side, the set *grain* had a relatively high number of articles but the accuracy was lower; as it was mentioned above, three groups (*grain*, *wheat*, and *corn*) had articles with very similar topics.

5 Conclusions

This research examined text classification using the Support Vector Machines (SVM) algorithm with three different data sets. The first dataset was 20Newsgroups, the second one contained expert medical data having two classes, and the last one was the standard Reuters dataset. For the newsgroups, the experiments reached very good results—only in one case the accuracy was 85.8%, in other cases it was around 99%. For the medical data, the best appropriate kernel provided the accuracy 70.1% that was lower than the baseline 82.76%. With the Reuters dataset, the results were again very good, even if there were two cases

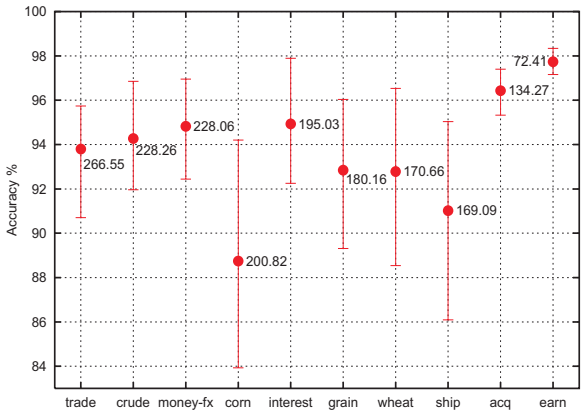


Fig. 1. The article length: Accuracy and its diffusion for the Reuters data. The number near accuracy gives the average article length in a particular category

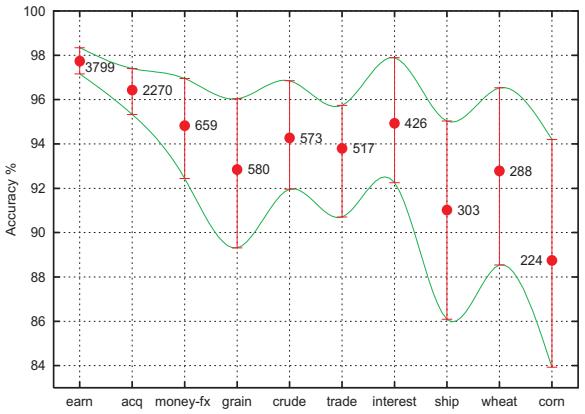


Fig. 2. The article number: Accuracy and its diffusion for the Reuters data. The number near accuracy gives the number of articles in a particular category

in which SVM could not provide quite satisfactory recognition of the classes. Different results were obtained because in the categorization of the newsgroups and of the Reuters dataset each topic group included more different words in more different topic.

The classification of the medical texts was another thing. The experiments had to delimit articles in one expert topic. In all the articles both from the negative and positive class, there were many identical words. The separating criterion was subjectively given by the opinion of one medical expert. SVM could not create an appropriate attribute space as the same words were more-or-less equally used in interesting and uninteresting articles. Another big difference was in the number of the training instances—in the case of the newsgroups, there were 500 examples from one group and 500 examples from the other. However, in

the medical data, just 64 training instances were actually available to obtain the balanced ratio between two classes (32 interesting and 32 uninteresting because the negative class included much lower number of instances than the positive one). If there are only a few training instances, like in this case, noise plays also a big role. In the Reuters data, there were also different counts of the training examples but the groups with the higher number of articles had also higher accuracy. Interestingly, in this case the length of the articles was not important.

Therefore, the conclusion is: having enough training instances which are different enough and balanced enough, it is possible to obtain good classification results with low errors. The problem is what does the word *enough* actually mean. This term is rather fuzzy and its values can change from case to case. Of course, these parameters co-operate together in some cases—if we have different topics of the classes (they have a bigger distance from each other) we do not need so many training instances and vice versa.

Acknowledgments

This research was partly supported by the Grant *Human-Computer Interaction, Dialog Systems and Assistive Technologies, MSM*-143300003.

References

1. Cristianini, N. and Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000
2. Cristianini, N., and Schölkopf, B.: Support Vector Machines and Kernel Methods The New Generation of Learning Machines, AI Magazine, American Association for Artifcial Inteligence, 2002, pp. 31-41
3. Godbole, S.: Exploiting Confusion Matrices for Automatic Generation of Topic Hierarchies and Scaling up Multi-way Classifiers. Annual Progress Report, Indian Institute of Technology, Bombay, January 2002, 17 pages
4. Hudík, T.: The Application of the Machine Learning Algorithm Support Vector Machines to Information Processing. Diploma Thesis, Faculty of Informatics, Masaryk University, Brno, 2004 (`www.fi.muni.cz/~xhudik/dipl.pdf.zip`)
5. Joachims, T.: Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. B. Schölkopf, C. Burges, and A. Smola (Eds.), MIT-Press, 1999
6. Sebastiani, F.: Machine Learning in Automated Text Categorisation, ACM Computing Surveys, Vol. 34, March 2002, pp. 1-47
7. Žižka, J., Bourek, A., and Frey, L.: TEA: A Text Analysis Tool for the Intelligent Text Document Filtering. In: Proc. of The Second International Conference on Text, Speech and Dialogue TSD-2000, September 2000, Brno, Czech Republic. Springer Verlag, 2000, LNCS 1902, pp. 151-156
8. `http://portal.acm.org/citation.cfm?id=130401`
9. `http://www.kernel-machines.org`
10. `http://www.ai.mit.edu/jrennie/20Newsgroups/`
11. `http://www.fi.muni.cz/~zizka/medocs/`
12. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

Využití vazeb mezi termy pro podporu uživatele WWW[2]

Jiří Jelínek

1Katedra managementu informací, FM, Vysoká škola ekonomická, Jarošovská 1117/II, 377 01, Jindřichův Hradec

jelinek@fm.vse.cz

Abstrakt. Problémem současného Internetu je absence sémantického popisu WWW stránek a navazujících struktur, které by mohly být využity pro efektivní vyhledávání. Změnu jistě přinese konstrukce stránek založená na sémantických nástrojích jako např. jazyce RDF, v současnosti je však nutné získávat podklady pro sémantické zpracování doplňkovými postupy. Příspěvek se zabývá některými z nich.

Systém efektivně podporující uživatele WWW při jeho práci na síti by měl mít mimo jiné schopnost pracovat s termy, které se vyskytují na stránkách navštívených uživatelem. Ty totiž mohou být uživateli nabídnuty jako další klíčová slova či přímo použity pro expanzi dotazů. Aby mohl být tento cíl splněn, je potřeba se kromě detekce významnosti jednotlivých termů zabývat též analýzou vzájemných vazeb termů mezi sebou a to jak v množině navštívených stránek, tak případně i mimo ni.

Příspěvek uvádí několik způsobů zpracování vazeb mezi termy, které jsou dále značně modifikovatelné a vycházejí z reprezentace těchto vazeb ve formě asociačních pravidel a grafických struktur, které mohou z těchto pravidel vycházet. Jejich hlavním cílem pak je odhalení skupin termů majících silnou vzájemnou vazbu a tedy vztahujících se k jednomu tématu či oblasti zkoumanými uživatelem. Diskutovány budou rovněž možnosti užití již existujících ontologických struktur jako doplňkové báze dat pro uvedenou analýzu.

Klíčová slova: web mining, web content mining, podpora uživatelů, graph mining, shlukování, ontologie

1 Úvod

Internet a jeho nejpopulárnější služba WWW daly uživatelům k dispozici obrovské množství dat a informací. Problémem se však stává ne jejich technická dostupnost, ale schopnost uživatele danou informaci objevit.

Systém efektivně podporující uživatele WWW při jeho práci na síti by měl mít mimo jiné schopnost pracovat s termy, které se vyskytují na navštívených stránkách. Ty totiž mohou být uživateli nabídnuty jako další klíčová slova či přímo použity pro expanzi dotazů s pomocí některé z vyhledávacích služeb.

Základním problémem dneška je bezesporu způsob získávání sémantických dat, zvláště pokud se navíc zájmy uživatele a tedy i zkoumané domény mění. Příspěvek se

[2] Příspěvek vznikl za podpory grantu č. 34/04 Interní grantové agentury VŠE Praha.

zaměřuje na vzájemné vazby mezi termy a uvádí několik způsobů jejich zpracování, které vycházejí z reprezentace těchto vazeb ve formě asociačních pravidel a grafických struktur na nich založených. Jejich hlavním cílem pak je odhalení skupin termů majících silnou vzájemnou vazbu a tedy vztahujících se k jednomu tématu či oblasti zkoumanými uživatelem.

Příspěvek vychází z širšího projektu podpory uživatele v prostředí WWW, který je rozvíjen na Katedře managementu informací Fakulty managementu VŠE v Jindřichově Hradci [5].

2 Shrnutí současného stavu

Problém uživatelské podpory v prostředí WWW, založené především na technikách Web Miningu (dále jen WM) je dnes velmi aktuální otázkou, kterou se zabývá značný počet jednotlivců i celých pracovišť. Příkladem mohou být např. práce [1, 3, 4]. Podpůrné systémy prakticky vždy vycházejí z údajů o chování a zájmech uživatele shromážděných v minulosti.

V tomto příspěvku se soustředíme především na postupy spadající do oblasti Web Content Mining (WCM), které se zaměřují na práci s obsahem WWW stránek. V popisovaných metodách se uplatní i klasické metody umělé inteligence a operačního výzkumu, především nástroje shlukové analýzy, principy tvorby asociačních pravidel a metody pro práci s reprezentací dat v podobě grafů.

Základem technik WCM jsou velmi často metody určené pro zpracování dokumentů. Obvykle se pracuje s *termy* jako základními popisnými jednotkami. Nejčastěji se používají metody využívající vektorovou reprezentaci dokumentů [6].

Významnou oblastí výzkumu je výběr relevantních termů [1] a odhalování vzájemných významových vazeb mezi nimi pro konstrukci pojmových struktur. Relevance termů je často poměřována tzv. *tf-idf* vahou založenou na frekvenci výskytu termů v jednotlivých dokumentech a inverzní frekvenci výskytu dokumentů s daným termem v celém souboru dokumentů [9]. Pro vyhledávání významových vazeb může být zajímavou cestou využití pravděpodobnostních přístupů, např. asociačních pravidel a bayesovských pravděpodobnostních sítí.

Automatická extrakce pojmových struktur nad danou množinou textů je velmi složitý problém, který zatím nebyl úspěšně obecně vyřešen. Finální zásah experta pro samotné zařazení pojmu se však zatím jeví jako nutný. Popisované přístupy se snaží tvorbu a detekci pojmových struktur podpořit a alespoň částečně zautomatizovat.

3 Extrakce a analýza vazeb mezi termy

Pro účely podpory uživatele by bylo velmi vhodné mít k dispozici dílčí pojmovou strukturu založenou na množině termů z navštívených stránek, která by mohla být využita pro lepší navigaci uživatele. Níže uvedené postupy ukazují několik cest, jak podobné struktury získat, přičemž důraz je kladen především na detekci významových vazeb mezi jednotlivými termy.

3.1 Příprava dat

Uváděné přístupy vycházejí z množiny dokumentů (např. WWW stránek navštívených v jednom sezení), jejichž plné texty jsou k dispozici. Na této množině je následně realizován výběr významných termů. Ten je v současnosti prováděn na základě výpočtu upravené *tf-idf* váhy pro všechna slova v množině dokumentů podle následujícího vzorce:

$$w_t = \frac{1}{n_t} \sum_{i=1}^{N} c_{ti} \,, \qquad (1)$$

kde w_t je upravená *tf-idf* váha termu t, n_t počet dokumentů, ve kterých se daný term vyskytuje, N je celkový počet dokumentů a c_{ti} počet výskytů termu t v dokumentu i. Termy s $w_t > práh$ pak tvoří množinu významných termů V_g, která slouží za základ dalšímu postupu. Hodnota prahu je údajem, který zadává uživatel především podle požadovaného počtu termů v množině V_g. Každý term v uvedené množině je definován vektorem $T_t = (r_t, n_t, c_t, w_t)$, kde r_t je název termu (slovo) a c_t součet všech výskytů slova v dané množině dokumentů (viz suma ve vzorci (1)).

Množina termů V_g je dále „prořezána“ pomocí množiny zakázaných slov V_z definované uživatelem či správcem systému. Výsledná množina $V = V_g - V_z$ slouží jako základ pro získání grafické struktury, jejíž uzly budou zkoumané termy a vzájemné vazby mezi nimi budou zachyceny hranami grafu.

Variantním postupem, který je rovněž zkoumán, je využití prostého slovníku významných pojmů o jejichž zařazení do ontologické struktury má uživatel systému zájem. Množina V pak obsahuje právě tyto pojmy, pro něž jsou na základě dané množiny dokumentů dopočteny další souřadnice popisného vektoru T_i.

Další kroky jsou založeny na předpokladu, že významově blízké termy se v dokumentech vyskytují často společně. Proto jsou na množině V definovány dvojice termů (t_i, t_j), pro které je zjištěn počet dokumentů o_{ij}, ve kterých se oba termy vyskytují současně. Dále je použito postupů známých z oblasti tvorby asociačních pravidel. Pro každý term t_i ve dvojici s termem t_j lze vypočítat podmíněnou pravděpodobnost

$$p_{i/j} = \frac{o_{ij}}{n_j} \,, \qquad i \neq j \qquad (2)$$

Výpočtem hodnot $p_{i/j}$ je ukončena příprava dat pro následující postupy. Zde je třeba upozornit, že variantně lze pro další úvahy použít podmíněnou pravděpodobnost $p'_{i/j}$ založenou na skutečných počtech výskytu termů a nikoliv na počtu dokumentů, ve kterých se dané termy vyskytovaly.

3.2 Detekce podstatných vztahů

Následující úvaha vychází z podmíněných pravděpodobností vypočtených podle vztahu (2) pro dva zvolené termy. Pro vytvoření grafické struktury obsahující podstatné vazby mezi termy je zavedena mezní hodnota $m \in \langle 0,1 \rangle$, která je použita pro odlišení různých vztahů mezi termy. Rozlišované typy vazeb jsou následující:

- $(p_{i/j} > m) \wedge (p_{j/i} < m)$ - lze usuzovat, že term t_i se vyskytuje ve větším počtu dokumentů, než term t_j. Z hlediska vzájemného vztahu je tedy term t_i termem obecnějším, častěji používaným.
- $(p_{i/j} < m) \wedge (p_{j/i} > m)$ - v tomto případě se term t_i naopak vyskytuje v omezeném počtu dokumentů a to výrazně menším, než term t_j. Z hlediska vzájemného vztahu je tedy term t_i termem specializovanějším.
- $(p_{i/j} > m) \wedge (p_{j/i} > m)$ - termy t_i a t_j se vyskytují často právě společně a lze tedy usuzovat, že jejich vzájemný vztah je vyvážený nebo rovnocenný.
- $(p_{i/j} < m) \wedge (p_{j/i} < m)$ - tato situace jasně ukazuje na malou vazbu mezi termy t_i a t_j. Jejich společný výskyt je tedy spíše náhodný.

Ve vytvářené grafické struktuře jsou pak pouze hrany, které splňují podmínky uvedené v prvních třech případech. Na základě odlišení těchto tří typů vazeb je možné transformovat neorientovaný graf na graf orientovaný. Hrany definované mezi termy a orientované podle výše uvedených vztahů mohou pak posloužit k získání množiny obecnějších (nadřízených), konkrétnějších (podřízených) a podobných (rovnocenných) termů.

Při užití popisovaného postupu je však potřeba mít stále na paměti skutečnost, že získaný orientovaný graf není zachycením pouze významových vztahů. Obsahuje i mnoho dalších vazeb, které není možné využít při tvorbě doménové pojmové struktury. V této fázi je stále nutné využití experta, který bude schopen vyloučit nadbytečné a nepotřebné vztahy, kterých však, díky některé z popsaných metod výběru termů do množiny V, nemusí být mnoho.

Při experimentech popisovaných v tomto bodu byla také zkoumána možnost využití stávajících ontologií a případná podobnost získaných grafických struktur. Pro tento účel byla zvolena ontologie WordNet [11]. Její část obsahující podstatná jména byla nejprve převedena do grafické reprezentace. Snahou bylo porovnat tuto strukturu s výstupy popsaného algoritmu, a to jak z hlediska zpracovávaných termů, tak z hlediska detekovaných vazeb mezi nimi. Bohužel je nutné konstatovat, že tento experiment neměl příliš dobré výsledky, obě množiny obsahovaly jen malé množství shodných termů. Tento stav je možno vysvětlit dvěma způsoby:

- Nástroje pro přípravu dat pro popsaný algoritmus nezachytily všechny podstatné termy. Bude nutné dále zlepšit přípravu dat a důsledně provádět lematizaci termů.
- Popisovaný algoritmus byl testován na cca 10 000 WWW stránkách se zaměřením na web mining, data mining a analýzu dat. Jednalo se tedy o

specifickou oblast se specifickou doménovou ontologií. Uživatel WWW se však obvykle zabývá takovouto specifickou oblastí. Oproti tomu WordNet je typ upper level ontologie, takže danou oblast detailně nepokrývá.

Je tedy otevřenou otázkou vyžadující další testování, zda ontologie typu WordNet bude pro popisované účely vůbec využitelná.

3.3 Detekce silně propojených podgrafů

Asociační pravidla získaná na základě vzorce (2) lze také použít pro konstrukci pravděpodobnostní sítě zachycující vazby mezi jednotlivými termy. Jedná se o orientovaný graf, kde uzly reprezentují termy a ohodnocené hrany jejich vztahy. Vazba od termu t_j k termu t_i byla definována jako podmíněná pravděpodobnost výskytu termu $p_{i/j}$ podle vztahu (2).

Pro detekci silně propojených podgrafů na takto získané grafické struktuře byl definován níže popsaný základní algoritmus, jehož cílem je objevit v grafu podgrafy (shluky uzlů) s větším počtem vnitřních vazeb, než je obvyklé v grafické struktuře jako celku.

Celý postup shlukování do k shluků je možné zjednodušeně formalizovat do následujících kroků:

1. Ze seznamu všech uzlů grafu vyberme jeden uzel, který se stane zárodkem prvního shluku a vytvoříme pro tento uzel seznam jeho sousedů, do kterého přidáme i uzel samotný. Každý shluk je tedy popsán množinou M_i svých prvků a množinou R_i vzniklou rozšířením množiny prvků M_i o uzly tvořící nejbližší okolí shluku.
2. Ze seznamu dosud nezařazených uzlů (které dosud nejsou v žádné množině M_k) vybereme jeden a vytvoříme seznam jeho sousedů.
3. Pro všechny existující shluky porovnáme množiny R_i se seznamem sousedů vybraného uzlu.
4. Vybraný uzel přidáme do toho shluku, pro který se našlo nejvíce společných prvků při porovnávání v kroku 3. Množina R_i daného shluku se aktualizuje. Pokud se v kroku 3 nenašly ani pro jeden shluk společné prvky, je počet shluků zvýšen o jeden, jehož jediným prvkem se stává vybraný uzel.
5. Pokud stále nejsou zařazeny všechny uzly grafu, vracíme se ke kroku 2.

Z podstaty uvedeného algoritmu jasně vyplývá, že získané podgrafy nemohou obsahovat stejné uzly (termy) a musí být tedy disjunktní.

Celý naznačený postup v sobě skrývá mnoho míst k modifikaci. Jedním z nich je definice funkce pro nalezení sousedů uzlu. Nemusí např. hledat pouze bezprostřední sousedy, tedy uzly s nimiž má vybraný společnou hranu, ale i jejich následníky do libovolně nastavitelné hloubky prohledávání. Rovněž můžeme sledovat nejen následníky uzlu, ale i uzly, z nichž je na náš uzel odkazováno, tedy předchůdce. Algoritmus může být rovněž modifikován, aby respektoval ohodnocení hran a

akceptoval pouze sousedy přístupné přes hrany s hodnotou vyšší než zadaný práh. Úvahy mohou být rovněž vedeny o způsobu výběru uzlů z jejich množiny. Prozatím byl použit náhodný výběr.

Ve zkoumané grafické struktuře je nutno definovat kritérium kvality shlukování. To vychází z předpokladu existence míry vzdálenosti (nebo síly vazby) mezi dvěma uzly. Mírou kvality rozdělení je pak

$$krit = \frac{v_{inter}}{v_{intra}} \tag{3}$$

Kritérium vychází ze síly vzájemné vazby, proto se snažíme o jeho minimalizaci. Míra vnitroshlukové vazby je vypočtena jako

$$v_{intra} = \frac{1}{\sum_{l=1}^{k} m_l (m_l - 1)} \sum_{l=1}^{k} \sum_{i=1}^{m_l} \sum_{j=1, j \neq i}^{m_l} v_{ij} , \tag{4}$$

kde k je počet shluků, m_l počet uzlů ve shluku l a v_{ij} je rovno ohodnocení hrany $p_{i/j}$, pokud hrana mezi t_i a t_j existuje, jinak $v_{ij} = 0$. Výraz v_{intra} udává průměrnou hodnotu $p_{i/j}$ vnitroshlukových hran vztaženou k maximální možné hodnotě vycházející z maximálního počtu takových hran. Průměr je vypočten přes všechny shluky. Podobně výraz

$$v_{inter} = \frac{1}{k(k-1)} \sum_{x=1}^{k} \sum_{y=1, y \neq x}^{k} v_{xy} \tag{5}$$

udává průměrnou sílu vazeb mezi jednotlivými shluky. Hodnota

$$v_{xy} = \frac{1}{m_x m_y} \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} v_{ij} \tag{6}$$

udává průměrnou sílu vzájemných vazeb mezi shluky x a y opět vztaženou k maximálnímu možnému počtu vzájemných vazeb.

Popsaná definice kvalitativního kritéria shlukování je použitelná rovněž pro neorientované grafy, kde bude $v_{ij} = v_{ji}$.

Vzhledem k použitému náhodnému výběru uzlů z celkové množiny je nutné uvedený výpočet provést opakovaně, přičemž jako výstup je vybráno takové rozdělení S termů do shluků, pro které je hodnota $krit$ nejmenší.

Výstupem celého algoritmu je množina podgrafů, přičemž v každém z nich se nacházejí termy mající silný vzájemný vztah. Předmětem výzkumu zůstává, jak jednoduše reprezentovat získané shluky. Zda je např. možné vybrat jediný term pro jeden shluk a jaký postup pro jeho výběr zvolit.

4 Další postup

Navržené metody zpracování vzájemných vazeb mezi termy jsou neustále vyvíjeny a ověřovány. Další postup prací se bude zaměřovat především na eliminaci některých dále uvedených problémů.

Základní vylepšení musí být provedeno v oblasti sběru a předzpracování vstupních dat, kde se ukazuje jako velmi potřebné provést lematizaci termů. Vhodné bude se též zabývat nástroji pro generování víceslovných termů.

Výzkum bude rovněž pokračovat v oblasti metod umožňujících automatickou redukci počtu vazeb v orientovaném grafu na základě ohodnocení hran a testování algoritmů pro detekci silně propojených podgrafů.

Pozornost bude do budoucna věnována i co nejefektivnější implementaci popsaných postupů neboť zejména výpočet kriteriální funkce je časově velmi náročný.

Otevřené pole pro další experimenty je též v oblasti možného využití stávajících ontologií pro zlepšení efektivity celého systému.

5 Shrnutí

Tak jak již bylo uvedeno v úvodu článku, zde prezentované postupy mají za cíl pomoci při konstrukci systému pro podporu uživatele WWW a zaměřují se na zpracování a využití vazeb mezi podstatnými termy v množině dokumentů. Popisovaná metodika je v současné době dále rozvíjena a testována na rozsáhlé množině cca 10000 WWW stránek.

Reference

1. Chan P. K.: A non-invasive learning approach. In: Web usage Analysis and User Profiling, pp. 39-55, LNAI 1836, Springer-Verlag London, UK, 2000
2. Cycorp Makers of the Cyc Knowledge Server for artificial intelligence-based Common Sense. http://www.cyc.com/.
3. Gaul W., Schmidt-Thieme L.: Mining Web navigation path fragments. In: Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD '00), pp. 1-5, 2000, Boston, USA
4. Géry M., Haddad H.: Evaluation of web usage mining approaches for user's next request prediction. In: (Workshop on Web Information and Data Management), Proceedings of the fifth ACM international workshop on Web information and data management, pp. 74-81, ISBN: 1-58113-725-7, ACM Press New York, NY, USA, 2003
5. Jelínek J.: Podpora orientace a vyhledávání v prostředí WWW. In: Poster proceedings of the 3rd annual conference Knowledge 04, pp. 13-16, VŠB-TUO Ostrava Czech Rep., 2004

6. Koštiaĺ I.: Using Latent Semantic Indexing for Intelligent Information Retrieval. In: Proceedings of the 2nd annual conference Knowledge 03. Ostrava 2003. pp. 321-330. ISBN: 80-248-0229-3.
7. Koval R., Navrat P.: Intelligent support for information retrieval in the WWW environment. In: Advances in Databases and Information Systems, Lecture Notes in Computer Science 2435, Springer Verlag, Berlin 2002, pp. 51-64, ISSN: 302-9743.
8. Laender A. H. F., Ribeiro-Neto B. A., da Silva A. S., Teixeira J. S.: A brief survey of web data extraction tools, In: ACM SIGMOD Record, Volume 31, Issue 2 (June 2002), COLUMN: Surveys, pp. 84 – 93, 2002, ISSN:0163-5808, ACM Press, New York, NY, USA
9. Schwarz J.: Současný stav a trendy automatické indexace dokumentů. In: Proceedings of the 2nd annual conference Knowledge 03. Ostrava 2003. pp. 212-221. ISBN 80-248-0229-3.
10. Velardi P., Fabriani P., Missikoff M.: Using text processing techniques to automatically enrich a domain ontology, In: Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001, Ogunquit, Maine, USA, pp.: 270 – 284, 2001, ISBN:1-58113-377-4, ACM Press, New York, NY, USA
11. WordNet, http://www.cogsci.princeton.edu/~wn/.

Annotation:

The Use of Term Links for the WWW User Support.

The problem of Internet is the absence of semantic description of web pages. In the future new languages for semantic description will be used (i.e. RDF) but today it is necessary to find another way to obtain semantic description. This paper is a part of outputs of broader web user support project that is solved on the Department of Information Management, Faculty of Management, Univ. of Economics in Jindřichův Hradec, Czech. Rep.

One of the features of the web support system should be the ability to work with terms from visited web pages. These terms can be presented to the user as new keywords or can be directly used for search request expansion. To fulfill this goal it is necessary to deal not only with the terms but also with their links to each other on the set of visited pages or also globally.

The contribution deals with several methods of term links processing. These methods are very flexible and the bases for them are association rules derived from the link and graph structures which are based on them. The main goal is to detect groups of terms with strong links between terms in each other. We can assume that these terms are connected to one subject explored by the user. The possibilities of using existing ontologies for obtaining better results will be also discussed.

Dolování ordinálních asociačních pravidel

Filip Karel, Jiří Kléma

Katedra kybernetiky, FEL, ČVUT - České vysoké učení technické v Praze,
Technická 2, Praha 6, 166 27
{karelf1,klema}@fel.cvut.cz

Abstrakt. Asociační pravidla byla prvoplánově navržena jako nástroj pro vyhledávání vazeb mezi binárními atributy. Přestože je přechod na domény obsahující i jiné typy atributů relativně přímočarý, může při něm docházet ke ztrátě užitečné informace. To platí zejména v případě atributů, jejichž hodnoty lze uspořádat - ordinálních atributů. Rozličné způsoby jejich transformace na binární atributy mohou vést ke kombinatorické explozi a v konečném důsledku i k velkému množství nevýznamných pravidel. Článek diskutuje alternativní přístup, ve kterém cedenty nejsou tvořeny konjunkcí literálů, ale jednoduchými operacemi zachovávajícími uspořádání.

Klíčová slova: ordinální atribut, asociační pravidlo, dolování znalostí.

1 Úvod

V dosud publikované literatuře přistupují různí autoři k problematice generování asociačních pravidel s ordinálními atributy odlišně. Zavádí odlišnou terminologii i postupy. Shoda však panuje v tom, že standardní postupy, které se používají u binárních či kategoriálních atributů, jsou neefektivní s často nelogickými či nepoužitelnými výsledky. Spojité či diskrétní atributy, jejichž hodnoty lze uspořádat, s sebou přinášejí řadu specifik. Prvním důležitým faktorem je optimalizace automatické diskretizace spojitých atributů. Další otázkou je měření kvality získaných pravidel. Někteří autoři používají klasické míry jako podpora (Supp) a spolehlivost (Conf) [2], [3], [4], [8], případně je doplňují dalšími pomocnými „zajímavostními" měrami. Další autoři poukazují na to, že v případě ordinálních atributů je vhodnější použití jiných měr [5], [6], [9], [10] a [11].
Např. v [5] jsou navrženy míry založené na poloze jednotlivých bodů v prostoru a jejich vzdálenosti, v [10] jsou mírou kvality statistické ukazatele, v [9] a [11] jsou v první fázi pravidla hodnocena podobnými ukazateli jako podpora a spolehlivost, ale ve druhé fázi generování konkrétních pravidel je použita nově definovaná intenzita inklinace. My budeme k hodnocení pravidel používat podporu a spolehlivost doplněnou o kvantifikátor zdvihu (Lift).
Vlastní algoritmy generování pravidel jsou si v principu podobné. Nejdříve se naleznou vhodné intervaly hodnot ordinálních atributů a ty jsou pak využity v binárních testech tvořících základ tvorby tradičních asociačních pravidel. Odlišnosti spočívají v metodice vytváření intervalů a v určování kvality pravidel. Za nejvíce odlišný můžeme označit přístup prezentovaný v [9] a [11]. Ten je založen na myšlence mapování ordinálních kategorií na řadu za sebou následujících celých čísel

tak, že ordinální význam kategorií zůstane zachován (výška: malý → 0, střední → 1, velký → 2). Mapování plní současně roli transformace do množiny celých čísel a normalizace. Levé a pravé strany pravidel (v dalším textu označované jako cedenty) se vytvářejí tak, že se hodnoty jednotlivých atributů sčítají. Závislost cedentů se posuzuje na základě veličiny intenzity inklinace, asociace se vyhledávají pouze u závislých cedentů. Zjišťováním závislosti cedentů a identifikací oblastí zesílené asociace se redukuje prohledávaný prostor pravidel a omezuje možnost nalezení nahodilých souvislostí.

Právě popsaným postupem se inspiruje i přístup zvolený v tomto textu. Numerické atributy jsou nejprve převedeny na diskrétní ordinální. Z ordinálních atributů se operacemi sčítání a odčítání vytvářejí cedenty. Pouze u závislých cedentů jsou vyhledávny oblasti zesílených asociací, tyto oblasti jsou popsány a následně rozloženy na klasická asociační pravidla, tedy asociace mezi konjunkcemi literálů.

Diskretizace spojitých atributů není z prostorových důvodů diskutována. Kapitola 2 se zabývá vytvářením pravidel nad cedenty délky 1 (triviální cedenty). Testuje nezávislost cedentů a rekapituluje postup, jak pro závislé triviální cedenty vytvářet asociační pravidla. Klíčová kapitola 3 diskutuje vytváření netriviálních cedentů založených na více atributech. Současně zobecňuje postupy uvedené v kapitole 2. V kapitole 4 je navržený postup porovnán s tradičními metodami vytváření asociačních pravidel. Srovnání je provedeno nad reálnou doménou STULONG [12].

2 Pravidla s triviálními cedenty

Pro jednoduchost uvažujme nejprve pravidla vyjadřující vztah pouze mezi dvěma triviálními cedenty. Testování nezávislosti cedentů se redukuje na dobře známou úlohu testování nezávislosti kategoriálních atributů, použijeme χ^2 test dobré shody. Vyřazujeme vzájemně nezávislé cedenty, čímž zabráníme vytváření potenciálně nahodilých pravidel. Zkusme otestovat závislost cedentů výška a váha osoby. Na obrázku 1 je příslušná kontingenční tabulka (oba atributy jsou diskretizovány do 5 kategorií) a rozdílová tabulka skutečných a očekávaných hodnot za předpokladu nezávislosti.

skutečné		VAHA					
		1	**2**	**3**	**4**	**5**	
VYSKA	**1**	77	77	34	12	3	203
	2	55	127	102	36	6	326
	3	24	93	143	60	17	337
	4	14	72	127	96	35	344
	5	1	17	57	80	44	199
		171	386	463	284	105	1409

rozdíl		VAHA					
		1	**2**	**3**	**4**	**5**	
VYSKA	**1**	53	21	-33	-29	-12	0
	2	15	38	-5	-30	-18	0
	3	-17	1	32	-8	-8	0
	4	-28	-22	14	27	9	0
	5	-23	-38	-8	40	29	0
		0	0	0	0	0	0

Obrázek. 1. Kontingenční tabulky skutečných a rozdílových hodnot cedentů VYSKA a VAHA

Výška a váha jsou evidentně závislé ($p<<0.001$), pokračujeme tedy vyhledáváním asociačních pravidel. Vycházíme z rozdílové kontingenční tabulky, zaměřujeme se na

oblasti kladných hodnot, které odpovídají oblastem zesílené asociace. Hledáme největší obdélníky obsahující výhradně kladné hodnoty (viz. obrázek 1). Z každého obdélníku získáváme jedno pravidlo, pravidla ohodnotíme pomocí klasických měr kvality jako je podpora, spolehlivost a zdvih [2].

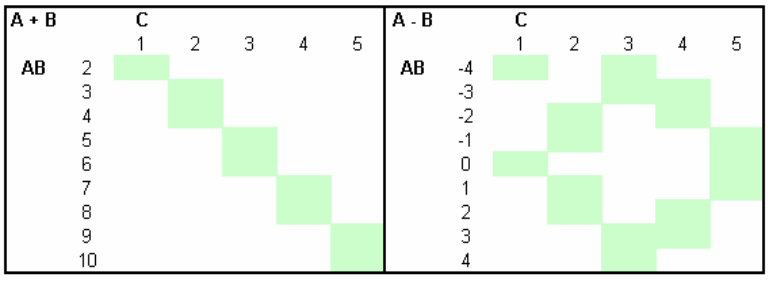
Tabulka 1 – pravidla a jejich měřítka kvality

č.	pravidlo	spolehlivost	zdvih	podpora
1	vyska = 1..2 → vaha = 1..2	0,64	1,61	0,24
2	vyska = 1..3 → vaha = 2	0,34	1,25	0,21
3	vyska = 3 → vaha = 2..3	0,70	1,16	0,17
4	vyska = 3..4 → vaha = 3	0,39	1,19	0,19
5	vyska = 4 → vaha = 3..4	0,75	1,24	0,18
6	vyska = 4..5 → vaha = 4..5	0,47	1,70	0,18

3 Generování pravidel s netriviálními cedenty

Pro cedenty jednotkové délky jde o triviální postup. Asociační pravidla však obvykle tvoříme především pro netriviální cedenty, tj. chceme kombinovat více ordinálních atributů na straně antecedentu nebo sukcedentu. Problém, který potřebujeme vyřešit, je, jak reprezentovat hodnoty více atributů hodnotou jedinou.

Řešením může být prosté sečtení nebo odečtení hodnot jednotlivých atributů. V případě dvou atributů tvořících antecedent můžeme uvažovat buď sčítání A+B nebo odčítání A-B, zbylé možnosti -A-B, B-A jsou pouhým doplňkem předchozích dvou. Jaké jsou rozdíly mezi těmito dvěmi možnostmi? Mějme následující situaci. A a B jsou ordinální diskretizované atributy, nabývají náhodně a na sobě nezávisle hodnot 1 až 5. Atribut C nabývá také hodnot 1 až 5 a je na atributech A a B závislý. V 80% hodnot je přímo úměrný hodnotě A a B. Tato závislost je lineární. Ve zbylých 20% případů je hodnota atributu C náhodná. Na obrázku 2 máme vyznačené oblasti kladných hodnot v rozdílových kontingenčních. V prvním případě získáme hodnotu antecedentu AB součtem A+B, ve druhém případě rozdílem A-B.

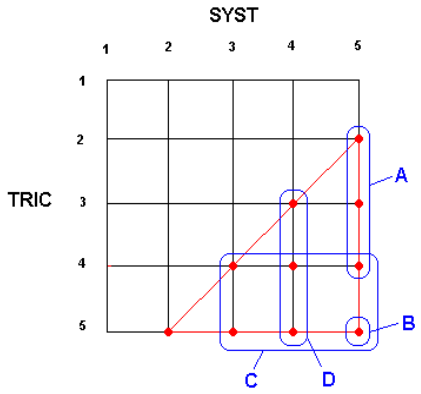


Obrázek 2. Oblasti kladných hodnot v rozdílových kontingernčních tabulkách

Z obrázků je zřejmé, že nejčitelnější a nejvýhodnější obrazec pro další generování je získán pokud je netriviální atribut utvořen podle toho jak jeho jednotlivé triviální atributy ovlivňují atribut na straně sukcedentu. A to tak, že pokud je hodnota atributu

sukcedentu přímo úměrná hodnotě atributu antecedentu, tak hodnotu ve výpočtu netriviálního atributu přičítáme a pokud je nepřímo úměrná, hodnotu odečítáme. Hodnota χ^2 je také vyšší pokud je cedent utvořen ve shodě s tím, jak ovlivňuje cedent na opačné straně pravidla. Pokud jej vytvoříme nevhodně, pak hrozí riziko, že nenalezneme existující závislost a pomineme významná pravidla. Vyvstává tedy problém optimálního vytváření cedentů. Jednoduchým řešením je hodnocení pomocí testu nezávislosti. Antecedentová kombinace, která dosáhne vyšší hodnoty χ^2, zřejmě více ovlivňuje sukcedent a je zde předpoklad nalezení „kompaktnějších" pravidel. Připomeňme, že vícenásobné testování nezávislosti k časové náročnosti celého algoritmu přispívá minimálně.

Nyní již přistupme ke generování pravidel. Budeme hledat závislost sukcedentu SUBSC (kožní řasy subscalpular) na antecedentu TRIC/SYST (kombinace kožní řasy triceps a systolického tlaku). Hodnota χ^2 je vyšší vytvoříme-li antecedent TRIC/SYST sečtením hodnot atributů TRIC a SYST než jejich odečtením. Oba antecedentové atributy nabývají po diskretizaci hodnot 1 až 5, součtový antecedent TRIC/SYST tedy nabývá hodnot 2 až 10. Aplikujeme postup uvedený v kapitole 2, pro všechny obdélníky rozdílové kontingenční tabulky zavádíme navíc dvě měřítka kvality Tc a Pp [9]. Tc je podpora daného obdélníku a Pp je poměr kladných hodnot obsažených v daném obdélníku a součtu všech kladných hodnot v celé tabulce. Z těchto obdélníků (potažmo pravidel) vyřadíme ty, které nesplňují podmínky minimální Pp a Tc. Vhodnou volbou měřítek Tc a Pp dochází k další výrazné úspoře prohledávaného prostoru při téměř nulové ztrátě generovaných pravidel a nalezených závislostí.



Obrázek 3. Vybrané obdélníky v zajímavé oblasti catributů TRIC/SYST.

Uvažujme, že jedním z pravidel získaných dosavadními kroky je i pravidlo TRIC/SYST = 7..10 → SUBSC = 5. Nyní je třeba provést dekompozici cedentu TRIC/SYST. Situaci demonstruje obrázek 3. Červeně je vyznačena oblast, která odpovídá součtu atributů TRIC + SYST = 7..10, tj. oblast odpovídající antecedentu našeho pravidla. K rozkladu musíme evidentně testovat všechny obdélníky, které lze vytvořit uvnitř inkriminované oblasti. Na obrázku 3 jsou jako příklad vybrané čtyři obdélníky. V tabulce 2 jsou pravidla, která odpovídají těmto obdélníkům i s jejich měřítky kvality. Měřítka opět slouží k eliminaci pravidel. Lze použít například

jednoduchou heuristiku, která pro každý obdélník vybírá maximálně 2 pravidla. První s maximální podporou, druhé s maximálním zdvihem (resp. spolehlivostí), oboje pouze v případě překročení prahových hodnot. V našem příkladu jde o pravidla B a C. Závislost počtu generovaných pravidel na veličinách minimální podpora, spolehlivost a zdvih je téměř lineární.

Tabulka 2 - pravidla získaná z vyznačených obdélníků A, B, C a D

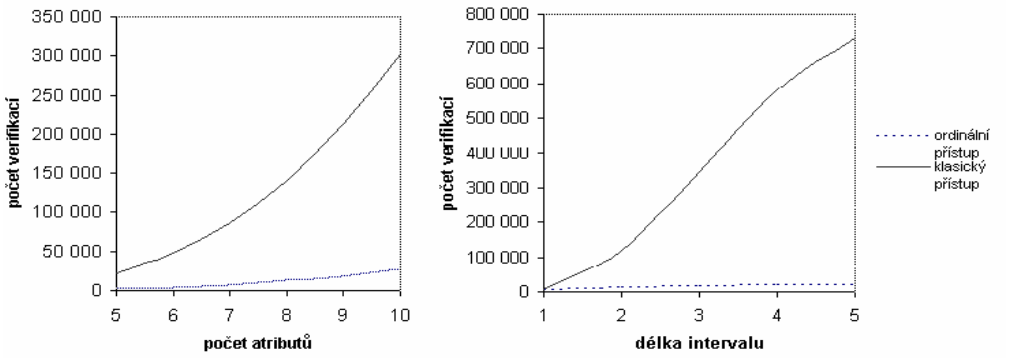
obd.	pravidlo	spolehlivost	zdvih	podpora
A	TRIC = 2..4 ∧ SYST = 5 → SUBSC = 5	0,18	1,64	0,02
B	**TRIC = 5 ∧ SYST = 5 → SUBSC = 5**	**0,60**	**2,10**	**0,09**
C	**TRIC = 4..5 ∧ SYST = 3..5 → SUBSC = 5**	**0,46**	**1,60**	**0,17**
D	TRIC = 3..5 ∧ SYST = 4 → SUBSC = 5	0,38	1,31	0,05

4 Srovnání s klasickým přístupem

V tomto odstavci srovnáme ordinální přístup s přístupem klasickým. Ve zvolené modifikaci klasický přístup využívá stejné diskretizace jako ten ordinální. Z ordinálních atributů ovšem generuje atributy binární. Používá metody intervalů, tj. binární atribut vyjadřuje příslušnost k množině sousedících kategorií tvořících interval dané délky. Parametrem je maximální délka intervalu. Pracujeme-li s atributem, jehož hodnoty jsou rozděleny do kategorií 1,..,5 a maximální délkou intervalu 2, můžeme generovat intervaly {1}, {1,2}, {2}, {2,3}, atd. Následně ověřujeme všechny jevy vyjádřitelné konjunkcí daného počtu binárních testů. Klasický přístup budeme reprezentovat systémem LISp Miner, procedurou 4ft-Miner [13]. Srovnání provedeme nad reálnou doménou STULONG. Jde o epidemiologickou studii primární prevence aterosklerózy. Reálné výsledky zaměřené na analýzu trendů využívající ordinálních asociačních pravidel byly zpracovány a prezentovány na Discovery Challenge ECML/PKDD 2004 [14]. V rámci srovnání budeme pracovat s databází obsahující 859 záznamů. Jeden záznam odpovídá jedné sledované osobě, každá osoba je popsána 10 ordinálními atributy jako např. váha, výška, tlak nebo cholesterol. Každý atribut je diskretizován do 5 kategorií.

Časovou náročnost obou přístupů budeme hodnotit na základě počtu provedených verifikací, tj. u kolika kandidátských pravidel ověřujeme splnění příslušných měr kvality. Nejprve budeme studovat nárůst počtu verifikací s rostoucím počtem atributů v databázi (levá část obrázku 4). Počet atributů nejprve uměle omezíme na 5, pak postupně budeme zvyšovat až do maximálního počtu 10. Antecedent je v tomto experimentu tvořen 2 atributy (má délku 2), sukcedent má délku 1. Délka intervalů jednotlivých atributů je menší nebo rovna 3.

Je vidět, že u klasického přístupu roste počet verifikací velmi rychle. Pro 10 atributů je počet verifikací vyšší než 300 000. U ordinálního přístupu (Pp = 0,1; Tc = 0,1) je počet verifikací přibližně 30 000. Počet nalezených pravidel je přitom řádově stejný (viz. dále). Ordinální přístup řadu verifikací eliminuje testováním nezávislosti cedentů, k dalšímu zjednodušení dochází zaměřením na oblasti zesílené asociace - obdélníky s kladnými hodnotami v kontingenční tabulce.



Obrázek 4. Závislost počtu verifikací na počtu atributů a délce intervalu.

U klasického přístupu tvoří verifikace pravidel prakticky 100% časové náročnosti. U ordinálního přístupu je tomu jinak, verifikace konkrétních pravidel tvoří od 60% (u relativně nezávislých dat) do 90% (u velmi závislých dat) celkové časové náročnosti. Zbylý čas je spotřebován na optimalizaci konstrukce cedentů, testování jejich nezávislosti a identifikaci oblastí zesílené asociace. I tak však dosahuje celková časová náročnost maximálně 15 až 20% klasického přístupu.

Výhoda ordinálního přístupu dále narůstá pokud zvyšujeme maximální povolenou délku intervalu (uvažujeme všech 10 atributů, zbylé parametry jsou nastaveny stejně jako v předchozím experimentu). Z pravé části obrázku 4 je zřejmé, že již při délce 5 je počet verifikací v klasickém přístupu až 35 násobný ve srovnání s ordinálním.

Pro větší délky intervalu je u klasického přístupu počet pravidel zhruba třikrát vyšší. Jedná se však většinou o opakující se pravidla, která nijak nepřispívající k celkovému porozumění závislostí mezi cedenty. U ordinálního přístupu stoupá počet pravidel pomalu zejména proto, že z každého obdélníku vybíráme pouze 2 nejlepší pravidla. Omezení zajišťuje, že nejsme zahlceni mnoha podobnými pravidly popisujícími stejnou nebo příbuznou asociaci.

Srovnávání počtu nalezených pravidel (asociací) však obecně nemůžeme provádět při stejných hodnotách parametrů minimální podpory, spolehlivosti a zdvihu. Jak již bylo uvedeno dříve, u ordinálního přístupu slouží tato měřítka spíše k eliminaci velmi špatných pravidel. U ordinálního přístupu při mnohem nižších měřítkách kvality dosahujeme až o 90% méně verifikací, řádově stejného počtu generovaných pravidel a stejného či vyššího počtu nalezených asociací. Z toho také vyplývá, že poměry počet pravidel / počet verifikací, počet asociací / počet verifikací a počet asociací / počet pravidel je u ordinálního přístupu mnohem vyšší.

Jak bychom nastavili jednotlivé parametry tak, abychom obdrželi přibližně stejné počty pravidel? Pokud ve zvolené doméně u klasického přístupu volíme MinSupp=0.1, MinConf=0.6 a MinLift=0.8, získáme při 212 631 verifikacích 449 pravidel postihujících 94 různých asociací (odlišnost asociací byla hodnocena subjektivně). Pokud volíme u ordinálního přístupu MinSupp=0.05, MinConf=0.4, MinLift=0.5, Pp=0.1 a Tc=0.1 získáme při 19 321 verifikacích 509 pravidel postihujících 207 různých asociací. Je tedy zřejmé, že u ordinálního přístupu lze vyhledávat i slabší asociace bez kombinatorické exploze spojené se snižováním prahových hodnot.

Riziko vynechání asociace u ordinálního přístupu hrozí pokud je sukcedent závislý na jednom ze dvou antecedentových atributů a na druhém závisí minimálně. Pokud např. platí A=vysoké & B=nízké → C=nízké a A=nízké & B=vysoké → C=vysoké, pak je součet A+B při vytváření antecedentu pokaždé stejný. Přitom C je jednou nízké a jednou vysoké. To by však znamenalo, že celý řádek v rozdílové tabulce by byl kladný. To je však nemožné, protože rozdílové tabulce musí být součty řádků a sloupců nulový. Většinou je však z ostatních získaných pravidel zřejmá klíčová role daného jednoho atributu.

5 Závěr

Článek diskutuje problematiku získávání ordinálních asociačních pravidel. Inspiruje se přístupy navrženými v literatuře a navrhuje vlastní algoritmus pro dolování asociačních pravidel v ordinálních doménách. Postup je založen na myšlence testování závislosti cedentů. Případné numerické atributy jsou nejprve převedeny na diskrétní ordinální. Z ordinálních atributů se vytvářejí cedenty jednoduchými operacemi sčítání a odčítání. Pouze u závislých cedentů jsou vyhledávny oblasti zesílených asociací, tyto oblasti jsou popsány a následně rozloženy na klasická asociační pravidla, tedy asociace mezi konjunkcemi literálů. Článek se zabývá volbou vhodného algoritmu diskretizace numerických atributů, optimalizací metod vytváření a testování závislosti cedentů a také postupy jejich následného rozkladu. Navržený postup je porovnán s tradičními metodami vytváření asociačních pravidel.
Navrhovaný postup nelze chápat jako postup čistě konkurenční k tradičním metodám dolování asociačních pravidel založených na variantách algoritmu APRIORI. Důvodem je to, že se nejedná o úplný algoritmus zaručující nalezení všech pravidel vyhovujících vstupním mírám zajímavosti pravidla. Problematická je také kombinace ordinálních a nominálních atributů majících více než 2 kategorie. Tyto vlastnosti však mohou být v řadě případů „plně" vyváženy menší složitostí algoritmu danou významně menším počtem ověřovaných asociací. Ta v důsledku vede k menšímu počtu generovaných pravidel při současném omezení jinak obvyklých podobností mezi jednotlivými pravidly. Tyto vlastnosti se mohou příznivě projevit zejména při práci s rozsáhlými databázemi, popřípadě v situacích, kdy vyhledáváme složité asociace vyjádřené kombinací většího počtu literálů.

Reference

1. Agrawal R., Imeliski T., Swami A. Mining Association Rules Between Sets of Items in Large Databases. *ACM SIGMOD Conference on Management of Data.* pp. 207-216, Washington, D.C., 1993.
2. Srikant R., Agrawal R. Mining Quantitative Association Rules in Large Relational Databases. *ACM SIGMOD Conference on Management of Data.* Montreal, Canada, 1996.

3. Fukuda T., Morimoto Y., Morishita S., Tokuyama T., Mining Optimized Association Rules for Numeric Attributes, *ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996.
4. Fukuda T., Morimoto Y., Morishita S., Tokuyama T., Data Mining Using Two-dimensional Optimized Association Rules: Schemes, Algorithms and Visualization, *ACM SIGMOD Conf. on Management of Data*, Tuscon, AZ, 1997.
5. Miller R.J., Yang Y., Association Rules over Interval Data, *ACM SIGMOD Conference on Management of Data*, Tuscon, AZ, 1997.
6. Imberman S., Domanski B., Finding Association Rules from Quantitative Data Using Data Booleanization, 1999.
7. Webb G.I., Discovering Associations with Numeric Variables, *ACM SIGMOD Conference on Management of Data*, San Francisco, CA, 2001.
8. Rastogi R. Shim K., Mining Optimized Association Rules with Categorical and Numeric Attributes, *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, 2002.
9. Guillaume S., Discovery of Ordinal Association Rules, *Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD'02), 322-327, Taipei, Taiwan, 6-8 May 2002.
10. Aumann Y., Lindell Y., A Statistical Theory for Quantitative Association Rules, *Journal of Intelligent Information Systems*, vol. 20, 255--283, 2003.
11. Guillaume S., Ordinal Association Rules towards Association Rules, *Data Warehousing and Knowledge Discovery: 5th International Conference DaWaK*, Prague, Czech Republic, 3-5 September 2003.
12. Projekt STULONG, WWW page, http://euromise.vse.cz/stulong.
13. Projekt LISp Miner, WWW page, http://lispminer.vse.cz/.
14. Kléma J., Nováková L., Karel F., Štěpánková O., Trend Analysis in Stulong Data, *In ECML/PKDD'04 workshop proceedings: A Collaborative Effort in Knowledge Discovery from Databases*, 56--67, 2004.

Annotation:

Ordinal association rules mining

Association rules have exhibited an excellent ability to identify interesting association relationships among a set of binary variables describing huge amount of transactions. Although the rules can be relatively easily generalized to other variable types, the generalization can result in a computationally expensive algorithm generating a prohibitive number of redundant rules of little significance. This danger especially applies to ordinal variables. The paper presents and verifies an alternative approach to the ordinal association rule mining. In this approach, ordinal variables are not immediately transformed into a set of binary variables. Instead, it applies simple arithmetic operations in order to construct the cedents, tests their independence and searches for areas of increased association which are finally decomposed into conjunctions of literals. This scenario outputs rules that do not syntactically differentiate from classical association rules.

Generovanie konceptuálnych popisov textových dokumentov použitím všeobecnej ontológie

Robert Kende, Slavomír Hudák

Katedra kybernetiky a umelej inteligencie, FEI, Technická Univerzita Košice,
Letná 9, 040 01, Košice

{Robert.Kende, Slavomir.Hudak}@tuke.sk

Abstrakt. Článok popisuje postup, ktorým je možné generovať konceptuálne modely textov písaných v prirodzenom jazyku. Výstup je vo forme sémantickej siete kde sú pojmy a relácie medzi pojmami indukované daným textom. Indukovaná sémantická sieť sa snaží modelovať význam textu. Hlavnou motiváciou je možnosť automatickej kvantifikácie významovej podobnosti textov v prirodzenom jazyku.

Kľúčové slová: konceptualizácia, generovanie ontológií, extrakcia znalostí.

1 Úvod

Ontológie, ako jedny z foriem vyjadrenia znalostí, sa stávajú stále viac populárnymi v oblastiach znalostného inžinierstva, systémov manažmentu znalostí, až po systémy pre spracovanie prirodzeného jazyka. Podporujú procesy vyhľadávania relevantných informácií obohatením daných informácií o tzv. *"background znalosti"*.
Manuálne prístupy vytvárania ontológií sú však príliš nákladné, časovo náročné a kladú vysoké požiadavky na ich tvorcu - experta. Expert musí mať dobré znalosti o doménovej oblasti, ktorú modeluje. Preto čoraz častejšie sa objavujú snahy o zautomatizovanie daného procesu [7], [9], [10], [12].
V nasledujúcich kapitolách príspevku bude popísaný nami navrhovaný spôsob generovania konceptuálnych popisov (doménovj ontológie) pre textové dokumenty v prirodzenom jazyku.

2 Získavanie znalostí z textu

Vo všeobecnosti, získavanie sémantických znalostí z textu je kľúčová a vo svojej podstate dosť náročná úloha. Nárast záujmu o získavanie sémantických znalostí z obrovských textových dátových množín (korpusov) vedie k vývoju nových automatizovaných metód, ktoré v sebe skĺbujú techniky z oblasti Strojového učenia (ML) a Spracovania prirodzeného jazyka (NLP). Ľubovoľný textový dokument nie je len sústavou slov, fráz a viet v zmysle syntaktickej stavby daného textu, ale je tvorený sémantickými jednotkami popisujúcimi význam jednotlivých termov. Porozumenie textu si vyžaduje lingvistické znalosti týkajúce sa morfológie slov, sémantiky (významu) slov, štruktúry viet, susednosti slov a pod. Tieto dodatočné informácie sa môžu zahrnúť do reprezentácie dokumentu a tým lepšie pochopiť jeho obsah. Medzi základné problémy, s ktorými sa stretávame pri procesoch spracovania textových dokumentov patria:

- **synonymický problém –** ten istý pojem, resp. jeho význam sa dá vyjadriť viacerými slovami.
- **polysemický problém –** jedno slovo má viacero významov.

V našom prístupe problém synoným je riešený využitím lexikónu WordNet (kap. 3). Ďalším problémom, ktorý má vplyv na kvalitu spracovania textu je problém hľadania základného tvaru slov – **stemovania.**

3 WordNet

WordNet (WN) je lexikálna databáza anglických slov a slovných spojení, ktorá sa snaží zachytávať vzájomné lexikálne/sémantické vzťahy medzi jednotlivými obsiahnutými slovami/pojmami. Jednotlivé slová slovnej zásoby, ktorú WN obsahuje sú rozdelené podľa ich slovného druhu na podstatné mená, prídavné mená, slovesá a príslovky (ostatné slovné druhy zatiaľ nie sú implementované).
Každé slovo v ľubovoľnom jazyku je možné chápať ako asociáciu jeho lexikálneho tvaru (syntaxu) a jeho významu (sémantiky). Mapovanie týchto vzájomných vzťahov je možné vyjadriť tzv. *lexikálnou maticou:*

$$E = (E_{ij})$$

v ktorej každý prvok E_{ij} vyjadruje, či lexikálny tvar F_j môže byť v príslušnom kontexte použitý pre vyjadrenie významu M_i (1-môže, 0-nemôže).
Jedným z cieľov WN je vyjadriť vzájomné vzťahy medzi pojmami M_r, M_s (tzv. sémantické relácie).
Každý pojem vo WN je reprezentovaný pomocou množiny slov, tzv. *synsetov*. Medzi ľubovoľnými dvoma slovami zo synsetu platí relácia *synonymity* R_S, ktorú je možné definovať podľa [13]:
dve slová *x* a *y* sú v relácii synonymity (sú synonymá) v danom kontexte C práve vtedy, ak substitúciou jedného z nich druhým sa v C nezmení pravdivostná hodnota.

Táto relácia je symetrická, t.z. že ak xR_Sy potom aj yR_Sx. Je nutné poznamenať, že jednotlivé synsety nehovoria, čo dané pojmy popisujú, ale vyjadrujú len ich existenciu. Príkladom takýchto synonym z WN môžu byť napr. slová {car, auto, automobile, machine, motorcar}. Celková štruktúra WN sa tak dá vyjadriť pomocou sémantickej siete, v ktorej každý uzol reprezentuje nejaké slovo, resp. pojem a hrana medzi uzlami ich vzájomný vzťah.

Medzi ďalšie základné relácie vyjadrené v databáze WN patria:

- relácia ***antonymity*** – je relácia vyjadrujúca opačný význam dvoch slov (napr. chorý/zdravý). Definícia tejto relácie je trochu zložitejšia ako v predchádzajúcej synonymickej relácii. Antonymom slova *x* je slovo, ktoré je možné vyjadriť záporom *not-x*, ale nie vždy. Napr. slová „bohatý“ a „chudobný“ sú antonymá, avšak ak povieme, že niekto „nie je bohatý“, neznamená to, že je „chudobný“.
- relácia ***hypernymity*** a ***hyponymity*** – sú sémantické, navzájom inverzné relácie vyjadrujúce vzťah abstrakcie, resp. špecifikácie medzi dvoma pojmami (napr. „strom“ je abstrakciou (hypernymum) pre „javor“ a naopak „javor“ je špecifikáciou (hyponymum) pre „strom“). Vo všeobecnosti, koncept reprezentovaný pomocou synsetu x = {x , x’ , ... } je hypernymum, resp. hyponymum pojmu reprezentovaného pomocou synsetu y = {y , y’ , ... } práve vtedy, ak je akceptovateľná veta typu: *„x je (druhom) y“* resp. *„y je (druhom) x“*. Dané relácie je možné v sémantickej sieti reprezentovať hranou typu „ISA“, pričom orientácia hrany (vzhľadom na incidujúce uzly) definuje, o ktorú z uvedených relácií sa jedná. Daná relácia je asymetrická (ak „x ISA y“ a súčasne „y ISA x“ potom x ≡ y) a tranzitívna (ak „x ISA y“ a súčasne „y ISA z“ potom „x ISA z“). Jednou z dôležitých vlastností danej relácie, ktorá vyplýva z tranzitívnosti, je dedičnosť, pri ktorej hyponymy dedia všetky vlastnosti po svojich všeobecnejších pojmoch. Na základe týchto relácií je vo WN databáze postavená napr. celá štruktúra organizovania podstatných mien, ktorá vytvára tzv. *lexikálny dedičný systém*, v ktorom najvšeobecnejšie pojmy (t.j. nemajú žiadne hypernymá) ako napr. {entity, something} ležia na vrchole hierarchie pojmov a najšpecifickejšie pojmy (t.j. nemajú žiadne hyponymá) ako napr. {buggy, roadster} ležia na najnižšej úrovni danej hierarchie.
- relácia ***meronymity*** a ***holonymity*** – sú taktiež sémantické, navzájom inverzné relácie vyjadrujúce medzi pojmami vzťah typu „časť-celok“ (napr. v zodpovedajúcom význame: „krídlo“ je časťou (meronymum) „vtáka“ a naopak „vták“ je celkom (holonymom) „krídla“). Vo všeobecnosti, koncept reprezentovaný pomocou synsetu x = {x , x’ , ... } je meronymum, resp. holonymum pojmu reprezentovaného pomocou synsetu y = {y , y’ , ... } práve vtedy, ak je akceptovateľná veta typu: *„x je časťou y“,* resp. *„x má časť y“*.

4 Metóda tvorby konceptuálneho popisu z textu

Ako už bolo spomínané, jedným z najväčších problémov s ktorým sa stretávame pri vytvorení konceptuálneho popisu textu je fakt, že mnoho slov (v angl. jazyku je to dokonca drvivá väčšina) má viac významov. Význam každého slova (slovného spojenia) v texte je silne závislý od kontextu, v ktorom dané slovo bolo použité. Pri špecifikovaní významu slova teda musíme stále uvažovať nejaký kontext daného slova. Vo všeobecnosti kontext slova môže byť uvažovaný ako:

- celý text
- kapitola
- odstavec
- veta
- „okno“ n slov

V našom prístupe definujeme kontext slova pomocou jeho okolia *n* slov.
V nasledujúcej časti je uvedená navrhnutá metóda pre generovanie konceptuálneho popisu z anglického textu.

Parametre:

o – veľkosť syntaktického okolia slova *w* (príklad *o=3, w=loves* : … John *loves* Mary …)
O – syntaktické okolie slova *w*
M – výsledná matica konceptuálneho popisu
SY – množina všetkých synsetov WordNet
OS – množina synsetov pre aktuálny kontext
w_i - slovo
S_j - synset z SY
R – sémantická relácia medzi synsetmi
prah – parameter orezania

Algoritmus:

1. Nastav okno *O* na začiatok skúmaného textu
2. Pre všetky slová w_i z *O* vyber S_j z SY, také, že w_i je slovo patriace do S_j a vlož do *OS*
3. Pre všetky S_n z *SY* nového slova w_n :
   a. Ak S_n sa v *M* nachádza inkrementuj m_{nn}
   b. Ináč, pridaj S_n do *M,* nastav $m_{nn} = 1$
   c. ak ∃ také *R*, že S_n *R* S_j, kde $S_j \in$ OS, tak inkrementuj m_{jn} a m_{nj}

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1j} & \cdots & m_{1,n-1} & m_{1,n} \\ m_{21} & m_{22} & \cdots & m_{2j} & \cdots & m_{2,n-1} & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots \\ m_{i1} & m_{i2} & \cdots & m_{ij} & \cdots & m_{i,n-1} & m_{i,n} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots & \vdots \\ m_{n-1,1} & m_{n-1,2} & \cdots & m_{n-1,j} & \cdots & m_{n-1,n-1} & m_{n-1,n} \\ m_{n,1} & m_{n,2} & \cdots & m_{n,j} & \cdots & m_{n,n-1} & m_{n,n} \end{pmatrix} \begin{matrix} \downarrow \mathbf{S_n} \\ \\ \\ \\ \\ \\ \leftarrow \mathbf{S} \end{matrix}$$

Obr. 1. Pridanie n-tého synsetu Sn do matice M

4. posuň kontextové okno
5. pokiaľ nie je koniec textu choď na krok 2.
6. normalizuj *M*
7. pre všetky prvky m_{ii} (diagonálne prvky matice *M*) kde $m_{ii} > prah$ vlož S_i a všetky S_j s príslušnými hranami pre $m_{ij} \neq 0$ do Grafu *G*

Popis jednotlivých krokov algoritmu

V 1. kroku algoritmu sa nastaví pozícia okna *O* veľkosti *o* slov na začiatok skúmaného textu. V kroku 2 algoritmu sa pre každé slovo daného okna zistí množina synsetov (čiže množina všetkých významov daného slova). Vytvorí sa matica M rozmeru *k* x *k*, kde *k* je počet všetkých rôznych synsetov zodpovedajúcich jednotlivým slovám z *O* a inkrementujú sa ich zodpovedajúce početnosti m_{ii}. Zistia sa všetky relácie medzi novopridanými synsetmi. V prípade ak existuje relácia R medzi synsetom S_i a S_j inkrementuje sa početnosť m_{ij} (krok 3 algoritmu). V nasledujúcej iterácii sa posunie kontextové okno o jednu pozíciu vpravo. Postupnými iteráciami algoritmu sa M pridávaním nových synsetov rozširuje. Po spracovaní celého textu sa získaná matica M normalizuje tak, že jej prvky sú mapované do intervalu <0,1>. Použitím prahu z matice M vytvoríme M' tak, že $m'_{ij} = 1$ ak $m_{ij} > prah$ inak $m'_{ij} = 0$. M´ definuje maticu susedností grafu G. Ak chceme uvažovať aj významnosť jednotlivých synsetov môžeme brať v úvahu hodnoty prvkov m_{ij} pre označkovanie uzlov a hrán G.

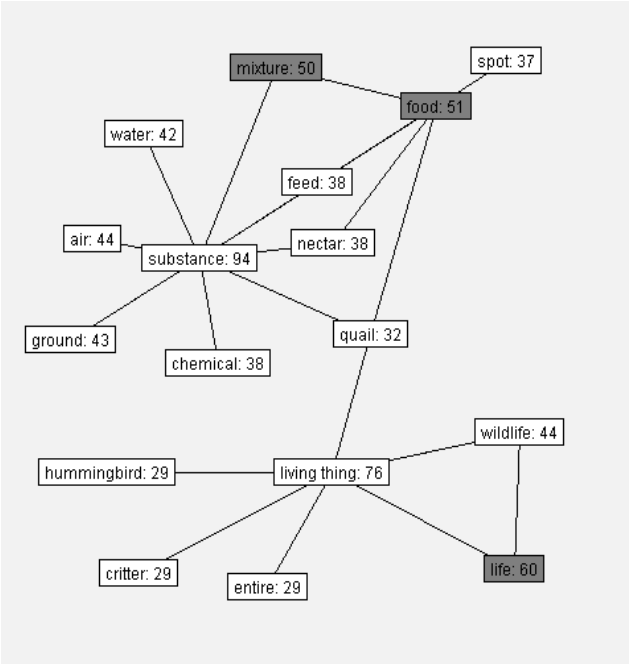
Príklad

Pre testovanie nami navrhnutej metódy konceptualizácie bol použitý korpus dokumentov Reuters-21578 [16]. Jednotlivé články boli predspracované a následne použité pre vytvorenie konceptuálneho popisu. Výslednú popis sme vyjadrili formou grafu (príklad na obr. 3).

Ukážka textu článku a vygenerovaného konceptuálneho popisu

Summer has arrived and brought along potential drought conditions and hot, smoldering days. We can help our urban and suburban wildlife by providing the essentials of life: water, food and shelter. Water is easily accomplished, with no need for an elaborately designed water garden. To attract ground-feeding birds like doves and quail, try a shallow tray, such as a 10-inch pot saucer with a few pebbles in the bottom. Place the tray on the ground in a shady spot away from potential predators. Feeding the birds during summer is often regarded as misguiding them by allowing them to give up looking for food on their own. During some droughts, however, food sources often dwindle away and a supplemental feed station might just prove to make a difference between life and starvation.

Obr. 2. Ukážka textu článku z korpusu Reuters.



Obr. 3. Konceptuálny popis pre text z obr. 2.

5 Záver

Článok popisuje metódu určenú na vytváranie konceptuálnych popisov pre textové dokumenty. Dôležitým atribútom metódy je odhad významov kandidátskych pojmov a to na základe ich kontextu ich kontextu. Konceptuálne popisy textov vygenerované touto metódou majú formu grafu pripomínajúce sémantické siete. Sémantické podobnosti pojmov v grafe sú odvádzané zo všeobecnej ontológie (v tomto prípade WordNet). Konceptuálne popisy v takejto forme sú vhodné pre definovanie metrík významovej podobnosti textov. Ak konceptuálny popis budeme chápať ako graf s označkovanými uzlami a hranami, problém sa pretransformuje na problém určovania podobnosti grafov. Pre kvantifikovanie takejto podobnosti existuje niekoľko vhodných algoritmov z teórie grafov ktorými sa článok nezaoberá. Veľmi zaujímavou vlastnosťou metódy je, že do výsledného konceptuálneho popisu sú zahŕňané aj pojmy, ktoré sa v pôvodnom texte explicitne nevyskytli. Vhodným rozšírením metódy by bolo použitie lingvistických techník na spracovanie textov v prirodzenom jazyku. Hlavným príspevkom popísanej metódy v tomto prípade by bol hlavne odhad významu pojmov z kontextu pri sémantickej analýze, ktorý je pomocou metódy možný.

6 Acknowledgement

This work is done within the VEGA project 1/1060/04 “Document classification and annotation for the Semantic web” of Scientific Grant Agency of Ministry of Education of the Slovak Republic.

7 Referencie

1. Arjona J. L., Corchuelo R, Toro M. *Automatic Extraction of Information from the Web*. Universidad de Sevilla.
2. Bagga A. *Meaning Extraction: A Quantitative Analysis*. Duke University. Durham. April 1996.
3. Bagga A. Meaning Extraction: Coreference. Cross Document Coreference and Information Extraction Methodologies. PhD Thesis. Duke University. Durham. 1998.
4. Faure D., Nédellec C. A Corpus-based Conceptual Clustering Method for Verb Frames and Ontology Acquisition.
5. Fensel D. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag. 2001.
6. Guarino N. Formal Ontology and Information Systems. Proceedings of *FOIS’98*. Trento, Italy. pp3-15. 6-8 June 1998.

7. Harabagiu S. M., Maiorano S. J. Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. Southern Methodist University.
8. Hobbs J. R. *The Generic Information Extraction System*. In Fifth Message Understanding Conference (MUC-5), pp87-91, 1993.
9. Hotho A., Maedche A., Staab S. *Ontology-based Text Document Clustering*. University of Karlsruhe.
10. Li X., Szpakowicz S., Matwin S. A WorNet-based Algorithm for Word Sense Disambiguation. Concordia University Montreal.
11. Maedche A., Pekar V., Staab S. Ontology Learning Part One-On Discovering Taxonomic Relations from the Web.
12. Miller G. A., Beckwith R., Fellbaum Ch., Gross D., Miller K. *Introduction to WordNet: An On-Line Lexical Database*. (Revised August 1993).
13. Miller G. A. Nouns in WordNet: A Lexical Inheritance System. (Revised August 1993).
14. Wiemer-Hastings P., Graesser A., C., Wiemer-Hastings K. *Inferring the Meaning of Verbs from Context.* The University of Memphis.
15. Carnegie Group, Inc. and Reuters, Ltd. Test Collection Reuters-21578. Voľne prístupne na: http://www.daviddlewis.com/resources/testcollections/reuters21578/

Genetické algoritmy pro redukci dimenze a analýzu binárních dat

Aleš Keprt, Václav Snášel

Katedra informatiky, Fakulta elektrotechniky a informatiky
VŠB Technická Univerzita Ostrava
17. listopadu 15, 708 33, Ostrava-Poruba, Ceská Republika
Ales@Keprt.cz, Vaclav.Snasel@vsb.cz

Abstrakt. Zaměříme-li se specificky na binární prostory, redukce jejich dimenze na bázi vyhledávání podstatných příznaků není vůbec snadná. Toto řešení má za cíl binární faktorová analýza (BFA), pomocí běžných algoritmů je to však velmi obtížné. Ukázalo se, že vhodným nástrojem pro tuto úlohu jsou některé typy genetických algoritmů.

Klíčová slova: Genetický algoritmus, analýza binárních dat, information retrieval

1 Úvod

Naším cílem je zkoumání binární faktorové analýzy (BFA) – specificky binární analytické metody, jejíž účel a smysl však leží v jiném typu analýzy než u dnes již rozšířených konceptuálních svazů. Jedná se o nehierarchickou nelineární analýzu čistě binárních dat, kde z principu věci není možno použít ani znalosti lineární algebry, ani matematické (funkcionální) analýzy. Experimentálně bylo zjištěno, že běžné nebinární metody, pouze doplněné o následné převedení výsledků do binární podoby, nedávají uspokojivé výsledky. Proto bylo postupně navrženo a realizováno několik nových metod, které pracují s booleovou aritmetikou na binární bázi. Byly to postupně neuronové sítě, kombinatorické hledání řešení a převod problému BFA na problém stavění konceptuálních svazů.

2 Klasická faktorová analýza

BFA vychází z klasické (čili nebinární) faktorové analýzy (FA). Ta vychází z předpokladu, že jevy, které lze pozorovat (a zaznamenat), jsou jen důsledkem skrytých **faktorů**, tedy jevů stojících v pozadí. Každá měřitelná a zaznamenatelná veličina, označována jako **proměnná**, je ve vyjádření FA lineární kombinací faktorů. Toto pojetí a základní impulz k rozvoji FA dala psychologie. Ta se totiž snaží na základě pozorování vnějšího chování jedinců určit, jakého jsou charakteru, jaké mají duševní poruchy apod. Na příkladu psychologie je idea FA jasná. Matematicky jde o snahu vyjádřit datovou matici X jako součin dvou (mnohem) menších matic $\mathsf{F} \cdot \mathsf{A}$.

$$\mathsf{X}_{[n\times p]} \approx \mathsf{F}_{[n\times m]} \cdot \mathsf{A}_{[m\times p]}$$

Sloupce matice X reprezentují p **proměnných** (variables), řádky reprezentují n **případů** (cases). Matice F, nazývaná **factor scores**, vyjadřuje stejná fakta jako matice X, ovšem pomocí faktorů, kterých má být mnohem méně než proměnných: $m << p$. Matice A vyjadřuje vztah mezi proměnnými a faktory. FA tedy provádí redukci dimenze vektorového prostoru (z p na m), opírá se přitom o poznatky statistiky a lineární algebry. Předpokladem úspěchu FA je normální rozložení (náhodné veličiny – statistický pojem). Začínáme výpočtem korelační matice, dál existuje několik numerických metod, které lze použít. Celkově je FA natolik výpočetně náročná, že v minulosti (bez výpočetní síly počítačů) nebyla moc použitelná.

Kvalitu řešení posuzujeme dle rozdílu součinu $\mathsf{F} \cdot \mathsf{A}$ oproti původní matici X. Protože obecně nelze rozložit jakoukoliv matici na součin dvou menších matic, ve výše uvedeném vzorci je použit symbol $\approx$ jako vyjádření přibližné rovnosti.

3 Binární faktorová analýza

BFA má symbolicky stejné zadání jako klasická FA. Máme binární datovou matici X a snažíme se ji vyjádřit pomocí součinu dvou menších binárních matic $\mathsf{F} \cdot \mathsf{A}$.

$$\mathsf{X} \approx \mathsf{F} \odot \mathsf{A}$$

Řádky F a A musejí být nenulové. $\odot$ je binární součin matic – odpovídá klasickému součinu, ale s použitím booleovské aritmetiky. Ta je přirozeným nástrojem pro práci s binárními hodnotami, od klasické aritmetiky se liší jen při sčítání, kde platí: $1 + 1 = 1$. Buňky binárních matic nazýváme **bity**.

Nástroje klasické FA tentokrát nelze použít, a to hned z několika důvodů. Především zde máme diskrétní prostor (na rozdíl od lineárního=vektorového prostoru), nelze vůbec hovořit o normálním rozložení (náhodné veličiny) a nemá ani smysl sestavovat korelační matici. Tím tedy padají všechny klasické metody řešení FA.

Úspěch BFA měříme pomocí chybové funkce **discrepancy**, značíme d, která je definována jako počet rozdílných bitů mezi součinem $\mathsf{F} \odot \mathsf{A}$ a původními daty X. Tato jednoduchost výpočtu d je pro nás často velkou výhodou.

$$\hat{\mathsf{X}} = \mathsf{F} \odot \mathsf{A}$$

$$d = \sum_{i=1}^{n} \sum_{j=1}^{p} |\hat{x}_{ij} - x_{ij}|$$

Cílem BFA samozřejmě je, aby hodnota d byla co nejmenší, v ideálním případě $d = 0$. Stejně jako u klasické faktorové analýzy, ani zde nelze počet faktorů m nijak rozumně spočítat, musí tedy být součástí zadání. Případně lze provést výpočet pro několik hodnot m a dle výsledku vybrat nejvhodnější variantu.

4 Genetický algoritmus pro výpočet BFA

5.1 Co je to genetický algoritmus

Genetický algoritmus je počítačová simulace, ve které jsou jedinci populace abstraktních reprezentací kandidátních řešení optimalizačního problému stochasticky vybíráni, rekombinováni, mutováni a potom odstraněni nebo ponecháni v populaci podle jejich kvality neboli vhodnosti (**fitness**), viz [3]. Tento princip simuluje chování přírody ve smyslu tzv. boje o přežití. Zákonitost přírodního výběru a přežití nejsilnějších jedinců se osvědčuje při řešení algoritmicky obtížně zvládnutelných problémů, mezi které patří i BFA. V případě BFA je nasnadě, že jedinci v populaci budou matice F a A (buď v páru, nebo jen jedna z nich – podle konkrétního návrhu algoritmu, viz níže). Tito jedinci budou pak „bojovat“ o přežití. Výsledkem by mělo být řešení BFA.

5.2 Standardní genetický algoritmus

Nelze očekávat, že by existoval jeden univerzální GA, kterým by bylo možno přímo řešit všechny problémy. Speciálně u BFA narážíme na fakt, že se pohybujeme v diskrétním prostředí, které znesnadňuje použití jakýchkoliv obvyklých postupů (jak již bylo zmíněno výše). Většina konkrétních GA vychází z původního Hollandova [2] a Goldbergova [1] algoritmu, obvykle označovaného SGA (Simple G. A., do češtiny se toto „simple“ překládá jako „standardní“).

Jedinec je v SGA reprezentován bitovým řetězcem vždy konstantní délky (čili „kusem paměti“). Každý jedinec představuje kandidátní řešení, čili jednu možnou variantu řešení daného problému. **Vhodnost jedinců** = „kvalita řešení“ je dána fitness funkcí η, která každému jedinci přiřadí nezáporné reálné číslo – hodnotu vhodnosti (nula je nejhorší). Noví jedinci jsou vytvářeni vždy ve vlnách – **generacích**. Délka života je vždy jedna generace. K vytváření nových jedinců používá SGA tři operátory:

Mutace - Je to náhodná změna bitu v řetězci.

Křížení - Dva bitové řetězce AB a CD se v náhodném místě rozdělí a spojí se do kříže (odtud pojem křížení) v nové jedince AD a CB.

Selekce - Pravděpodobnost, že jedinec bude vybrán k reprodukci = „vytvoření potomků“ je tím vyšší, čím vyšší je jeho fitness hodnota.

Na tomto místě záměrně vynecháváme hlubší detaily SGA, neboť pro námi navržené řešení BFA nejsou podstatné. Obecně platí, že úspěch SGA závisí na vlastnostech fitness funkce (spojitost a „tvar“ funkce), vhodně zvolené počáteční generaci (v našem případě vždy náhodně generovaná), nastavení pravděpodobností mutace a křížení a případně na nutnosti výpočtu fitness funkce převodem z funkce jiných vlastností (to je i případ BFA – naše d má jiné vlastnosti než má mít fitness funkce).

5.3 Aplikace SGA pro řešení BFA

Datová reprezentace jedinců je v případě BFA velmi snadná – můžeme totiž přímo vzít binární matice. Jedinec je tedy tvořen maticemi F a A, uloženými v paměti po řádcích zleva doprava (běžné ukládání matic v paměti).

Mutaci lze realizovat buď inverzí jednotlivých bitů, nebo inverzí celých řádků (práce se sloupci je implementačně obtížná, proto ji vynecháváme).

Křížit lze matice buď na úrovni bitů, kdy se matice kříží v libovolném bodě, nebo na úrovni řádků, kdy zůstává zachována celistvost řádků matic.

Discrepancy d má opačný průběh než fitness funkce η (menší hodnoty jsou lepší), je nutno použít nějakou formu převodu. Jelikož d je diskrétní funkce shora omezená celkovým počtem bitů v maticích, lze použít tento jednoduchý převodní vzorec ($n \times m$ a $m \times p$ jsou rozměry matic F a A, i označuje konkrétního jedince):

$$\eta(i) = m \cdot (n + p) - d(i)$$

Výpočet začíná s náhodnou populací. V každém kroku algoritmu je vytvořena celá nová generace takto: Do pomocného pole se nakopíruje každý jedinec i tolikrát, jakou hodnotu má jeho $\eta(i)$. (Toto kopírování je poměrně náročné na paměť a je jedním z viditelných problémů původního SGA.) Potom se náhodně vybírají páry jedinců z tohoto pole a zkříží se. Každý bit všech jedinců nové generace je pak s pravděpodobností p_m mutován. Zde popsaný postup se v literatuře vyskytuje v různých nuancích. Vzhledem k tomu, že cílem této práce je popis nového lepšího algoritmu, některé detaily SGA zde nebyly podrobně rozepsány.

5.4 Výsledky SGA

Při řešení BFA vykazoval algoritmus SGA poměrně slabých výsledků. Nějaké řešení sice našel, hodnota d byla však ve všech testech více než dvojnásobná ve srovnání s výsledky dosaženými metodou formálních konceptů (viz [6], [8]). Populace, i když vycházely z náhodného počátečního nastavení genů, obvykle postupně konvergují do stavu, kdy všichni jedinci jsou identičtí ve smyslu binární rovnosti. Jakmile nastane tato situace (stav úplné konvergence), vývoj jde dále kupředu jen pomocí mutací. Odtud přímo plyne fakt, že funguje-li mutační operátor na úrovni celých řádků matic, je schopnost hledat nová řešení silně omezena. To se potvrdilo i experimentálně.

Závěr: SGA dokáže řešit BFA, avšak podstatně hůře než výše zmíněná metoda formálních konceptů, používání SGA je zde tedy ve své podstatě zbytečné.

5 Genetický algoritmus GABFA

6.1 Princip GABFA

Zjištění, že klasický genetický algoritmus SGA neřeší BFA moc dobře, nemusí automaticky znamenat, že problém BFA nelze úspěšně řešit jiným genetickým algoritmem. Tato kapitola představuje modifikovaný genetický algoritmus, nazvaný GABFA (zkratka z „Genetický Algoritmus pro Binární Faktorovou Analýzu“), který vykazuje diametrálně odlišné (mnohem lepší) výsledky. Tento algoritmus vznikl postupným zkoušením různých modifikací SGA a analýzou chování systému při aplikaci na umělá i reálná data. Hlavním zdrojem inspirace konečné verze algoritmu GABFA, která je zde představena, byly práce profesora Oleje zabývající se využitím genetických algoritmů pro učení neuronových sítí a systémovou automatizaci v průmyslu, viz [9].

Princip fungování GABFA se od SGA v mnoha ohledech liší. Stejný zůstal jen konceptuální pohled: Řešení BFA je hledáno pomocí postupného genetického vývoje jedinců v populaci. Každý jedinec žije pouze jednu generaci, reprodukce opět funguje na bázi operátorů selekce, křížení a mutace.

6.2 Datová reprezentace, využití znalostí

Využití našich znalostí může pomoci k rychlejšímu nalezení řešení (tj. během menšího počtu generací). Ukázalo se však, že přílišné zasahování do pseudonáhodného chování GA spíše škodí, než pomáhá, proto musíme postupovat velmi opatrně. Experimentálně se ukázalo, a potvrzují to i práce jiných autorů, že je lepší do výpočtu GA raději žádné znalosti nezanášet.

Pokud je to možné, pokusíme se provádět pseudo-dělení binárních matic. Každý jedinec pak obsahuje jen matici A, zbytek dopočítáme jako $\mathsf{F} = \mathsf{X}/\mathsf{A}$.

Dále je vhodné aplikovat veškeré známé metody předzpracování (preprocessing) na matici X, viz [4], [5], [6], [7], [8].

6.3 Možné způsoby parametrizace algoritmu

Velikost populace $|pop|$ – stačí i poměrně malé hodnoty, např. 200 jedinců.

Pravděpodobnost mutace p_m – obvykle v intervalu $[0.001 - 0.1]$. Příliš velká hodnota devastuje populaci (viz Černobyl 1986). Oproti živým organizmům je zde však velmi silně aplikován výběr silnějších jedinců, takže devastující dopad mutací je v GABFA omezen.

Pravděpodobnost křížení (crossover) p_c – v intervalu $[0.1 - 0.5]$. Viz níže.

Nastavení těchto tří číselných hodnot ovlivňuje chování algoritmu. Požadujeme-li větší „jistotu“, je možné provést výpočet s několika různými nastaveními parametrů a vybrat potom nejlepší řešení ze všech. Není to však nutné.

Za „parametr“ nepovažujeme způsob převodu d na η. Jelikož GABFA má menší nároky na vlastnosti funkce η, stačí pouhá změna znaménka $\eta = -d$.

6.4 Inicializace (první populace)

První populace je vytvořena náhodně. To je velmi rychlé, je proto možno vytvořit na začátku více jedinců a umožnit tak rychlejší rozběhnutí GA. Praxe však ukázala, že vytváření velkých populací je zbytečné, neboť GABFA je natolik robustní, že dosáhne řešení z „téměř“ libovolné počáteční populace.

6.5 Krok výpočtu (jedna generace)

Krokem výpočtu rozumíme vytvoření další generace z generace stávající. Na vstupu očekáváme libovolnou populaci o velikosti $\geq |pop|$ a výstupem je opět populace o velikosti $\geq |pop|$. Ta je potom použita jako vstup v dalším kroku.

1. Všem jedincům spočítáme fitness hodnotu. Zapamatujeme si nejlepší řešení. Najdeme-li jedince, jehož $\eta = 0$, výpočet končí (máme nejlepší řešení).
2. Seřadíme jedince sestupně podle jejich fitness hodnot.
3. Selekce – Zmenšíme populaci na velikost $|pop|$. (Vezmeme $|pop|$ nejlepších.)

4. Křížení – Všichni jedinci se křížení zúčastní, bez ohledu na jejich fitness hodnoty. Pro každého jedince i provedeme následující postup:
   (a) Náhodně vybereme partnera j, $j \neq i$.
   (b) Každý řádek z i nahradíme s pravděpodobností p_c řádkem na stejné pozici v j.
   (c) Přidáme nově vzniklého jedince do populace pro další generaci.
5. Postup křížení popsaný v bodě 4 opakujeme třikrát.
6. Mutace – Procházíme celou novou populaci a každý bit změníme s pravděpodobností p_m na opačnou hodnotu.

Takto modifikovaný GA velice dobře řeší problém BFA (důkaz experimentálně – předvedeno níže). Přitom je důležité přesně dodržet předepsaný postup. V některé jeho body mohou svádět k zdánlivě banálním změnám z důvodu snadnější implementace, může to však mít fatální dopad na fungování algoritmu jako celku. Některé možnosti úprav algoritmu jsou popsány a diskutovány níže.

Hledané **řešení** jsme si zapamatovali v bodě 1 (tedy ne v poslední generaci).

6 Jiné varianty algoritmu GABFA

V odborné literatuře věnované genetice převládá názor, že je velmi malá pravděpodobnost, že by člověk dokázal nyní vymyslet princip radikálně lepší, než to, co příroda vytvořila po 4 miliardách let postupného vývoje. Proto se při hledání lepších variant GA inspirujeme především živou přírodou a hledáme takové analogie, které by mohly vést ke zlepšení našeho algoritmu.

7.1 Distribuovaný GABFA

Distribuovaným GA rozumíme logicky paralelní existenci několika instancí GABFA, což umožní provádět současně vývoj v různých oddělených populacích. Vybraní jedinci se *jednou za čas* přestěhují do jiné populace. Je prokázáno, že tento princip obecně zlepšuje GA.

Implementace této varianty je možná několika způsoby. Především je zajímavé, že už samotná logická paralelizace, tj. s využitím pouze jednoho procesoru a jednoho počítače, často vede ke zlepšení řešení. Navíc implementace fyzicky paralelního výpočtu s využitím více počítačů je velmi snadná, neboť jednotlivé výpočetní uzly (počítače) spolu komunikují jen sporadicky, když realizují stěhování jedinců.

7.2 Diploidní (nebo obecně multiploidní) chromozomy

V terminologii informatiky zatím vždy jednu hodnotu (bit matice) ukládáme do jednoho genu (rovněž bit – v paměti). Diploidní chromozomy pak zavádějí princip běžný v živé přírodě, kdy jednotka informace je v paměti uložena ve dvou bitech. Do mutací vstupuje každý bit samostatně; při křížení mohou být jedinci rozděleni na dvě poloviny, což umožňuje zcela nový přístup k této operaci. Právě to je důvodem úspěchu této varianty. Byly publikovány práce, které experimentálně dokazují, že diploidní chromozomy zlepšují GA.

7.3 Zavedení dvou nebo více pohlaví

Horní a dolní polovina matice A by mohly každá reprezentovat odlišné pohlaví. Tento princip lze zobecnit i na více než dvě pohlaví, je však zajímavé, že varianta více pohlaví se v přírodě nevyskytuje (alespoň nám není známa). Úplná implementace této varianty však naráží na řadu skrytých problémů a v literatuře popisující reálně fungující aplikace GA se prakticky nevyskytuje.

7.4 Jiný počet potomků

GABFA vytváří ke každému jedinci právě 3 potomky s náhodným partnerem. Každý jedinec má tedy minimálně 3 a průměrně 6 potomků. Jelikož p_c obvykle nastavujeme < 0.5, první rodič má větší vliv než k němu náhodně vybraný partner. To redukuje vliv náhodných hodnot na křížení a potažmo celý GABFA.

Olej [9] doporučuje vytvářet 4 potomky (bez bližšího vysvětlení, zřejmě s odkazem na jiné své publikace). Jsou-li pravděpodobnosti p_c a p_m zvoleny velmi malé, počet potomků má větší vliv na chování algoritmu. Experimentálně bylo zjištěno, že větší počet potomků zrychlí konvergenci a uspíší nalezení řešení, které je však méně kvalitní. Menší počet potomků zpomalí konvergenci – pokud ji však zcela nezastaví, umožní nám najít lepší řešení. Testy ukázaly, že v případě GABFA je nejlepší vytvářet 3 nebo 4 potomky.

Větší počet potomků také umožní najít řešení už při menším počtu generací. Vytváříme však více potomků v každé jednotlivé generaci, čili každý krok výpočtu trvá o to déle. Nelze tedy srovnávat různé GA pomocí jednoduchých grafů zobrazujících závislost chyby nejlepšího řešení proti počtu použitých generací.

7.5 Přežívání nejlepších jedinců – nedoporučeno

Problém „ztráty vítěze“ je možno vyřešit tak, že jeden nebo i více nejlepších jedinců je automaticky okopírováno do další generace. Použití tohoto zdánlivě chytrého principu však vede k tomu, že celý GA přestane fungovat. Ukázalo se, že jedinec, který je nejlepší v některé z prvních generací, začne hodně rychle „kopírovat“ sám sebe do celé populace. Další vývoj populace je tak znemožněn.

7.6 Mutace po řádcích – nedoporučeno

Mutace má obecně velmi pozitivní vliv na vývoj populace. Díky bodu 5 algoritmu GABFA je v každé generaci vytvořeno 3× více jedinců, než bude nakonec použito v dalších výpočtech. Tím je eliminováno nebezpečí „vadných“ mutací, jelikož jedinci devastovaní mutací jsou jednoduše z dalších výpočtů vyřazeni. Samotná možnost mutace naopak dává šanci na náhodné nalezení nějakého lepšího řešení.

Mutace po řádcích by zde znamenala zaměnit celý jeden faktor za náhodnou hodnotu. Testy ukázaly, že tato úprava mutace velmi sníží kvalitu celého algoritmu – pozitivní přínos mutace se prakticky vytratí.

7 Testy

Podobně jako u jiných metod z oblasti soft computingu, ani funkčnost genetických algoritmů nelze formálně dokázat. Proto důkazy provádíme experimentálně.

GABFA jsme testovali na datech použitých v dříve publikovaných pracích o BFA (např. [4], [5], [6], [8], [11]), pro snazší srovnání s jinými známými metodami.

Výsledky jsou poměrně jednoznačné, proto namísto podrobných výsledků zde pro lepší přehlednost uvádíme jen souhrnný závěr z testů. Provedli jsme několik desítek testů opakovaně pro různé datové sady (tj. pro různé matice X).

V případě řídké matice je výpočet extrémně rychlý a přesný. Genetický algoritmus v tomto případě velmi rychle najde globální minimum, řádově z miliardy možností vybere správné řešení během malého zlomku sekundy. Na řídkých datech se však také osvědčil algoritmus využívající formální koncepty – u řídkých dat je totiž výsledný konceptuální svaz poměrně malý, a tak jeho prohledávání je také velmi rychlé (často rychlejší než výpočet genetickým algoritmem).

V případě běžné matice generované uměle je již vidět přínos genetického algoritmu GABFA. Uměle generovaná matice neobsahuje informační šum (ať už by byl jakéhokoliv typu – pokud jej tam tedy nepřidáme záměrně), takže opět lze s výhodou využít metodu výpočtu přes formální koncepty. U běžné matice (která pochopitelně není řídká) je však genetický algoritmus jednoznačně výhodnější, neboť konceptuální svaz je obecně velmi velký a jeho prohledávání je pomalé.

V případě úplně reálných dat je již naprosto jasně vidět, že genetický algoritmus nechává ostatní metody výpočtu BFA daleko za sebou. Dokáže efektivně eliminovat nepodstatnou část informace a nalézt dobré řešení rozkladu matice.

8 Závěr

Genetický algoritmus GABFA je velmi dobrou metodou řešení BFA. Pracuje přitom velmi obecně a nevyžaduje od analyzované datové sady žádné zvláštní vlastnosti. Jeho funkčnost však (zatím) není dokázána formálně.

Reference

1. Goldberg, D.E.: *Genetic Algorithms in Search...* Addison-Wesley, **1989**.
2. Holland, J.H.: *Adaptation in Natural and Artificial Systems...* Ann Arbor, **1975**.
3. Hondroudakis, A., et al.: *An Introduction to Genetic Algorithms Using RPL2.*
4. Húsek, D., et al.: O jednom neuronovém přístupu... In: *Znalosti 2004*, pp. 327-337.
5. Keprt, A.: Binární faktorová analýza a komprese obrazu NN. In: *Wofex 2003.*
6. Keprt, A.: Using Blind Search and Formal Concepts for BFA. In: *Dateso 2004.*
7. Keprt, A.: Binary Factor Analysis. In: *Wofex 2004.* VŠB TU Ostrava, pp. 298–303.
8. Keprt, A., Snášel, V.: Binary Factor Analysis with Help of Formal Concepts. In: *CLA 2004.* VŠB TU Ostrava, pp. 90–101, ISBN 80-248-0597-9.
9. Olej, V.: Comparison Of Distributed GA and ES. In: *Mendel '97.* pp. 99–104.
10. Olej, V.: *Modelovanie ekonomických procesov...* M & V, **2003**, ISBN 80-90324-9-1.
11. Řezanková, H., et al.: Using Standard Statistical Procedures for BFA. In: *SIS 2003.*

Annotation:

Genetic Algorithms for Dimension Reduction and Binary Data Analysis

We present a new genetic algorithm GABFA for reduction of binary space dimension based on Binary (Boolean) Factor Analysis. The presented algorithm seems to be faster and more robust than other known algorithms.

Abdukcia bez podmienok konzistentností

Martin Lang

Fakulta matematiky, fyziky a informatiky Univerzity Komenského, Bratislava
lang@drp.fmph.uniba.sk

Abstrakt. Abdukcia je jedným zo spôsobov logického uvažovania. Využitie má najmä pri problematike diagnostikovania, plánovania a prevencie.
Klasické prístupy k definovaniu abdukcie (napríklad [6], [4], [5]) nie sú stavané na prácu s nekonzistentnými poznatkami, ktoré sú však celkom bežné.
V tejto práci použijeme modulárnu reprezentáciu znalostí (teóriu v pozadí), ktorá bude vedieť pracovať aj s nekonzistenosťami, čím dosiahneme vernejšiu reprezentáciu znalostí a abdukcie. Modularita teórie v pozadí umožní kontextovo závislé určenie, ktoré z odporujúcich si znalostí prijmeme za pravdivé.
Určenie konkrétneho modulu teórie v pozadí, ku ktorému sa abdukovateľná hypotéza viaže a určenie ich vzájomnej preferencie umožní vybrať z množiny abduktívnych vysvetlení to, ktoré je v danej situácii najvhodnejšie.

Kľúčové slová: Abdukcia, logické uvažovanie, nekonzistentnosť, preferencia

1 Úvod

Abdukcia je spôsob logického myslenia, pri ktorom odpovedáme na otázku, ktoré fakty zapríčiňujú dané pozorovania. Abdukcia má využitie najmä v problematike diagnostikovania, plánovania a prevencie.

Základnými komponentami abduktívneho myslenia sú pozorovania, teória v pozadí a (abduktívne) vysvetlenia daných pozorovaní. Pod pojmom teória v pozadí rozumieme znalosti, ktorými disponujeme, pozorovaniami opisujeme fakty, ktoré nastali a ktoré logicky nevyplývajú zo samotnej teórie v pozadí. Keď však teóriu v pozadí obohatíme o nejaké hypotézy, dokážeme už pozorovania vysvetliť (vyplývajú z rozšírenej teórie).

Formálnemu popisu abdukcie už boli venované viaceré práce (napríklad [6], [4], [5]), všetky však vyžadujú konzistentnosť pozorovaní, teórie v pozadí a abduktívneho vysvetlenia. Ide o podmienku pomerne zväzujúcu. Veľmi často je potrebné vysvetliť konflikt pozorovaní s našimi znalosťami. Navyše znalosti samotné nebývajú konzistentné a prijatá hypotéza (vysvetlenie) môže byť v rozpore s teóriou, ktorú prijímame.

V tejto práci sa budeme venovať abdukcii bez podmienok konzistentností. Abdukciu postavíme nad modulárnou teóriou v pozadí, ktorá sa s nekonzistentnosťami bude vedieť vysporiadať. Modularita teórie v pozadí umožní kontextovo závislé určenie, ktoré z odporujúcich si znalostí prijmeme.

V takto založenej abdukcii umožníme abduktívne vysvetlenia daných pozorovaní do teórie v pozadí nie len prijímať, ale ich aj odstraňovať.

Nie všetky abduktívne vysvetlenia sú si rovnocenné, niektoré sú v danej situácii vhodnejšie, iné menej vhodné. Preto zavedieme pojem kontextovo závislej preferencie medzi abdukovateľnými hypotézami. Táto preferencia umožní nájdenie maximálne preferovaného abduktívneho vysvetlenia daných pozorovaní.

Zavedením dôverčivého a skeptického prístupu k hľadaniu abduktívnych vysvetlení umožníme lepšie využitie abdukcie v situáciách ako je diagnostikovanie (využíva sa tu najmä dôverčivý prístup) a plánovanie (kde sa využíva skeptický prístup).

V prípade záujmu o väčšiu detailnosť textu odkazujem čitateľa na rozsiahlejšiu prácu [3].

2 Abdukcia

Ako príklad klasického prístupu k abdukcii vhodne poslúži základná definícia abdukcie podľa [6]. Po jej predstavení zameriame našu pozornosť na rôzne prístupy k hľadaniu abduktívnych vysvetlení.

2.1 Klasický pohľad na abduktívne vysvetlenie

Nech Γ je teória nad jazykom $\mathcal{L}$, ε je množina viet (priestor všetkých možných pozorovaní), $\Delta \subseteq \varepsilon$ je množina uskutočnených pozorovaní. Φ je množina možných vysvetlení, pričom $\varepsilon \cap \Phi = \emptyset$. Množinu Φ budeme nazývať množinou abdukovateľných hypotéz.

Ak je dané $\Delta \subseteq \varepsilon$ a teória v pozadí Γ, abduktívne vysvetlenie množiny viet Δ na základe teórie Γ nazývame minimálnu konečnú množinu $A \subseteq \Phi$ takú, že

$$A \cup \Gamma \textit{ je konzistentná množina,} \tag{1}$$

$$\Gamma \not\models \Delta, \tag{2}$$

$$\Gamma \cup A \models \Delta. \tag{3}$$

2.2 Rôzne pohľady na hľadanie abduktívnych vysvetlení

Abduktívne anti-vysvetlenie. Ak hľadáme skutočnosti, ktoré nám objasňujú vznik danej situácie, hovoríme o hľadaní abduktívneho vysvetlenia.

Ak určité fakty nie sú splnené a je túto skutočnosť potrebné vysvetliť, použijeme prístup nazývaný anti-vysvetlenie (podľa [5]). Anti-vysvetlenie je využívané najmä pri problematike prevencie, kedy je potrebné určiť opatrenia potrebné na to, aby nedošlo k vzniku danej situácie.

Dôverčivý prístup. Niekedy je pri hľadaní abduktívnych vysvetlení postačujúce, ak zo všetkých možných situácii existuje aspoň jedna, v ktorej sú splnené dané pozorovania. Preto v tomto článku zavádzame pojem dôverčivého prístupu k abdukcii.

Využitie tohto prístupu je najmä v procese diagnostikovania, kedy už aj náznak problému má byť dôvodom na jeho dôkladné overenie.

Skeptický prístup. Pre situácie, v ktorých je potrebné, aby dané pozorovania boli prítomné vo všetkých situáciách, ktoré môžu vzniknúť, zavádzame pojem skeptického prístupu k abdukcii. Využíva sa najmä pri plánovaní, kedy chceme mať istotu, že sa náš plán za každých okolností uskutoční.

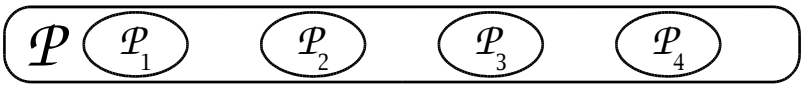
3 Nový prístup k abdukcii - preferenčný abduktívny dynamický program

Aby sme mohli abstrahovať od detailov a venovať sa priamo novým prvkom abdukcie, zavádzame pojem abstraktného dynamického modulárneho logického programu (ADMLP). ADMLP poslúži ako základný mechanizmus, nad ktorým postavíme abdukciu. Svojou modularitou a využitím dynamickej preferencie jednotlivých modulov umožní vernejšiu reprezentáciu znalostí a uvažovanie s nekonzistentnosťami (záujemcom o konkrétnejšiu reprezentáciu použiteľnej teórie v pozadí dávam do pozornosti [3]).

V ďalšej časti definujeme abdukciu nad ADMLP z pohľadu dôverčivého aj skeptického prístupu. Neskôr zavedieme (dynamickú) preferenciu, ktorá nám umožní rozlišovať v danej situácii najvhodnejšie abduktívne vysvetlenia.

3.1 Abstraktný dynamický modulárny logický program (ADMLP)

ADMLP nad jazykom $\mathcal{L}$ je dvojica $\mathcal{P} = (\mathcal{P}_I, I)$, kde I je množina indexov a $\mathcal{P}_I$ je množina logických programov nad $\mathcal{L}$, indexovaných podľa prvkov $i \in I$. Prvky $\mathcal{P}_i \in \mathcal{P}_I$, kde $i \in I$, nazývame moduly programu $\mathcal{P}$. Predpokladáme jazyk, v ktorom logický program môže byť nekonzistentný.



Obr. 1. Príklad ADMLP $\mathcal{P} = (\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}, \{1, 2, 3, 4\})$.

Statická sémantika. Predpokladajme, že nad $\mathcal{L}$ je definovaná statická sémantika, ktorá programu P nad $\mathcal{L}$ priradí množinu modelov:

$$Sem(P) = Mod(P), \tag{4}$$

pričom

$$Sem(P) = \emptyset, \tag{5}$$

ak P nie je konzistentný program nad $\mathcal{L}$.

Dynamická sémantika ADMLP. Predpokladajme ADMLP $\mathcal{P}$. Nie je vylúčené, že $\bigcup_{i \in I} \mathcal{P}_i$ je nekonzistentná množina. Cieľom dynamickej sémantiky je, aby sme aj v takýchto prípadoch dostali nepázdnu množinu modelov.

Predpokladajme sémantiku $DynSem$, ktorá programu $\mathcal{P}$ priradí množinu modelov Mod. Pri výpočte modelov (nekonzistentného) programu $\mathcal{P}$ sa do úvahy berie preferencia modulov Φ. Táto preferencia nie je vopred daná, je určená dynamicky:

$$\Phi \subseteq Mod \times I \times I. \tag{6}$$

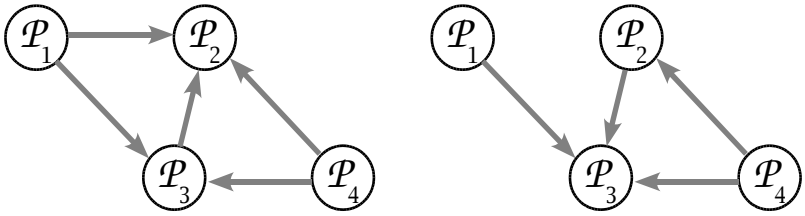
Model programu $\mathcal{P}$ je určený ako pevný bod zobrazenia:

$$DSem(\mathcal{P}, \Phi(m)) = m. \tag{7}$$

Sémantiku $DynSem$ definujeme ako množinu modelov spĺňajúcich podmienku 7:

$$DynSem_{\Phi}(\mathcal{P}) = \{m : DSem(\mathcal{P}, \Phi(m)) = m\}. \tag{8}$$

Poznámka 1. Ako príklad sémantiky vyhovujúcej ADMLP môže byť chápaná sémantika použitá v [3].



Obr. 2. Príklad grafickej reprezentácie relácie preferencie $\Phi(m_1)$ a $\Phi(m_2)$.

Operátor update. Update $A_i \in A_I$ modulu $\mathcal{P}_i \in \mathcal{P}_I$ definujme ako množinu hypotéz E_i prijímaných do modulu $\mathcal{P}_i$ a množinu hypotéz F_i odstraňovaných z modulu $\mathcal{P}_i$.

$$A_i = E_i \cup F_i, \tag{9}$$

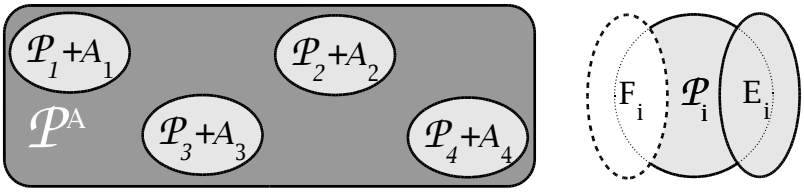
$$E_i \cap F_i = \emptyset. \tag{10}$$

Operátor update $\mathcal{U}$ programu ADMLP $\mathcal{P} = (\mathcal{P}_I, I)$ množinou (updatov) $A = (A_I, I)$ nad jazykom $\mathcal{L}$ definujme nasledovne:

$$\mathcal{U}(\mathcal{P}, A) = \mathcal{P}^A = (\mathcal{P}_I^A, I), \tag{11}$$

kde

$$\mathcal{P}_I^A = \{(\mathcal{P}_i \cup E_i) \setminus F_i : \mathcal{P}_i \in \mathcal{P}_I, E_i \cup F_i = A_i \in A_I\}. \tag{12}$$



Obr. 3. Update $\mathcal{P}^A$ programu $\mathcal{P} = (\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}, \{1, 2, 3, 4\})$ teóriou $A = (\{A_1, A_2, A_3, A_4\}, \{1, 2, 3, 4\})$. Detailný pohľad na update modulu $\mathcal{P}_i$ teóriou $A_i = E_i \cup F_i$, $\mathcal{P}_i^A = \{(\mathcal{P}_i \cup E_i) \setminus F_i\}$.

3.2 Abdukcia nad ADMLP

Nech $\mathcal{P} = (\mathcal{P}_I, I)$ je ADMLP nad $\mathcal{L}$, Δ sú pozorovania. $\mathcal{A} = (\mathcal{A}_I, I)$ nazveme množinu abdukovateľných hypotéz, I je indexová množina, $\mathcal{A}_i \in \mathcal{A}_I$ je množina abdukovateľných hypotéz prislúchajúca modulu $\mathcal{P}_i$.

Sémantika abdukcie nad ADMLP. Množinu $A = (A_I, I)$, kde $A_i \subseteq \mathcal{A}_i, i \in I$ nazveme abduktívnym vysvetlením [anti-vysvetlením] pozorovaní Δ s ohľadom na teóriu v pozadí $\mathcal{P}$, ak platia nasledovné tvrdenia:

Pre dôverčivý prístup:

$$\not\exists m \in DynSem_\Phi(\mathcal{P}) : m \models \Delta \; [m \not\models \Delta], \tag{13}$$

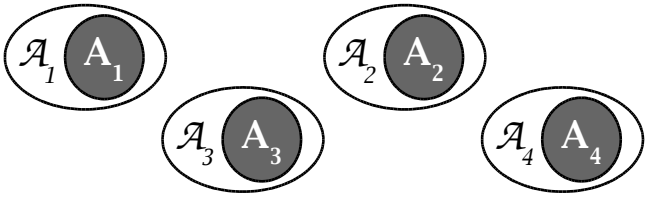
$$\exists m \in DynSem_\Phi(\mathcal{P}^A) : m \models \Delta \; [m \not\models \Delta]. \tag{14}$$

Pre skeptický prístup:

$$\not\forall m \in DynSem_\Phi(\mathcal{P}) : m \models \Delta \; [m \not\models \Delta], \tag{15}$$

$$\forall m \in DynSem_\Phi(\mathcal{P}^A) : m \models \Delta \; [m \not\models \Delta]. \tag{16}$$

Inak hovoríme, že abduktívne vysvetlenie [anti-vysvetlenie] pozorovaní Δ s ohľadom na teóriu v pozadí $\mathcal{P}$ neexistuje.



Obr. 4. Množina abdukovateľných hypotéz $\mathcal{A} = (\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}, \{1, 2, 3, 4\})$ a abduktívne vysvetlenie $A = (\{A_1, A_2, A_3, A_4\}, \{1, 2, 3, 4\})$.

3.3 Preferenčný abduktívny dynamický program

Preferenčný abduktívny dynamický logický program nad jazykom $\mathcal{L}$ je štvorica $(\mathcal{P}_I, \mathcal{A}_I, I, \mathcal{K})$, kde $\mathcal{P} = (\mathcal{P}_I, I)$ je ADMLP nad $\mathcal{L}$, $\mathcal{A} = (\mathcal{A}_I, I)$ je množina abdukovateľných hypotéz, $\mathcal{K} \subset (\bigcup_{i \in I} \mathcal{A}_i) \times (\bigcup_{i \in I} \mathcal{A}_i)$ je relácia preferencie abdukovateľných hypotéz (kontext), spĺňajúca:

$$(a_1, a_2) \in \mathcal{K} \Rightarrow (a_2, a_1) \notin \mathcal{K}, \tag{17}$$

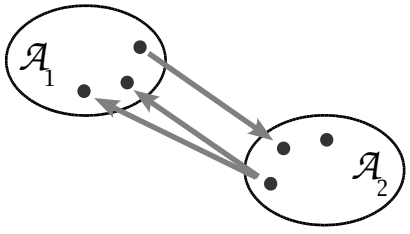
$$(a_1, a_2) \in \mathcal{K}, (a_2, a_3) \in \mathcal{K} \Rightarrow (a_1, a_3) \in \mathcal{K}. \tag{18}$$

Kontext $\mathcal{K}$ môže byť závislý od modelu programu $\mathcal{P}$. Má to pre nás zmysel pri hľadaní abduktívnych vysvetlení konkrétneho modelu. Ak by sme uvažovali viacero modelov z množiny $DynSem_{\Phi}(\mathcal{P}^A)$, dostali by sme vo všeobecnosti rôznu reláciu preferencie $\mathcal{K}$.

$$DSem(\mathcal{P}^A, \Phi(m)) = m, \Delta \subseteq m, \tag{19}$$

$$\mathcal{K} = \Psi(m), \Psi \subset Mod \times (\bigcup_{i \in I} \mathcal{A}_i) \times (\bigcup_{i \in I} \mathcal{A}_i). \tag{20}$$

Pri množine abduktívnych vysvetlení $A = (A_I, I)$, kde $A_i \subseteq \mathcal{A}_i$, $i \in I$ vyžadujeme splnenie podmienok 13, 14, 15 a 16.



Obr. 5. Grafická reprezentácia relácie preferencie $\mathcal{K}$ abdukovateľných hypotéz $\mathcal{A} = (\{\mathcal{A}_1, \mathcal{A}_2\}, \{1, 2\})$, kde $\mathcal{A}_1 = \{h_{11}, h_{12}, h_{13}\}$, $\mathcal{A}_2 = \{h_{21}, h_{22}, h_{23}\}$, $\mathcal{K} = \{(h_{21}, h_{11}), (h_{21}, h_{12}), (h_{13}, h_{22})\}$.

Preferovanejšie abduktívne vysvetlenie. Definujme hĺbku abduktívneho vysvetlenia A ako množinu všetkých abdukovateľných hypotéz, ktoré sú aspoň natoľko preferované, ako hypotézy množiny A:

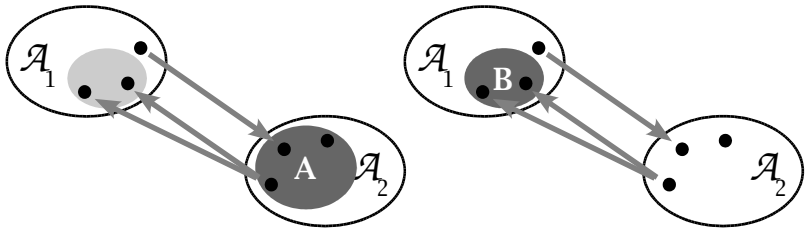
$$\mathcal{K}_A = A \cup \{v \in \bigcup_{i \in I} \mathcal{A}_i : \exists a \in A, (a, v) \in \mathcal{K}\} \tag{21}$$

Skutočnosť, že $B \subseteq \mathcal{A}$ je preferovanejšie ako $A \subseteq \mathcal{A}$ značíme $A < B$. Definujme:

$$A < B \Leftrightarrow \mathcal{K}_B \subset \mathcal{K}_A. \tag{22}$$

Abduktívne vysvetlenie A je rovnako preferované, ako abduktívne vysvetlenie B práve vtedy, ak

$$\mathcal{K}_A = \mathcal{K}_B. \tag{23}$$



Obr. 6. Abduktívne vysvetlenie B je preferovanejšie ako vysvetlenie A ($A = \{h_{21}, h_{22}, h_{23}\}$, $\mathcal{K}_A = \{h_{11}, h_{12}, h_{21}, h_{22}, h_{23}\}$, $B = \{h_{11}, h_{12}\}$, $\mathcal{K}_B = \{h_{11}, h_{12}\}$, $\mathcal{K}_B \subset \mathcal{K}_A$).

Maximálne preferované abduktívne vysvetlenie. Abduktívne vysvetlenie A nazývame maximálne preferované abduktívne vysvetlenie pozorovania Δ s ohľadom na teóriu v pozadí $\mathcal{P}$, ak neexistuje preferovanejšie abduktívne vysvetlenie B.

Keďže nie každé dve abduktívne vysvetlenia sú porovnateľné, môže existovať celá množina maximálne preferovaných abduktívnych vysvetlení. Je možné uvažovať aj o maximálne preferovaných abduktívnych anti-vysvetleniach.

Minimálne abduktívne vysvetlenie. Abduktívne vysvetlenie A pozorovania Δ nazveme menším ako vysvetlenie B, ak

$$A \subset B. \tag{24}$$

Abduktívne vysvetlenie A nazveme minimálnym, ak pre všetky abduktívne vysvetlenia B platí:

$$(B \subseteq A) \Rightarrow (B = A). \tag{25}$$

Aj v tomto prípade môžeme čakať celú množinu minimálnych abduktívnych vysvetlení a môžeme taktiež uvažovať o minimálnych abduktívnych antivysvetleniach.

4 Zhrnutie

Práca prezentuje triedu modulárnych logických jazykov, ktoré nevyžadujú podmienku konzistentnosti a využívajú kontext na vysporiadanie sa s nekonzistentnosťami. Nad takýmto jazykom je založená abdukcia, v ktorej nie je nutné splnenie podmienok konzistentnosti. Abdukcia je rozšírená o (kontextovo závislú) preferenciu abdukovateľných hypotéz, ktorá umožní identifikovanie najvýhodnejších (maximálne preferovaných) abduktívnych vysvetlení daných pozorovaní[1].

Reference

1. J. J. Alferes, J. A. Leite, L. M. Pereira, P. Quaresma. *Planning as Abductive Updating*, In D. Kitchin (ed.), Procs. of AISB'00, 2000.
http:// citeseer.ist.psu.edu/alferes00planning.html.
2. Katsumi Inoue and Chiaki Sakama. *Negation as failure in the head*. Journal of Logic Programming, 35:39-78, 1998.
http://citeseer.ist.psu.edu/ inoue98negation.html.
3. Martin Lang. Abdukcia bez podmienok konzistentností (diplomová práca). http://www.drp.fmph.uniba.sk/∼lang/master_thesis/.
4. J. A. Leite, J. J. Alferes and L. M. Pereira. *Multi-dimensional Dynamic Knowledge Representation*. In T. Eiter, W. Faber and M. Truszczynski, Procs. of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01), pages 365- 378, Springer-Verlag, LNAI 2173, 2001.
http://centria.di.fct.unl.pt/ jleite/papers/lpnmr01.pdf.
5. C. Sakama and K. Inoue. *Updating extended logic programs through abduction*. In M. Gelfond, N. Leone, and G. Pfeifer, editors, LPNMR'99. Springer, 1999.
http://citeseer.ist.psu.edu/sakama99updating.html.
6. Ján Šefránek. *Inteligencia ako výpočet*. IRIS, 2000. ISBN: 80-88778-96-4.

Annotation:

Abduction without consistency constraints
This paper is aimimg at an improved definition of abduction, which does not need consistency constraints. Second improvement is based on definition of structural hierarchy of abducable knowledges. This hierarchical structure allows better knowledge representation and reasoning.

[1] Tento článok bol podporený grantom VEGA 1/0173/03.

Learning rules to predict next page in a click-stream

Vladimír Laš[1], Tomáš Kočka[2,1], and Petr Berka[1]

[1]FIS, VŠE nám. W. Churchilla 4, 130 67, Praha
vladalas@hotmail.com, berka@vse.cz

[2]Adastra, Benešovská 10, 101 00, Praha
Tomas.Kocka@AdastraCorp.com

Abstract. We present a set covering algorithm to describe sequences of www pages visits in click-stream data. It utilizes the approach of rule specialization like the well known CN2 algorithm, however it describes sequences instead of sets unlike the CN2 algorithm. It can be used to predict next page to be viewed by a user or to describe the most typical paths of www pages visitors and the dependencies among the www pages. We have successfully used the algorithm on real data from an internet shop and we mined useful information from the data.

Keywords: click-stream, decision rules, web usage mining

1 Introduction

According to the W3C Web Characterization Activity, click-stream is a sequential series of page view (displays on user's browser at one time) requests. A user session is the click-stream of page views for a single user across the entire web, a server session is a click-stream in a user session for a particular web site. A set of server sessions (visits) is the necessary input for web usage mining tools.

We can categorize web mining into three areas: web content mining, web structure mining and web usage mining [8]. Web usage mining mines the data derived from interactions of the users while browsing the web. The web usage data includes the data from web server access logs, proxy server logs, browser logs, user profiles, registration data, cookies, user queries etc. A web server log is an important source for performing web usage mining because it explicitly records the browsing behavior of site visitors. The typical problem (solved in the data preprocessing step) is thus distinguishing among unique users, server sessions episodes etc.

Web usage mining focuses on techniques that could predict user behavior while the user interacts with the web. The applications of web usage mining could be classified into two main categories: (1) learning user profile or user modeling, and (2) learning user navigation patterns [5]. The methods used for web usage mining are (descriptive) statistical analysis, association rules (to relate pages that are often referenced together), clustering (to build usage clusters or page clusters), classification (to create

user profiles), sequential pattern discovery (to find inter-session patterns such that the presence of a set of page views is followed by another set of page views in a time-ordered set of episodes), or dependency modeling (to capture significant dependencies among various variables in the web domain) [7]. Some systems have already been developed for this area: WebSIFT (that uses clustering, statistical analysis or association rules) [3], WUM (that looks for association rules using some extension of SQL) [6], or WebLogMiner (that combines OLAP and KDD) [9].

The algorithm described in this paper contributes to the area of sequential pattern discovery by learning decision rules to predict next page the user will visit based on his session history (pages visited just before). We describe the algorithm in section 2 and give some experimental results obtained on real web log data in section 3.

2 Learning Rules

Decision rules in the form

$$Ant \Rightarrow Class$$

where *Ant* (antecedent, condition) is a conjunction of values of input attributes (called categories or selectors) and *Class* is a category of class attribute *C*, are one of most popular formalisms how to express classification models learned from data.

The commonly used approach to learning decision rules is the set covering approach also called „separate and conquer". The basic idea of this approach is to create a rule that covers some examples of a given class and remove these examples form the training set. This is repeated for all examples not covered so far as shown in Fig. 1. There are two basic ways how to create a single rule (step 1 of the algorithm):

1. by rule generalization, i.e. by removing categories from antecedent of a potential rule (starting from a rule with categories of all input attributes in the antecedent) – this method is used in the AQ algorithms by Michalski (see e.g. [4]).
2. by rule specialization, i.e. by adding categories to the antecedent of a potential rule (starting from a rule with empty antecedent) – this method is used e.g. in CN2 [2] or CN4 [1].

Set covering algorithm

1. find a rule that covers some positive examples and no negative example of a given class (concept),
2. remove covered examples form the training set D_{TR},
3. if D_{TR} contains some positive examples (not covered so far) goto 1, else end.

Fig. 1. Set covering algorithm for two class problems

We follow the set covering approach based on rule specialization in our algorithm as well. The main difference to conventional set covering algorithms is due to the fact that instead of unordered set of categories we deal with an ordered sequence of page views (pages for short). So we are looking for rules in the form

$$Ant \Rightarrow page\ (p)$$

where *Ant* is a sequence of pages
page is a page view that directly follows the sequence *Ant*
p is the validity of the rule, i.e. $p = \frac{||Ant//page||}{||Ant||}$.

In the formula above we denote the number the occurrences of a *sequence* in the data by ||*sequence*|| and a concatenation of two sequences *s*1 and *s*2 by *s*1//*s*2.

The main idea of our algorithm is to add (for a particular page to be predicted) rules of growing length of *Ant* – we thus create rules by rule specialization. We check each rule against its generalization included in the model so far. Adding a new rule to the model is determined by χ^2 test that tests the null hypothesis that the validity of these two rules are the same. If the hypothesis is rejected at a given significance level then the rule in question is added to the model, its generalization is updated by recomputing its validity by ignoring (removing) sequences that are covered by the newly added rule.

As we assume that most relevant for prediction of occurrence of *page* are pages that are closest to *page*, the specialization of the rule $Ant \Rightarrow page$ is done by adding a new page to the beginning of the sequence *Ant*. Analogously, a generalization of the rule $Ant \Rightarrow page$ is done by removing a page from the beginning of the sequence *Ant*. The algorithm is shown in Fig. 2.

The input parameters of the algorithm are l_{max} (max. length of the sequence *Ant*), n_{min} (min. relative frequency of a page – this parameter set to zero will enable to create a rule that page *Y* never follows after page *X*). The parameter l_{max} is also used for data preprocessing; we transform each server session[1] of arbitrary length into a set of episodes of length $l_{max} + 1$ using a sliding window. So e.g. the two sessions

A,B,E
A,B,C,E,D

will be for $l_{max} = 2$ transformed into following set of episodes

Ø,Ø,A
Ø,A,B
A,B,E
Ø,Ø,A
Ø,A,B
A,B,C
B,C,E
C,E,D

[1] Recall that a server session corresponds to one visit of one user on the analyzed web site.

Initialization
for each page *page* occurring in the data
1. compute its relative frequency in the data as

$$P = \frac{\text{\# of occurrence of } page \text{ in the input episodes}}{\text{\# of all input episodes}}$$

2. if $P \geq n_{min}$
2.1 add *default* ⇒ *page* into the list of rules *Rules*
2.2 add *page* into list of pages *Pages*
2.3 add *default* ⇒ *page* into list of implications *Impl*

Main loop
while *Impl* not empty do
1. take first rule *Ant* ⇒ *page* from *Impl*
2. if length of *Ant* < l_{max} then
   2.1. for each page *pp* from *Pages*
      2.1.1 find the most specific generalization of the rule *pp//Ant* ⇒ *page* in *Rules* (denote it *AntX* ⇒ *page*)
      2.1.2 compare (using chi2 test) the validity of rules *pp//Ant* ⇒ *page* and *AntX* ⇒ *page*
   2.2. from all created rules *pp//Ant* ⇒ *page* select the one with the most significant difference in validity (denote this rule pp_{best}*//Ant* ⇒ *page*)
   2.3. if pp_{best}*//Ant* ⇒ *page* significantly at a given significance level differs from *AntX* ⇒ *page* then
      2.3.1 add rule pp_{best}*//Ant* ⇒ *page* to *Rules* and *Impl*
      2.3.2 re-compute the validity of rule *AntX* ⇒ *page* by taking into account only episodes containing *AntX* and not containing *Ant*
      2.3.3 recursively update *Rules* (i.e. find the most specific generalization of *AntX* ⇒ *page* , compare this generalization with *AntX* ⇒ *page*, remove *AntX* ⇒ *page* from *Rules* if the difference is not significant etc.)
3. remove *Ant* ⇒ *page* from *Impl*

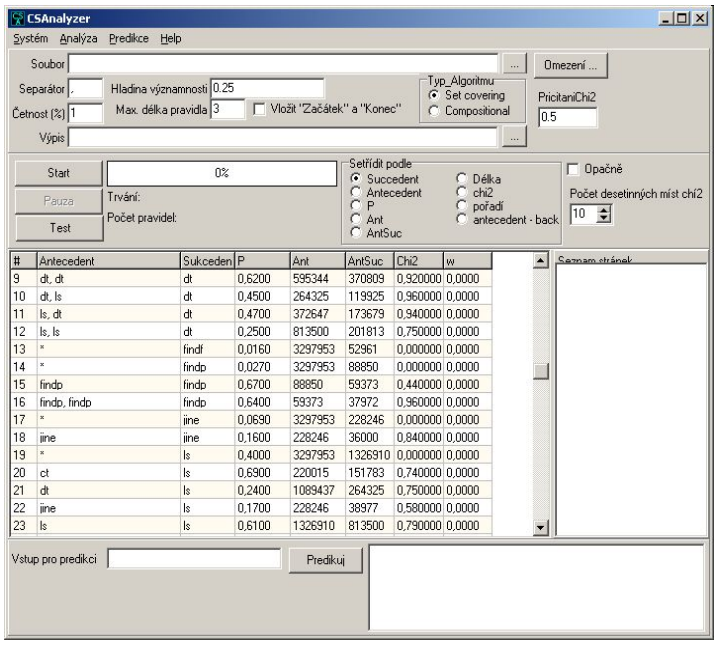
Fig. 2. The rule learning algorithm

The system is implemented in the Borland's Delphi (version 7.0) for the operation system Windows (W-98 – W-XP). The minimal configuration of the computer is not set; it's helpful to use computers with processor at least 600 MHz for more extensive analysis. The system doesn't use more than quadruple of the size of an input file in the memory.

The algorithm for searching rules itself is divided into two parts (as you could see above) – the initialization and the main loop. The main loop is implemented as a recursive function that is called for particular rules. At the beginning, the function generates specializations, after it modifies frequency of the actual rule and to the end it evaluates the chi-square test for the actual rule.

The algorithm for searching rules runs in second thread, so found rules can be showed immediately in the core thread of the application - user can stop the algorithm when its continuation isn't needful. The user could restrict the space of searching by setting maximal length of rules. Pages that don't have required minimal frequency are added to a set OTHERS and the system works with this set as with a single page.

Found rules can be saved to a file (and loaded back) or exported to Excel. The application offers a simple algorithm to predict a next page in a sequence (the system returns for each possible page a rule which has the longest antecedent consistent with the sequence and predicts the page which has the highest relative frequency associated to its rule).



#	Antecedent	Sukceden	P	Ant	AntSuc	Chi2	w
9	dt, dt	dt	0,6200	595344	370809	0,920000	0,0000
10	dt, ls	dt	0,4500	264325	119925	0,960000	0,0000
11	ls, dt	dt	0,4700	372647	173679	0,940000	0,0000
12	ls, ls	dt	0,2500	813500	201813	0,750000	0,0000
13	*	findf	0,0160	3297953	52961	0,000000	0,0000
14	*	findp	0,0270	3297953	88850	0,000000	0,0000
15	findp	findp	0,6700	88850	59373	0,440000	0,0000
16	findp, findp	findp	0,6400	59373	37972	0,960000	0,0000
17	*	jine	0,0690	3297953	228246	0,000000	0,0000
18	jine	jine	0,1600	228246	36000	0,840000	0,0000
19	*	ls	0,4000	3297953	1326910	0,000000	0,0000
20	ct	ls	0,6900	220015	151783	0,740000	0,0000
21	dt	ls	0,2400	1089437	264325	0,750000	0,0000
22	jine	ls	0,1700	228246	38977	0,580000	0,0000
23	ls	ls	0,6100	1326910	813500	0,790000	0,0000

Fig. 3. Screenshot of the system

3 Experiments

We tested our algorithm on a real data obtained from one Czech internet shop. We will describe our experiments following the methodology CRISP-DM (Cross-Industry Standard Process for Data Mining):

Business Understanding and Data Understanding. We obtained from the data provider log files which contained data from cca 30 days (about 3 millions records). The log file contained the usual information: time, IP address, page request and referee. In addition to this, the log data contained also a generated session ID so the identification of users was relatively easy – we treated a sequence of pages with the

same ID as a visit of one user[2]. An example of records in the log file is shown in Fig. 4.

The log file allowed us to identify two types of information about the visited page: page type and page content. By page type we understand information related to a general structure of the internet shop (detail of a product, shopping chart, product comparison etc), by page content we understand the product (or its category) offered on the page. These two points of view enable us to perform two different types of analyses: analysis of product preferences and analysis of shopping behavior.

```
  unix time; IP address; session ID; page request; referee
  1074589200;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/
dp/?id=124;www.google.cz;
  1074589201;194.213.35.234;3995b2c0599f1782e2b40582823b1c94;
/dp/?id=182;
  1074589202;194.138.39.56;2fd3213f2edaf82b27562d28a2a747aa;/
;http://www.seznam.cz;
  1074589233;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/
dp/?id=148;/dp/?id=124;
  1074589245;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/
sb/;/dp/?id=148;
  1074589248;194.138.39.56;2fd3213f2edaf82b27562d28a2a747aa;/
contacts/; /;
  1074589290;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/
sb/;/sb/;
```

Fig. 4. Part of the web log

Data Preparation. In the first step we identified particular users by the session ID and we created a file containing sequences of visited pages for each user. So for the log file shown in Fig. 4, we created (for the user *1993441e8a0a4d7a4407ed9554b64ed1*) the page-type sequence (session)

`start, dp, dp, sb, sb, end`

and the page-content sequence (session)

`start, 124, 148, end.`

In this example, *dp* denotes "detail of product", *sb* denotes "shopping basket", 124 denotes "loud-speaker " and 148 denotes "DVD player". Note, that we added two pages to each sequence found in the data, the `start` page and the `end` page.

In the second step we excluded sessions of length 1. This is a general recommendation as sessions of length 1 are usually created by web bots crawling the web space and collecting pages. Moreover, as we are interested in predicting next page in a click-stream, we need sequences of length 2 and more. This reduces the

[2] Because there was no possibility how to identify two sessions of one user we consider each session with distinct ID as one user.

number of sessions (sequences) to 200 000; the average number of visits by one user (the average length of session) was 16; the median was 8 and modus 2.

In the third step of data preparation, we created 2 basic files that entered to the analyses. In the first one, there were saved sequences of page type regardless of the product offered on this page. In the second one, there were saved sequences of page content (sequences of products) regardless of the page type (pages without products were excluded from these sequences). During this step, the products were divided to 30 categories.

Modeling. The modeling consisted in searching for rules using the algorithm described in section 2. The algorithm for searching was running repeatedly with various input parameters.

The first set of experiments was performed for the sequences of page types. Among the obtained results we can find the rule

```
dp, sb -> sb (Ant: 5174; AntSuc: 4801; P: 93%)
```

that covers the session of user *1993441e8a0a4d7a4407ed9554b64ed1* (see above). Another interesting rules were e.g.

```
 ct -> end  (Ant: 5502; AntSuc: 1759; P: 32%)
faq -> help (Ant: 594; AntSuc: 127; P: 21%)
```

In the listing above, *ct* stands for "contact", *Ant* stands for ||*Ant*|| and *AntSuc* stands for ||*Ant*//*Suc*||.

The second set of experiments was performed for the sequences of products (page contents). For the request of the data provider, we generated all rules of length 2 for the sequences of products. From these rules passes of users between products were seen very well. Examples of such rules are:

```
loud-speakers -> video & DVD (Ant: 14840, AntSuc: 3785, P: 0.26)
data cables -> telephones (Ant: 2560, AntSuc: 565, P: 0.22)
PC peripheries -> telephones (Ant: 8671, AntSuc: 1823, P: 0.21)
```

The models obtained in both sets of experiments can directly be used to predict the behavior of an user. So eg. for a sequence of pages `dp, sb` the system will predict *sb* as the next page, and for the sequence of products `loud-speakers` the system will predict `video & DVD`.

We have run our algorithm repeatedly for both page sequences (first set of experiments) and product sequences (second set of experiments). We also analyzed data about transitions between different internet shops operated by the data provider. The obtained accuracy when predicting next page in a click-stream was about 60% for page sequences and shop transitions (which we think is not bad) but only about 20% for product sequences. The parameters that mostly contribute to improvement of the accuracy were the significance level and the frequency. On the contrary, the max. length of a sequence (l_{max}) that mostly affects the number of found rules doesn't improve the accuracy.

Evaluation and Deployment. We presented our results to the data provider and together we choose those which were unknown and interesting for them (about ½ of presented knowledge). We made some small analyses directly on the meeting to check some hypotheses of the data provider that were not contained in our results. The data provider then used the found knowledge to modify pages of internet shop to engage consumers and offer them more comfort.

4 Conclusions

We present a new set-covering algorithm to learn decision rules for web usage mining. Although we developed our algorithm within a project of click-stream analysis, it can be used more generally, to predict occurrence of an event in a sequence of any types of events (e.g. transactions on banking accounts, or network intrusion). Our experiments with a real data of an internet shop showed usefulness of the proposed approach. In our future work, we plan to develop an algorithm for learning compositional rules for the same type of data and to compare both approaches.

5 Acknowledgments

The work presented in the paper is supported by the grant GACR 201/03/1318.

References

1. Bruha I., Kočková S. A support for decision making: Cost-sensitive learning system. *Artificial Intelligence in Medicine*, 6 (1994), 67-82.
2. Clark P., Niblett T. The CN2 induction algorithm. *Machine Learning*, 3 (1989), 261-283.
3. Cooley R., Tan P.N., Srivastava ,J. Discovery of interesting usage patterns from web data. Tech. Rep. TR 99-022, Univ. of Minnesota, 1999.
4. Kaufman K.A., Michalski R.S.: Learning from inconsistent and noisy data: The AQ18 approach. In: Proc 11th Int. Symposium on Methodologies for Intelligent Systems, 1999.
5. Kosala R., Blockeel H.: Web Mining Research: A Survey. *SIGKDD Explorations,* Vol. 2 Issue 1, 2000.
6. Spiliopoulou M., Faulstich, L. WUM: A tool for web utilization analysis. In Proc. EDBT Workshop WebDB'98, Springer LNCS 1590, 1999.
7. Srivastava J., Cooley R., Deshpande M., Tan P-N. Web Usage Mining: Discovery and Applications of Usage Paterns from Web Data. *SIGKDD Explorations*, Vol. 1 Issue 2, 2000.
8. Zaiane O., Han J. WebML: Querying the World-Wide Web for resources and knowledge. In: Workshop on Web Information and Data Management WIDM'98, Bethesda, 1998, 9-12.
9. Zaine O., Xin M., Han, J. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In: Advances in Digital Libraries, 1998.

Complexity of Finding Optimal Observation Strategies for Bayesian Network Models

Václav Lín

Faculty of Informatics and Statistics,
University of Economics, Prague, Czech Republic
xlinv05@vse.cz

Abstract. One of the major strengths of Bayesian networks is the ability to efficiently update the model when new information pertaining to the modeled system is gained. However, gaining information about the system is often associated with a certain cost. We are then faced with the problem of gaining as much information as possible without exceeding a specific limit on the total cost. This problem is solved by designing an optimal observation strategy, i.e., an ordering of observations of system variables that most decreases our uncertainty concerning the system state. We formalize this situation as a combinatorial problem and provide a lower bound of its complexity. The problem turns out to be $coNP$-hard.

1 Introduction

Bayesian networks [2] are probabilistic graphical models that have been used to represent real-world systems comprising uncertainty. One important advantage of Bayesian networks is the ability to efficiently update the model when new information about the system is gained, i.e., when we observe that some variable of the system is in certain state.

Gaining information about the system often takes some effort or is associated with a cost. We are then faced with the problem of gaining as much information as possible without exceeding a pre-defined limit on the total cost. Solution of the problem is determined by a decision strategy (or *observation strategy*) that prescribes in what order the variables within the system should be observed. In our present framework, an observation strategy is considered *optimal* if it (on average) yields the most information and never prescribes a sequence of observations that exceeds the cost limit.

In this paper we provide a lower bound of computational complexity of the problem of finding optimal observation strategies. We show that the problem is at least *coNP*-hard. As a consequence, it is very unlikely that there is a polynomial-time solution algorithm for the problem.

Section 1.1 provides an overview of some basic facts concerning Bayesian networks, and in Section 1.2 the problem of finding optimal observation strategies is formalized. In Section 1.3 we briefly summarize some recent research activities that are closely related to our work. Section 2 introduces complexity-theoretic notions that we need to establish our main result in Section 3. The paper concludes with a brief note on further work in Section 4.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 266–273, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

1.1 Bayesian Networks

The *Bayesian network* consists of a (finite) set $\mathscr{X}$ of random variables and a set of directed edges between the variables. Each variable $X_i \in \mathscr{X}$ has a finite range $\mathbb{X}_i$ of mutually exclusive states. The variables together with the directed edges constitute a directed acyclic graph (DAG). To each variable X with parents $pa(X)$, a conditional probability table (CPT) $P(X|pa(X))$ is attached. We say that two variables X_i and X_j are *d-separated* by a set $\mathscr{Z}$ of variables if for all *undirected* paths[1] between X_i and X_j, there is a variable Z such that edges either

- do not meet head-to-head in Z ($\rightarrow Z \leftarrow$) and $Z \in \mathscr{Z}$, or
- meet $\rightarrow Z \leftarrow$ and neither Z nor any of its descendants belongs to $\mathscr{Z}$.

The DAG structure implies conditional independence relations between the variables. By definition, it is required that all variables X_i and X_j that are d-separated by a set $\mathscr{Z}$ of variables are conditionally independent given $\mathscr{Z}$ in joint probability distribution P represented by the Bayesian network model (in symbols, $X_i \perp\!\!\!\perp_P X_j|\mathscr{Z}$). The distribution that satisfies this condition, and that has its CPTs equal to $P(X|pa(X))$, $X \in \mathcal{X}$, is unique and is given by

$$P(\mathscr{X}) = \prod_{X \in \mathscr{X}} P(X|pa(X)) \quad .$$

An example of a Bayesian network is given in Fig. 1

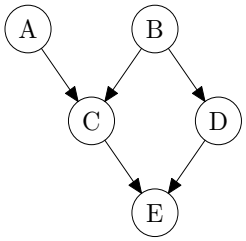


Fig. 1. Example of a Bayesian network over five variables A, B, C, D. The probability distributions associated to the nodes of the DAG are $P(A)$, $P(B)$, $P(C|A,B)$, $P(D|B)$ and $P(E|C,D)$. Product of these distributions specifies the joint distribution $P(A,B,C,D,E)$. Some of the (conditional) independence statements are, e.g., $A \perp\!\!\!\perp_P B$, $C \perp\!\!\!\perp_P D|B$, $E \perp\!\!\!\perp_P B|(C,D)$.

1.2 Optimal Observation Strategies for Bayesian Networks

In this section we formalize the problem of finding optimal observation strategy for a stochastic system modeled by a Bayesian network. The formal framework that we develop is a mild generalization of the framework used in [6, 5].

[1] By an undirected path we mean a path in the undirected version of the DAG.

Assume that the system of interest is described by a set $\mathscr{X}$ of discrete random variables. Values of some $X_i \in \mathscr{X}$ can be directly observed – such X_i are called *observable*. The set of all observable $X_i \in \mathscr{X}$ is denoted $\mathscr{O}$. To each $X_i \in \mathscr{O}$ a non-negative *observation cost* $c_i \in \mathfrak{R}$ is associated. We assume that the joint probability distribution $P(\mathscr{X})$ exists. For practical applications, it is useful, but not necessary, to assume that $P(\mathscr{X})$ is represented by a Bayesian network.

An *observation strategy* (or just a *strategy* for short) $\mathbf{s}$ is represented by a directed tree where the non-leaf nodes are labeled with elements of $\mathscr{O}$, and the directed edges are labeled by states of the elements of $\mathscr{O}$. If a node i corresponds to $X_i \in \mathscr{O}$, then there is exactly one directed edge going out of i for each element of $\mathbb{X}_i$. There are no other edges. Each path from the root to a leaf of $\mathbf{s}$ corresponds to a particular sequence of observations. We require that no variable is observed more than once along any path from root to leaf. The evidence compiled along the path from the root to leaf l is denoted by $\mathbf{e}_l$, the total cost of all observations along the path is denoted by t_l. The set of leaf nodes of $\mathbf{s}$ is denoted by $\mathcal{L}(\mathbf{s})$. A simple example of observation strategy is in shown Fig. 2.

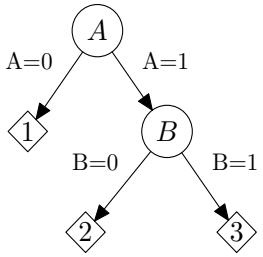


Fig. 2. A possible strategy over Boolean observable variables A and B given by the following decision rules: (1) Observe A first. (2) If $A = 0$ then terminate, else observe B. Evidence corresponding to the leaf nodes is $\mathbf{e}_1 = \{A = 0\}$, $\mathbf{e}_2 = \{A = 1, B = 0\}$, $\mathbf{e}_3 = \{A = 1, B = 1\}$. Assuming the observation costs to be $c_A = 2$, $c_B = 1$, we have $t_1 = 2$, $t_2 = t_3 = 3$.

The goal of the observations is to decrease our uncertainty about the set $\mathscr{Y} \subseteq \mathscr{X}$ of *hypothesis varibles* (where $\mathscr{Y}$ and $\mathscr{O}$ are not necessarily disjoint and $\mathscr{Y} \cup \mathscr{O}$ is not necessarily equal to $\mathscr{X}$). The uncertainty concerning $\mathscr{Y} = \{Y_1, \ldots, Y_k\}$ is measured by the Shannon entropy of the marginal distribution $P(\mathscr{Y})$. Shannon entropy of $P(\mathscr{Y})$ given evidence $\mathbf{e}$ is

$$H(P(\mathscr{Y}|\mathbf{e})) = -\sum_{y_1,\ldots,y_k} P(Y_1 = y_1, \ldots, Y_k = y_k|\mathbf{e}) \log P(Y_1 = y_1, \ldots, Y_k = y_k|\mathbf{e}) \ .$$

For a strategy $\mathbf{s}$, we define *expected entropy*

$$E_H(\mathbf{s}) = \sum_{l \in \mathcal{L}(\mathbf{s})} P(\mathbf{e}_l) H(P(\mathscr{Y}|\mathbf{e}_l)) \ .$$

Here, $P(\mathbf{e}_l)$ denotes the probability of getting from the root of $\mathbf{s}$ to leaf l. The *maximal observation cost* of a strategy $\mathbf{s}$ is

$$MOC(\mathbf{s}) = \max_{l \in \mathcal{L}(\mathbf{s})} t_l \ .$$

The problem of finding optimal observation strategies can be stated as follows: Given $\mathscr{X}$, $P(\mathscr{X})$, $\mathscr{O} \subseteq \mathscr{X}$, $\mathscr{Y} \subseteq \mathscr{X}$ and $K \in \mathfrak{R}$, find an observation strategy $\mathbf{s}^\star$ such that

$$\mathbf{s}^\star = \arg\min_{\mathbf{s} \in \mathbf{S}_K} E_H(\mathbf{s}) \ ,$$

where $\mathbf{S}_K = \{\mathbf{s} : MOC(\mathbf{s}) \leq K\}$. Such a strategy $\mathbf{s}^\star$ is called *optimal*.

This definition is in accord with the intuition outlined above. We are looking only for strategies that never prescribe a sequence of observations more expensive than the limit K. Among these strategies, we look for one that (on average) most decreases the entropy of $P(\mathscr{Y})$.

1.3 Related Work

Our work is closely related to automated construction of *adaptive tests* using Bayesian networks, as developed in [6, 5]. In this framework, a Bayesian network is used for modeling skills, knowledge and misconceptions of an examinee, together with the available test questions. The goal is to construct a test (i.e., an ordering of questions) that would reveal the most information about the examinee. The test is *adaptive* — answers to questions influence selection of the next question to be asked. In this paper, we study basically the same problem, i.e., the problem of constructing an optimal "question-asking" strategy. The only difference is that we take our limited resources and the costs of "questions" into account.

The cost of performing observations is also taken into account in systems for *decision-theoretic troubleshooting* using Bayesian networks [1, 3]. There, a Bayesian network models possible faults, repair actions and test actions (questions) associated to some faulty technical equipment, e.g. a car or a printer. The goal is to find an optimal *troubleshooting strategy* – a strategy of choosing test and repair actions that fix the faulty device at the lowest cost. This is very similar to our approach. The difference is twofold:

- we have only test actions,
- our goal is to gather as much information about the modeled system as possible, but not to "fix" it.

Computational complexity of troubleshooting was studied in [7]. However, the results are not directly applicable to our problem.

2 Complexity Classes *coNP* and *NP*

We assume the reader is familiar with classes P and NP, and with the concepts of NP-hardness and NP-completeness, as well as with the concept of polynomial reductions of decision problems [4].

coNP is a complexity class that is *complementary* to *NP*. Informally, this means that for each decision problem $A \in NP$ there is a decision problem $\overline{A} \in coNP$, called a *complement* of A, such that the answer to $\overline{A}$ is 'yes' whenever the answer to A is 'no', and vice versa. It is strongly believed that $coNP \neq NP$, eventhough no proof has yet been given. The following statement holds:

Proposition 1 ([4]). *If A is* NP*-complete, then $\overline{A}$ is* coNP*-complete.*

Relation of the two classes is perhaps best described by their complete problems. (We will use basic propositional logic terminology. For the reader's convenience, we include a brief overview in the appendix).

Definition 1 (Satisfiability**, [4]).** *Given a propositional formula φ in conjunctive normal form, is φ satisfiable?*

Theorem 1 (Cook's theorem, [4]). *Deciding* Satisfiability *is* NP*-complete. The problem remains* NP*-complete even when each clause of φ is restricted to contain exactly three literals.*

Definition 2 (Validity**).** *Given a propositional formula φ in disjunctive normal form, is φ valid?*

Proposition 2 ([4]). *Deciding* Validity *is* coNP*-complete. The problem remains* coNP*-complete even when each implicant of φ is restricted to contain exactly three literals.*

Finding a polynomial-time algorithm for a *coNP*-complete problem would resolve the notorious $P = NP$ question. Indeed, existence of a polynomial-time algorithm for Validity would imply existence of a polynomial-time algorithm for Satisfiability, and vice versa. Therefore, the following proposition holds.

Proposition 3. P = NP *if and only if* P = coNP.

3 Main Theorem

We define a *decision variant* of the problem of finding an optimal observation strategy and show it is *coNP*-hard.

Definition 3 (Strategy**).** *Given $\mathscr{X}$, $P(\mathscr{X})$, set of observable variables $\mathscr{O} \subseteq \mathscr{X}$, set of hypothesis variables $\mathscr{Y} \subseteq \mathscr{X}$ and $K, G \in \mathfrak{R}$, is there an observation strategy* $\mathbf{s}$ *with $E_H(\mathbf{s}) \leq G$ and $MOC(\mathbf{s}) \leq K$?*

Theorem 2. *Deciding* Strategy *is* coNP*-hard.*

Proof. We will prove this assertion by reduction of the Validity problem. That means that, for any instance of Validity, we construct an instance of Strategy that is computable in polynomial time, and that gives a 'yes' answer if and only if the answer to the corresponding instance of Validity is 'yes'. Once we have defined such a reduction, we know that the time complexity of Strategy must

be at least as high as that of VALIDITY. Since VALIDITY is *coNP*-complete, STRATEGY must be *coNP*-hard. Without a loss of generality, we assume that $P(\mathscr{X})$ is represented by a Bayesian network.

Let φ be the DNF formula of VALIDITY and let it contain v propositional variables and n implicants. Without a loss of generality, we assume that each implicant of φ contains exactly three literals (see Proposition 2) and $v > 0$.

We set $K = n$ and $G = 0$. For each propositional variable x_i we construct a random variable X_i, and for each implicant κ_j we construct a random variable O_j. Then we construct a random variable Y. Finally, if $n > 1$, we construct $n - 1$ random variables $Z_1, \ldots, Z_{n-1}$. All the constructed random variables are Boolean. The O_j's are the only observable variables, and Y is the only hypothesis variable. Cost of all observations is 1.

The DAG structure is specified by the following rules. There is an edge $X_i \rightarrow O_j$ if and only if the variable x_i appears in implicant κ_j. If $n = 1$, there is an edge $O_1 \rightarrow Y$. Else ($n > 1$), there are edges $O_1 \rightarrow Z_1$ and $O_2 \rightarrow Z_1$, and for $2 \leq k < n$ there are edges $Z_{k-1} \rightarrow Z_k$, $O_{k+1} \rightarrow Z_k$. Finally (still assuming $n > 1$), there is edge $Z_{n-1} \rightarrow Y$. There are no other edges. For example, the formula

$$(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3)$$

corresponds to the DAG structure in Fig. 3.

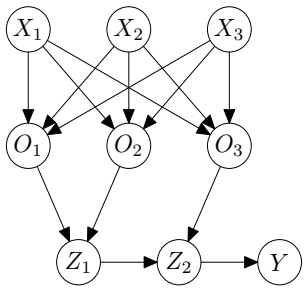


Fig. 3. The DAG for $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3)$.

Probability distributions on the X_i's are uniform, i.e., $P(X_i = 1) = \frac{1}{2}$ for $i = 1, \ldots, v$. Note that each configuration of values assigned to $(X_i)_{i=1}^{v}$ corresponds unambiguously to a truth assignment for φ.

Conditional probability distributions $P(O_j|pa(O_j))$, $j = 1, \ldots, n$, are defined so that $P(O_j = 1) = 1$ whenever the configuration of $pa(O_j)$ corresponds to a satisfying truth assignment for implicant κ_j. With a slight abuse of notation, we may write

$$P(O_j = 1|pa(O_j) = t) = \begin{cases} 1 & \text{if } \kappa_j[t] = 1 \\ 0 & \text{otherwise} \end{cases} ,$$

where the first occurence of t denotes a configuration of $pa(O_j)$ and the second denotes the corresponding truth assignment to κ_j.

Variables Z_k are 'OR-gates': conditional probability distributions $P(Z_k|pa(Z_k))$, $k = 1, \ldots, n-1$, are defined by equations

$$P(Z_k = 1|pa(Z_k) \neq \mathbf{0}) = 1 \ , \qquad P(Z_k = 0|pa(Z_k) = \mathbf{0}) = 1 \ ,$$

where $\mathbf{0}$ is a zero vector.

For Y we define conditional distribution by equations

$$P(Y = 1|pa(Y) = 0) = \frac{1}{2} \ , \qquad P(Y = 1|pa(Y) = 1) = 1 \ .$$

Please note that $pa(Y)$ is O_1 if $n = 1$, otherwise it is Z_{n-1}.

The Z_k's form an OR-gate. Therefore $P(pa(Y) = 1) = 1$ if and only if $P(O_j = 1) = 1$ for at least one j. Consequently, $H(P(Y)) = 0$ if and only if the configuration of values assigned to $(X_i)_{i=1}^v$ corresponds to a satisfying truth assignment for φ.

We need to show that this construction of Strategy from Validity is computable in time polynomial in the length of φ. The length of φ may be measured as the combined length of the implicants, i.e., it is $3n$ since each implicant contains exactly three literals. In our construction, we have created $\mathcal{O}(3n)$ variables X_i, n variables O_j, $n-1$ variables Z_k and one Y. That is in total $\mathcal{O}(5n)$ variables. Sizes of probability tables associated with the variables are constants independent of the length of φ: each $P(X_i)$ has 2 entries, each $P(O_j|pa(O_j))$ has 2^{3+1} entries, each $P(Z_k|pa(Z_k))$ has 2^{2+1} entries, and $P(Y|pa(Y))$ has 2^{1+1} entries.

To conclude the proof, we need to prove that φ is valid if and only if there is a strategy $\mathbf{s}$ with $MOC(\mathbf{s}) \leq K = n$ and $E_H(\mathbf{s}) \leq G = 0$.

$\Rightarrow$) Assume that φ is valid. Consider a strategy $\mathbf{s}$ that consists of observing all the observable variables $O_1, \ldots, O_n$. Its cost is $n = K$. For any truth assignment t, at least one of the implicants of φ is satisfied. Therefore, after all the observations have taken place, for any configuration of $(X_i)_{i=1}^v$ there is some j with $P(O_j = 1|pa(O_j)) = 1$. Consequently the distribution on Y is deterministic for any configuration of values assigned to $(X_i)_{i=1}^v$ and $E_H(\mathbf{s}) = 0$ as required.

$\Leftarrow$) Assume that φ is not valid. Then there exists a truth assignment t with $\kappa_j[t] = 0$. For such a t and for any evidence $\mathbf{e}$ obtained by observing an arbitrary subset of $\{O_1, \ldots, O_n\}$, $P(pa(Y) = 0|\mathbf{e}, (X_i)_{i=1}^v = t) = 1$. Since $P((X_i)_{i=1}^v = t) \neq 0$, we have

$$P(pa(Y) = 0|\mathbf{e}) = \sum_t P(pa(Y) = 0|\mathbf{e}, (X_i)_{i=1}^v = t) \cdot P((X_i)_{i=1}^v = t) \neq 0.$$

Consequently $P(pa(Y) = 1|\mathbf{e}) < 1$ for any evidence $\mathbf{e}$. It follows that there is no way of observing $O_1, \ldots, O_n$ that would make sure that the probability distribution on Y is deterministic. Consequently, any observation strategy $\mathbf{s}$ will have $E_H(\mathbf{s}) > 0$. ■

4 Conclusions and Further Work

By Theorem 2 and Proposition 3, there is no polynomial-time algorithm for finding optimal observation strategies unless $P = NP$. In our further research, we will study heuristic and approximate algorithms for the problem.

Acknowledgement. I would like to thank Jiří Vomlel for introducing to me the problem of finding optimal observation strategies, for careful reading of drafts of the paper, and for many helpful comments. My thanks are also due to Nikola Kaspříková for helping me out with the METAPOST grahics. I also thank anonymous reviewers of the paper for their comments.

References

1. Heckerman, D., Breese, J. S., Rommelse, K., "Decision-theoretic Troubleshooting," Communications of the ACM, 38(3), 1995.
2. Jensen, F. V., *Bayesian Networks and Decision Graphs*, Springer Verlag, New York, 2001.
3. Jensen, F. V. et al., "The SASCO Methodology for Troubleshooting Complex Systems," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2001.
4. Papadimitriou, C. H., *Computational Complexity*, Addison-Wesley Publishing Company, Reading, Pennsylvania, 1994.
5. Vomlel, J., "Bayesian Networks in Educational Testing," International Journal of Uncertainty, Fuzziness and Knowledge Based Systems, Vol. 12, Supplementary Issue 1, 2004, pp. 83-100.
6. Vomlel, J., "Building Adaptive Tests using Bayesian networks," Kybernetika, Vol. 40, Number 3, 2004, pp. 333 - 348.
7. Vomlelová, M., "Complexity of Decision-Theoretic Troubleshooting," International Journal of Intelligent Systems, Vol. 18, 2003, pp. 267-277.

A Propositional Logic Terminology

We only briefly review propositional logic terminology frequently used in the paper, more information can be found in any introductory textbook on mathematical logic. *Propositional variable* is a variable x that can attain one of two possible values: 1 (truth) or 0 (falsity). *Literal* is an expression of the form x or $\neg x$. *Clause* is a disjunction of literals, *implicant* is a conjunction of literals. Propositional formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses. Propositional formula is in *disjunctive normal form* (DNF) if it is a disjunction of implicants. *Truth assignment* is a mapping of propositional variables to the set $\{0, 1\}$. Truth value of formula φ under truth assignment t is denoted $\varphi[t]$. Propositional formula φ is *satisfiable* if there is a truth assignment t such that $\varphi[t] = 1$ (such a truth assignment is called *satisfying*). Propositional formula φ is *unsatisfiable* if there is no satisfying truth assignment for φ. Propositional formula φ is *valid* if $\varphi[t] = 1$ for any truth assignment t.

Finite State Automata and Image Storage

Marian Mindek

Katedra informatiky, FEI, VŠB Technická Univerzita Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba

marian.mindek@vsb.cz

Abstract. In this paper we introduce finite automata as a tool for specification and compression of a gray-scale image. We describe how to link together resultant automata and how such a principle can be used for creation of image database with added value.

Keywords: finite automata, image compression, image database

1 Introduction

Karel Culik II and Vladimir Valenta have proposed [1,2] fractal-coding technique, which is based on automata theory. We are issuing that idea and in first chapter, we will describe algorithm for gray-scale pictures based on simple solution for bi-level pictures. In another chapter we will demonstrate, how to use automata for compression pictures. Last, we will show background for creating database with added value for storing pictures.

2 Finite automata

For understanding of the following, we allege only necessary background. If you want know more about automata theory as a tool for specifying image, you can find it in [3].

A digitized image of the finite resolution *m x n* consists of *m x n* pixels each of which takes a Boolean value (1 for black, 0 for white) for bi-level image, or real value (practically digitized to an integer value 0 and 256) for a gray-scale image.

Here we will consider square images of resolution 2^n x 2^n. In order to facilitate the application of finite automata to image description we will assign each pixel at 2^n x 2^n resolution a word of length *n* over the alphabet $\Sigma=\{0,1,2,3\}$ as its address. A pixel at 2^n x 2^n resolution corresponds to a sub-square of size 2^{-n} of the unit square. We choose ε as the address of the whole unit square. Single digits as shown in Fig. 1 on the left address its quadrants. The four sub-square of the square with address *w* are addressed *w0, w1, w2* and *w3,* recursively. Address of all the sub-square (pixels) of resolution 4 x 4 is shown in Fig. 1, middle. The sub-square (pixel) with address *3203* is shown on the right of Fig. 1.

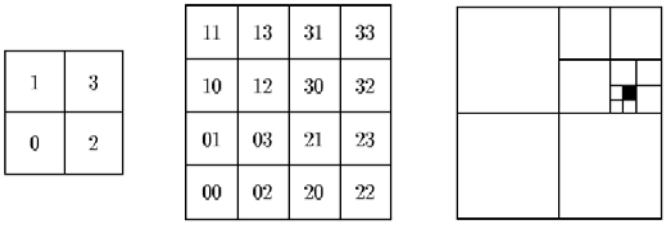


Figure 1. The addresses of the quadrants, of the sub-square of resolution 4 x 4, and the sub-square specified by the string 3203.

In order to specify a black and white image of resolution 2^m x 2^m, we need to specify a language $L \subseteq \Sigma^m$. Frequently, it is useful to consider multi-resolution images, which are images that is simultaneously specified for all possible resolution, usually in some compatible way. (We denote Σ^m the set of all words over Σ of the length m, by Σ^* the set of all words over Σ)

In our notation a bi-level multi-resolution image is specified by a language $L \subseteq \Sigma^*$, $\Sigma=\{0,1,2,3\}$, i.e. the set of addresses of all the black squares, at any resolution. Now, we are ready to give some examples. We assume that the reader is familiar with the elementary facts about finite automata and regular sets see e.g. [4].

A word in the input alphabet is accepted by the automaton if exist labeled path from the initial state to the final state. The set (language accepted by automaton A) is denoted $L(A)$.

Example. The 2 x 2 chess-board in Fig. 2 look the same for all resolution. The multi-resolution specification is the regular set $\{1,2\}\Sigma^*$. The 8 x 8 chess-board in Fig. 2 too is described by the regular set $\Sigma^2\{1,2\}\Sigma^*$ or by automaton A of Fig. 2.

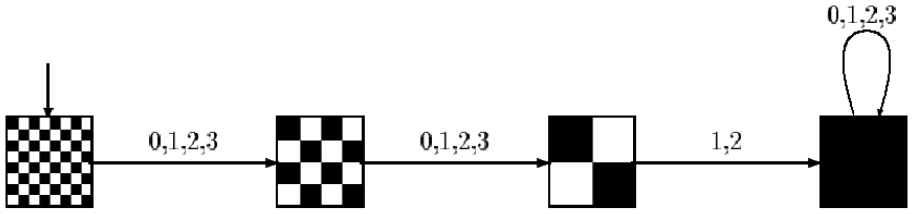


Figure 2. Finite automaton A defining the 8 x 8 chess-board.

Notice that here we used the fact that the regular expression $\Sigma^2\{1,2\}\Sigma^*$ is the concatenation of two regular expression Σ^2 and $\{1,2\}\Sigma^*$.

Example. Clearly, L_1= $\{1,2\}^*0$ are addresses of the infinitely many squares illustrated at the left of Fig. 3. If we place the completely black square defined by $L_2=\Sigma^*$ into all these squares we get the image specified by the concatenation $L_1L_2=\{1,2\}^*0\Sigma^*$ which is the triangle shown in the middle of Fig. 3.

Figure 3. The squares specified by {1,2}*0, a triangle defined by {1,2}*0Σ*, and the corresponding automaton.

Example. By placing the triangle $L = L_1L_2$ from the previous example into all the squares with addresses L_3={1,2,3}*0 we get the image L_3L={1,2,3}*0{1,2}*0Σ* shown at the left of Fig. 4.

Zooming [3] is easily implemented for images represented by regular sets and is very important for image compression shown in next section.

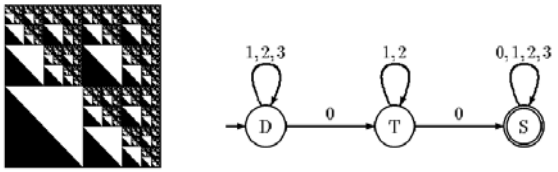


Figure 4. The diminishing triangles defined by {1,2,3}*0{1,2}*0Σ*, and the corresponding automaton.

We have just shown that a necessary condition for black and white multi-resolution image to be represented by a regular set is that it must have only a finite number of different sub-images in all the sub-squares with addresses from Σ*. We will show that this condition is also sufficient. Therefore, images that can be perfectly (i.e. with infinite precision) described by regular expressions (finite automata) are images of regular or fractal character. Any image can by approximated by a regular expression however; an approximation with a smaller error might require a larger automaton.

Now, we will give a theoretical procedure which, given a multi-resolution image, finds a finite automaton perfectly specifying it, if such an automaton exists.

Procedure *Construct Automaton*
For given image I, we denote I_w the zoomed part of I in the square addressed w. The image represented by state number x is denoted by u_x.

1. $i=j=0$.
2. Create state 0 and assign $u_0=I$.
3. Assume $u_i=I_w$. Process state i, that is for k=0,1,2,3 do:
   If $I_{wk}=u_q$ for some state q, then create an edge labeled k from state i to state q; otherwise assign $j=j+1$, $u_j=I_{wk}$,and create an edge labeled k from state i to the new state j,

The procedure *Construct Automaton* terminates if exists any automaton that perfectly specifies the given image and produces a deterministic automaton with the minimal number of states. Our algorithm for gray-scale image is based on this procedure, but it will use evaluated finite automata (as like WFA) introduced in the section 3 and only replacing black and white color to 256 color (or grayness) image, no creating loop and add some option for setup compression.

Procedure for reconstruction picture from automaton is very simple, than we describe only procedure *Reconstruct Image* for compressed pictures at next section.

3 Image compression

In this chapter, we will demonstrate in brief a method for image compression / decompression applicable on construction of resultant images storage. Our algorithm is based on algorithm shown in section 2. There lead 4 edges from each node at most and they are labeled with numbers representing image part. Every state stores information of average grayness of sub-square represented thereby state.

Procedure *Construct Automaton for Compression*
For given image I, we denote I_w the zoomed part of I in the square addressed w. The image represented by state number x is denoted by u_x.

1. $i=j=0$.
2. Create state *0* and assign $u_0=I$. (Image represented by empty word) and define average grayness of image I.
3. Assume $u_i=I_w$. Process state i, that is for k=0,1,2,3 do:
   If $I_{wk}=u_q$ (with small error) or if the image I_{wk} can be expressed as a part or expanded part of the image u_q for some state q, then create an edge labeled k from state i to state q;
   otherwise assign $j=j+1$, $u_j=I_{wk}$,and create an edge labeled k from state i to the new state j,
4. if $i=j$, that is all states have been processed, stop;
   otherwise $i=i+1$, go to 3.

The procedure *Construct Automaton for compression* terminates if there exists an automaton that perfectly (or with small-defined error) specifies the given image and produces a deterministic automaton with the minimal number of states. The number of state can be small reduced or extended by changing error or do tolerance for average grayness of image part. For reconstruct image from automata, we propose follow recursively algorithm.

Procedure *Reconstruct Image*
For given automata A, we make image I_w the zoomed part of I in the square addressed w. The image represented by state number x is denoted by u_x

1. Assign the initial state q_o to the image represented by the empty word, that is, to the whole image I, and define $i(q_o)$=1, $t(q_o)$=∅(ε), the average grayness of the image *I*, which we change to computed color, if we wont that.
2. Recursively, for a state q assign to square specified by a string u, consider four sub-square specified by a string *u0, u1, u2, u3*. Denote the image in square by I_{ua}. If the image is everywhere $t(q_o)$ and word has shorter then requested, assign a new input state q that representative image specified by a input word uX where X is denoted part of image. Otherwise, assign a new input state $q(uY)$ where Y is a next part of image.
3. Repeat step 3 for each state, and stop if no founded new input state or input word is an equal to requested.

The procedure *Reconstruct Image* was stop for every automata computed by a procedure *Construct Automaton (for compression),* or other similar algorithm.

For demonstration: all states are marked on Fig. 5, where darkness color is prime states and white color have latest states.

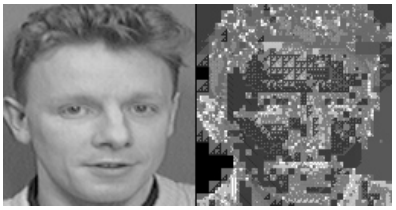


Figure 5. States of computed automata represented by color (on right-hand).

4 Image storage

4.1 Prevailing approach

There exist many methods for assembling images represented by final state automata. We will depict one of the betters, namely the one, which assembles resultant automata in direction from leaf node to the root. There are some example images (*Image1* and *Image2*) and their representation by means of final state automata computed by method *Construct Automaton for compression,* on Fig. 6 and 7. For simplicity were voted by acclamation black and white pictures with resolution 8x8 pixel. Every state of resultant automata has assigned average grayness value of a picture part, which it represents. Values are in intervals 0-255, where 0 represents black color and 255 white.

It is clear, that the following two images (on Fig. 6 and Fig. 7) have common some parts, for lucidity see the next picture - Fig. 8. Concrete: sub-square 1 is the same at both images, sub-square 0 in image1 is the same as sub-square 0 and 2 in

image2, sub-square 3 in image1 is similar to sub-square with address 23 in image2, etc.

Every common part is converted to the same state in the resultant automata, as we can see on Fig. 9. That way the lossless compression is performed. Moreover, an additional information can be obtained from the structure of the resultant automaton. For example, the similarity of saved pictures, common shapes, and alternatively parts of the picture. We can also follow-up similar pictures or the similar parts of various pictures.

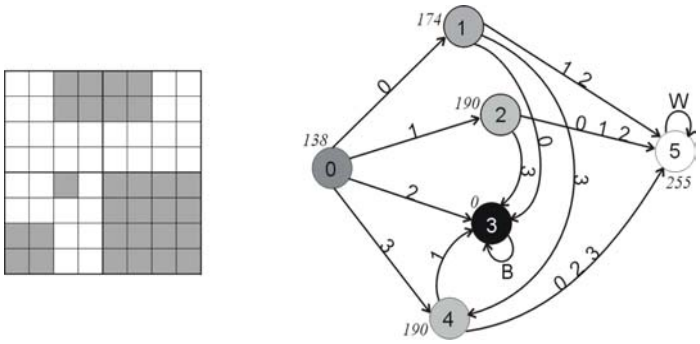


Figure 6. Example *image1*, and the corresponding automaton.

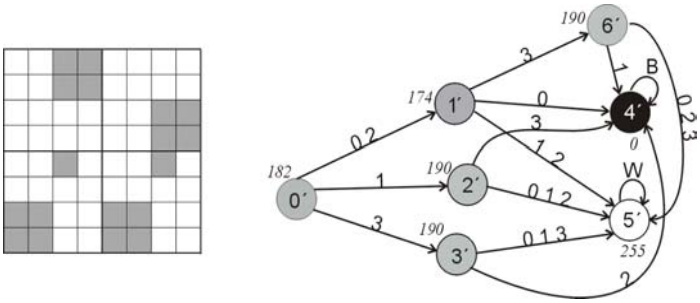


Figure 7. Example *image2*, and the corresponding automaton.

The algorithm for assembling resultant automata is very simple.

Procedure *Composition Automaton for storage*
For given automaton *A1*, and stored automaton *A2* we make automaton *A* that represent booth image.

1. Assign the some final state q_x from *A1* to the correspond state in stored automata *A2*. Otherwise if is not present, assign a new state and take q_{x+1} from *A1*.
2. Process all other state from *A1*:
   If not ∃ edge from state q_i labeled with same word *w* as edge from correspond state in stored automaton create new edge labeled *w* to new state *i´*;

The procedure *Composition Automaton for storage* stops for every automaton computed by a procedure *Construct Automaton (for compression),* or other similar algorithm. Resultant automaton is smaller than original automata and contains interesting information from both automata plus common interesting information. In the extreme situation the resultant automata may be bigger than original, but if we will store more and more images, the resultant automaton becomes bigger slower. In other extreme, we can store two (or more) almost the same images in automaton with non-increasing count of states. It is clear, that when we are storing images from the one family (medical super sound radiograph, X-ray, pictures of building, etc.), we can obtain much interesting information about our stored images and save more space.

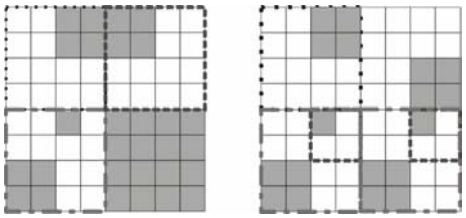


Figure 8. Example *image1* and *image2* with marked common parts.

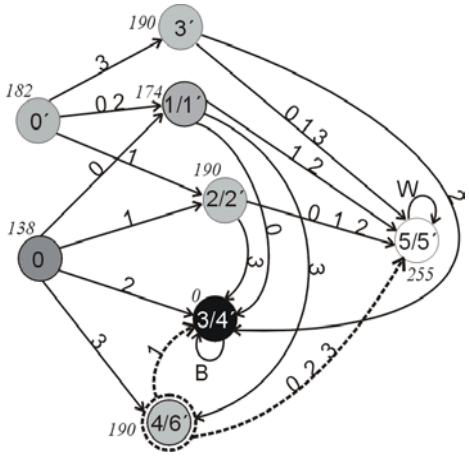


Figure 9. Resultant automaton.

4.2 Object oriented approach

We can use previously approach for more sophisticated preview to this problem, if we have images that have some equivalent parts (like as building tracing, radiogram,

equivalent. Procedure *Construct Automaton for Object Oriented Approach* looks as simple algorithm *Construct Automaton*, with some changes, of course.

If we have a vector image, we mostly know its equivalent parts. If we do not know that, we can use some known algorithm for searching of the same parts. Now, if the processing procedure locates a part of the processed image, which is the same to any other part processed previously, the procedure creates an edge from actual state to initial state of the appropriate automaton and continues in processing. Naturally, it is possible to locate more than one same part (also transparently) in processed image.

Produced automaton describes our image with the knowledge of the position and the count of the same parts. These principles get us a solid background for creating image database with high-included value. We can search images by its parts; every common part can be easily changed; we can save a lot of space; we can transfer through the network only the image parts we are interested in, or only a set of interesting images, etc. Subsequently, we can use the resultant automaton the same way as the classical automaton for raster image. It comes to this, that it is not necessary to use special algorithm for image reconstruction or a special program for drawing vector images.

5 Conclusions

In this paper, we have proposed an alternative solution for image compression and their storage. We described two possible approaches for working with automaton as storage of many images. This resultant automaton is able to underlie the creation of an image database with added value and to make easier the search, collation and working with images. This method is based on a finite automata compression, which can be lossless or loss (both types have high predicate ability about stored images and spare space and network communication).

References

1. K. Culik II and V. Valenta. Finite automata based compression of bi-level and Simple Color Images.
2. K. Culik II and J. Kari. Image compression Using Weighted Finite Automata, in *Fractal Image Compression: Theory a Techniques,* Ed. Yuval Fisher, Springer Verlag, pp 243-258 (1994)
3. Marian Mindek. Finite State Automata and Image Recognition: DATESO 2004, Ed. V. Snášel, J. Pokorný, K. Richta, pp 132-143 (2004), ISBN: 80-248-0457-3
4. J.E.Hopcroft and J.D.Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley (1979).

Klasifikace XML dokumentů

Martin Procházka, Jan Blaťák

Laboratoř vyhledávání znalostí
Fakulta informatiky
Masarykova Univerzita v Brně, Česká republika
{xproch11,xblatak}@fi.muni.cz

Abstrakt. V tomto článku prezentujeme novou metodu pro klasifikaci XML dokumentů, která využívá nejen vlastní data uložená v dokumentu, ale také jeho strukturu. Přitom však nevyžaduje dodatečné informace jako XML schéma nebo DTD. Je navržena nová metoda pro transformaci XML dat do podoby jediné tabulky, kterou lze poté zpracovat stávajícími systémy strojového učení. Uvedeme analýzu výsledků experimentů na dokumentech vytvořených z Internet Movie Database (IMDb).

Klíčová slova: XML, XML mining, schemaless XML classification

1 Úvod

XML (eXtensible Markup Language) [9] je otevřený standard pro definování vlastního strukturovaného jazyka (tzv. XML dialektu), představujícího formát pro ukládání informací. Tuto definici představuje XML schéma nebo DTD soubor, který se pak využívá pro ověření správnosti (validaci) zpracovávaného dokumentu. Data uložená v XML pak těží zejména z výhod otevřených standartů a snadné výměny mezi aplikacemi. XML dokumenty pak navíc můžeme reprezentovat pomocí stromové struktury, která umožňuje členit dokument na jednotlivé části a poskytuje další užitečné informace. Protože XML umožňuje modelovat složité závislosti, může být použito i pro ukládání dat s členitou strukturou.

Nabízí se dvě možné cesty jak pracovat s XML daty: transformovat je do podoby vhodné pro nějaký zavedený systém pro dolování znalostí (hovoříme pak o tzv. propozičních datech), nebo adaptovat stávající systémy tak, aby mohly zpracovávat přímo XML dokumenty. V tomto článku podáme přehled stávajících metod pro dolování v XML a uvedeme novou metodu pro transformaci XML dokumentů a jejich klasifikaci. Metodu experimentálně ověříme na datech z internetové filmové databáze (Internet Movie Database, IMDb[1]) a porovnáme ji s předešlými přístupy.

2 Dolování v XML

Dolování v XML bylo až donedávna neprozkoumanou oblastí a dalo by se říci, že tomu tak stále ještě je. XML často zpracováváme tak, že provedeme extrakci

[1] http://www.imdb.com

vlastních dat na základě našich potřeb a znalosti konkrétního dialektu. Získáme jedinou tabulku, kterou předložíme propozičnímu učícímu systému. Hovoříme pak o tzv. propozicionalizaci [5]. Převod dat však můžeme do jisté míry provést automaticky, např. pomocí systémů pro multirelační dolování znalostí [2]. Vzhledem ke stále rostoucímu počtu XML dokumentů však roste potřeba metod, které pracují přímo s XML dokumenty a využívají všech informací, které jsou v nich obsaženy – tedy i těch o struktuře.

Pro dolování ve struktuře XML navrhl Termier a kol. [7] algoritmus TreeFinder, který hledá pomocí upraveného algoritmu Apriori [1] nejčastěji se vyskytující nejspecifičtější podstromy. Ve výsledném podstromu jsou zachovány tranzitivní vztahy předek–potomek ale ne nutně rodič–dítě. Algoritmus nejprve z každého vstupního XML dokumentu vytvoří transakci, která obsahuje všechny možné kombinace (element–element). Elementy jsou přitom buď ve vztahu rodič–dítě a nebo předek–potomek. V této množině se pomocí AprioriTree naleznou nejčastější relace a z nich se znovu sestaví nejméně obecný strom.

Zaki a kol. [10] navrhl metodu XRules pro klasifikaci XML dokumentů. Vychází z algoritmu TreeFinder, na jehož základě je vytvořen algoritmus XMiner, pro hledání tzv. klasifikačních asociačních pravidel [6] (asociační pravidlo s identifikátorem třídy v závěru) v XML dokumentech. Z pravidel, která jsou splněna pro klasifikovanou instanci, se vybírá podmnožina se shodnou třídou a nejvyšším *kombinovaným efektem* (pokrytí, spolehlivost a korelace). Nevýhodou tohoto přístupu pak může být fakt, že celá klasifikace je prováděna pouze na základě struktury XML.

Proto Theobald a kol. [8] vyvinul metodu převodu XML na propoziční data, která využívá jak části strukturní informace tak dat uložených v dokumentu. Nové rysy jsou vytvářeny jako kombinace vlastních dat a základní strukturní informace: atribut–hodnota, element–atribut, element–termín, nebo využívají pouze strukturu: element–element (rodič–dítě), element–element–element (levé dítě–rodič–pravé dítě). Problém různého pojmenování elementů v dokumentech z různých zdrojů byl řešen mapováním ontologií. Metodu úspěšně použili na klasifikaci dat z IMDb. Na tuto metodu navazujeme při návrhu nové metody prezentované v tomto článku.

3 Klasifikace dat v XML

Klasifikace je jednou z nejčastěji používaných úloh dolování znalostí. Uplatňuje se totiž v mnoha doménách, např. při filtrování spamů (klasifikace dokumentů). Většinu stávajících systémů strojového učení však nelze použít přímo na data v XML. Mohli bychom sice použít některý systém pro relační dolování [2], ty ale vyžadují definici doménové znalosti a jejich použití je tak dost náročné. Nejjednoduší by bylo informaci o struktuře úplně pominout a klasifikovat data jako prostý text. Tím se však připravíme o informace, které mohou pomoci při analýze dat. Vhodnější je proto transformovat data na jedinou tabulku takovou metodou, která zachová alespoň část strukturní informace. Následující text věnujeme popisu metody pro klasifikaci XML dokumentů založenou na

tomto postupu. Dokumenty jsou tedy nejprve převedeny do jediné tabulky (viz 3.1), ze které jsou poté vybrány pouze významné rysy (viz 3.2). Navržený systém jsme implementovali jako skript v jazyce Python a nyní je dostupný na `http://www.fi.muni.cz/~xproch11/xml`.

3.1 Transformace dat

Při návrhu metody jsme vycházeli zejména z prací M. Theobalda a kol. [8] a A. Termiera a kol. [7]. Hlavní rozdíl proti Theobaldovu přístupu spočívá ve způsobu vytváření rysů nesoucích informaci o struktuře dokumentu. Zavedli jsme nové rysy, které jsou obecnější a jsou proto schopny lépe zpracovat různě strukturované dokumenty. Jako příklad můžeme uvést rys typu předek–potomek, který je schopen popsat dokumenty, ve kterých jsou stejné elementy zapsány jednou jako řetězec a podruhé ve formě stromu.

Vytvářené rysy můžeme rozdělit do tří skupin podle typu nesené informace: *strukturní*, *strukturně-datové* a *datové* (termíny).

Strukturní rysy

element–element: daný element se vyskytuje v dokumentu (např. `movie@movi` nebo `genre@genre`)
element–element (rodič–dítě): zachycuje dva uzly spojene hranou (např. `movie@genres`, `genres@genre` nebo `roles@role`
element–element (předek–potomek): všechny tranzitivní relace od kořene směrem k listům (k rysům z předchozího příkladu by přibylo `movie@genre` nebo `movie@role`).

Strukturně-datové rysy

element–atribut: kombinace jména elementu a jména atributu, který je v něm obsažen (např. z `<role actor="Bruce Willis">` bude výsledkem `role@actor`)
element–atribut–hodnota: ke jménu elementu a atributu přibývá hodnota atributu (např. `role@actor@Bruce Willis`)
atribut–hodnota: spojení jména atributu a jeho hodnoty (`actor@Bruce Willis`)
element–hodnota: se jménem elementu je spojena hodnota atributu, který je v něm obsažen (např. `role@Bruce Willis`)
element–termín: spojení jména elementu s termínem, kde termíny jsou řetězce oddělené neviditelnými znaky z oblasti uzavřené elementem (z `<role>Man in coon skin hat</role>` dostaneme `role@Man`, `role@in`, `role@coon`, `role@skin`, `role@hat`).

Datové rysy

termín: jedná se o zobecnění rysu typu element–termín, kde termín není vázán na nadřazený element (dostaneme např. rysy `@Man`, `@in`, tedy vlastní data).

3.2 Výběr rysů

Počet vytvořených rysů je většinou příliš vysoký a jen malá část z nich je přitom vhodná k sestavení klasifikátoru. Ze sta dokumentů lze zkonstruovat i několik tisíc rysů, nicméně jejich velká část pokrývá velmi malé množství dokumentů.

Abychom urychlili fázi učení, používáme v navržené metodě pro výběr významných rysů filtrování [4]. Jako metriku jsme použili *informační zisk* (*information gain*, IG), která je založena na principu minimalizace entropie. Míra se ukázala jako velmi vhodná pro různé typy dat [3]. Hodnota této metriky tedy byla použita pro uspořádání všech vytvořených rysů, ze kterých se poté vybíralo prvních N, kde N zadává uživatel.

4 Experimenty

Metodu jsme testovali na datech z IMDb, ze kterých byly vytvořeny nové XML dokumenty. Data jsme získali od autorů práce [8]. Tato databáze obsahuje název, rok výroby, žánry, seznam herců a řadu dalších informací o filmech. Data přitom nemusejí být uplná. U mnoha filmů z počátku dvacátého století, především pak u němých snímků, není k dispozici seznam rolí a herců, kteří je ztvárňují. Naproti tomu u nových titulů máme k dispozici nejen úplný popis rolí, ale také realizační štáb či krátký popis zápletky a odkazy na příbuzná díla.

Úloha spočívala v klasifikaci filmů do dvou tříd. Třídy byly definovány na základě žánrů a tato informace byla samozřejmě z dat odstraněna. Zavedli jsme třídy komedie, drama, western a akční film. Z každé třídy bylo náhodně vybráno 50 filmů, které jsou reprezentovány samostatnými XML dokumenty.

Testovali jsme čtyři různé metody transformace dat rozdělené podle použitého typu rysů: CDATA (datové rysy), STRUCT (strukturní rysy), STRUCT + CDATA (strukturní a datové rysy) a ALL (všechny typy rysů). Vybírali jsme vždy prvních 1 až 5, 10, 20, 40, 60, 80, 100, 200, 300, 400, 500 a 1 000 nejlepších rysů. Jako hodnoty atributů byly ve výsledné tabulce použity hodnoty IG konkrétních rysů. Pro klasifikaci jsme použili algoritmy Naïve Bayes, IBk, SVM a J4.8 ze systému WEKA[2] s implicitním nastavením. Učení a testování bylo prováděno desetinásobnou křížovou validací s 90 trénovacími a 10 testovacími příklady.

5 Výsledky

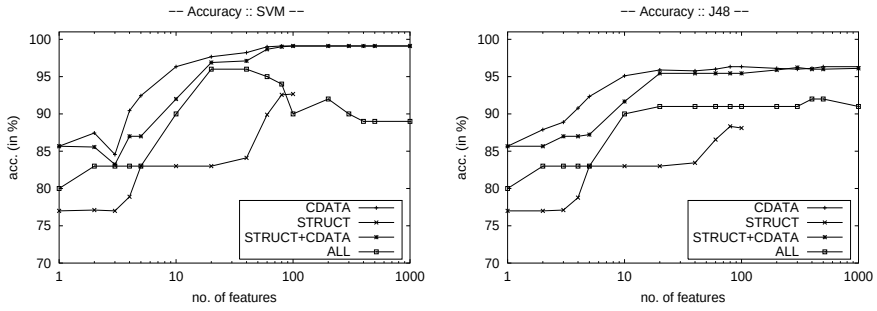
Westerny proti akčním filmům. Klasifikace akčních filmů proti westernům byla řešená také v práci M. Theobalda a kol. Použili jsme proto pro naše experimenty stejné nastavení (to znamená stejný počet učících a testovacích příkladů), abychom mohli porovnat dosažené výsledky.

[2] http://www.cs.waikato.ac.nz/ml/weka/

Při transformaci dat bylo vytvořeno 4 795 (CDATA), 105 (STRUCT), 4 900 (STRUCT+CDATA) a 15 170 (ALL) rysů. Pro názornost uvádíme zástupce jednotlivých typů rysů. Jedná se o vytvořené rysy, které získaly vysoké hodnoty informačního zisku.

```
movie@soundmix          (rodič-potomek)
colourinfo@and          (element-termím -- black and white)
soundmix@soundmix       (element-element)
plot@author@maurice     (element-atribut-hodnota)
plot@maurice            (element-hodnota)
@silent                 (termín)
```

Závislost přesnosti klasifikace učičů SVM a J4.8 na typu a počtu použitých rysů zobrazuje levý graf [3] na obr. 2. Z grafů je patrné, že klasifikace na základě vlastních dat (CDATA) je lepší, nebo stejně dobrá, jako při použití nových rysů. Graf na obr. 2 ukazuje vývoj hodnoty F_1 míry. S referenční metodou [8] jsme

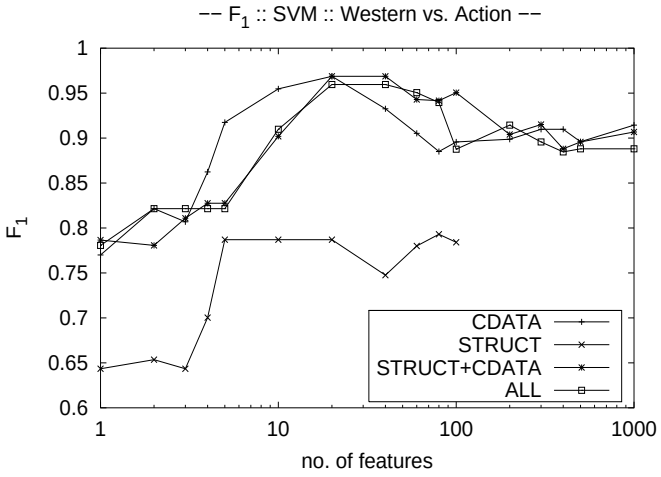


Obr. 1. Závislost přesnosti klasifikace na typu a počtu použitých rysů.

srovnávali právě přes touto metrikou vypočtené hodnoty pro algoritmus SVM. Autoři práce ukázali, že jejich metoda dosahuje lepších výsledků než klasifikace pomocí pouhých dat. Maximální hodnota F_1 míry se přitom pohybuje kolem 0.85 pro 100 a více rysů. V našich experimentech jsme dosáhli celkově vyšší přesnosti a to i při menším počtu rysů. Například již pro 10 rysů jsme naměřili 0.91 a maxima (0.96) jsme dosáhli při použití pouhých 20 rysů. Závislost přesnosti klasifikace zbylých dvou učičů (IBk a NB) na počtu rysů je podobná jako u SVM, nedosahují však tak dobrých výsledků.

Musíme tu však poznamenat, že nejlepší výsledky byly dosaženy pro CDATA (viz obr. 1), což je způsobeno výskytem silných datových rysů, které velmi dobře rozdělují data. Vyšší přesnost oproti referenční metodě pak byla s největší pravděpodobností dosažena díky použití míry IG namísto MI-score pro výběr rysů. Tato zjištění však ještě neznamenají, že se strukturní rysy při klasifikaci nemohou uplatnit.

[3] Všechny grafy jsou zobrazeny pro lepší přehlednost s použitím logaritmické škály na ose x.



Obr. 2. Závislost přesnosti klasifikace a hodnoty míry F_1 na typu a počtu použitých rysů. Jako pozitivní třída byly zvoleny westerny stejně jako v [8].

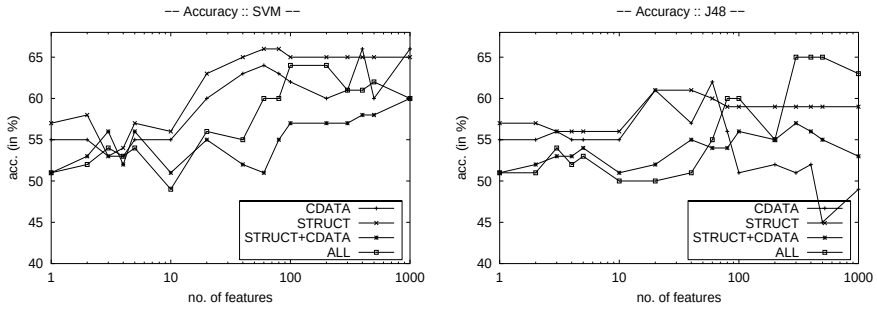
Komedie proti dramatu. Abychom skutečně ověřili kvalitu námi navržené metody, vybrali jsme z databáze IMDb data, která není možné přesně klasifikovat pomocí jednoduchých rysů. Vybrali jsme proto filmy označené jako komedie a drama, na kterých klasifikátor SVM dosáhl nejvíce 66% přesnosti (při výběru 400 a 1 000 rysů). Při výběrů příkladů jsme pak volili takové filmy, u kterých bylo uvedeno co nejvíce možných informací. Opět jsme pak testovali všechny čtyři metody tvorby rysů.

Přes špatné výsledky v přesnosti na této úloze, je z grafů na obrázku 3 vidět, že strukturní rysy mají pozitivní vliv na přesnost klasifikace složitějších dat. Byly zkonstruovány rysy představující zajímavé znalosti. Komedie měly často víceslovné názvy a tím pádem byly mezi významnými rysy `title@and`, `title@a`, atp.

Strukturní rysy vedly ke zvýšení přesnosti již při malém počtu atributů. Hodnoty 66 % bylo systémem SVM dosaženo již při klasifikaci dat obsahujících pouhých 60 rysů. Při klasifikaci dat systémem J4.8 se při tomto počtu rysů zase nejlépe osvědčila metoda CDATA, přestože nejlepší dosažený výsledek tohoto učiče je s metodou ALL při 300 až 500 rysech. Systémy IBk a NB dosáhly shodně nejvyšší přesnosti při metodě ALL a 200 rysech, přičemž jsou celkově až o 5 % horší než SVM nebo J4.8.

5.1 Diskuze

Důvodem proč strukturní a strukturně-datové rysy nezvyšují vždy přesnost, může být jejich velká specifičnost. Tím vysvětlujeme celkově menší přesnost metody ALL, kdy byly zkonstruovány rysy, které vedly k přeučení. Například rys



Obr. 3. Závislost přesnosti klasifikace dramatu vůči komediím na typu a počtu použitých rysů.

casting@position@3 (element–atribut–hodnota) může mít vysoké IG na trénovacích datech, ale není obecný a povede ke snížení přesnosti na testovacích datech.

Pokud bychom tedy nahlíželi na XML dokument jako na množinu stejně významných termínů, zjistíme, že obecně je přesnost v klasifikaci při malém počtu rysů vyšší u strukturu-využívajících metod, ale při vyšším počtu rysů je rozdíl zanedbatelný. Samotné termíny ale mohou být u složitějších a objemnějších dat příliš obecné a struktura může plnit funkci váhování, kdy element ve kterém se termín nachází, představuje kontext (např. `colourinfo@black` vůči `plot@black`). Jak je vidět z grafů na obrázku 3, mají strukturní rysy pozitivní vliv na přesnost při klasifikaci složitějších dat. Při klasifikaci westernů a akčních filmů vítězí vlastní data nad strukturou dokumentů. Je to způsobeno tím, že jsou informace u obou typů filmů velmi odlišné. Pro klasifikaci se tak nejlépe uplatnily rysy identifikující němý nebo černobílý film. Naproti tomu se při klasifikaci komedií vůči dramatům prosadí rysy založené na struktuře, protože se v samotných datech nevyskytují termíny, které by jednoznačně určovaly výslednou třídu.

Zjistili jsme, že používat více než tisíc rysů nemá význam, protože v nich už nenajdeme takové, co by obsahovaly informaci, která pomůže zvýšit přesnost, ale naopak zkomplikují proces učení.

6 Závěr

Navrhli a implementovali jsme metodu pro klasifikaci XML dokumentů bez použití schématu. Provedli jsme experimenty na XML dokumentech vytvořených z IMDb a uvedli srovnání s referenční metodou.

Dva prezentované experimenty ukazují protikladné typy problémů, se kterými se můžeme setkat. V prvním experimentu, který jsme převzali z prezentace referenční metody, nezvýšily strukturní rysy přesnost klasifikace. Zjistili jsme, že informačně silné rysy, které popisovaly parametry média (barva, zvuk, atp.), a které se dají získat reprezentací samotného dokumentu jednoduchým vektorem, vedou k lepší přesnosti. Druhý experiment (komedie proti dramatu) ukazuje,

že užití struktury má význam až u složitějších dat, kdy samotné termíny jsou příliš obecné a jejich váhovaní (např. spojení jména elementu a termínu) vede ke zvýšení přesnosti klasifikace.

7 Poděkování

Náš největší dík patří autorům článku [8] (M. Theobald a kol.), kteří nám laskavě k experimentování poskytli XML verzi databáze IMDb. Rádi bychom také poděkovali L. Popelínskému a anonymním recenzentům za cenné rady a připomínky. Tato práce je částečně podporována z grantů MŠMT 143300003 a MSM 0021622418.

References

1. Agrawal R. and Srikant R. Fast algorithms for mining association rules in large databases. In Bocca J. B., Jarke M., and Zaniolo C., editors, *VLDB'94, Proc. of 20th Intl. Conf. on Very Large Data Bases, September 12–15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
2. Džeroski S. and Lavrač N., editors. *Relational Data Mining*. Springer-Verlag, Berlin, September 2001.
3. Forman G. Choose your words carefully: An empirical study of feature selection metrics for text classification. In *Proc. of the 6th European Conf. on Principles of Data Mining and Knowledge Discovery*, pages 150–162. Springer-Verlag, 2002.
4. John G. H., Kohavi R., and Pfleger K. Irrelevant features and the subset selection problem. In *Intl. Conf. on Machine Learning*, pages 121–129, 1994.
5. Kramer S., Lavrač N., and Flach P. Propositionalization approaches to relational data mining. In Džeroski S. and Lavrač N., editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, September 2001.
6. Li W., Han J., and Pei J. CMAR: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, pages 369–376, 2001.
7. Termier A., Rousset M.-C., and Sebag M. TreeFinder: a first step towards XML data mining. In *Proc. of the 2000 IEEE Intl. Conf. on Data Mining*, 2002.
8. Theobald M., Schenkel R., and Weikum G. Exploiting structure, annotation, and ontological knowledge for automatic classification of XML data. In *Intl. Workshop on the Web and Databases*, 2003.
9. Worl Wide Web Consortium. *Extensible Markup Language (XML) 1.0*, W3C Recommendation edition, 2000.
10. Zaki M. J. and Aggarwal C. C. Xrules: Effective structural classifier for XML data. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, 2001.

Annotation:
XML documents classification
We present a new method for classifying XML documents that does not require any additional information like a XML scheme or DTD. A new method for transforming XML data into one table is introduced. We present results of experiments with data from Internet Movie Database (IMDb). We show that our method overcomes the previous work in terms of accuracy and F_1 measure.

Use of Knowledge Discovery Techniques in Evaluation of Foreign Direct Investment Survey

Tomáš Sabol[1,2], Ján Hreňo[1], Peter Bednár[2], and Peter Butka[2]

[1]Department of Banking and Investment, Faculty of Economics,
Technical University of Košice, Boženy Němcovej 32, 040 01 Košice, Slovakia

[2]Department of Cybernetics and Artificial Intelligence,
Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9/B, 041 20 Košice, Slovakia

{Tomas.Sabol, Jan.Hreno, Peter.Bednar, Peter.Butka}@tuke.sk

Abstract. The paper is focused on the application of KDD techniques on the results of the survey carried out within the FP5 Project "ProductivityGap". The aim of this survey, carried out in five countries of Central and Eastern Europe (CEE), was to determine the extent to which multinational companies (MNC) and other foreign investors into CEE actually support the transfer of foreign technology and its implementation in their subsidiaries in CEE. The intention was to map the starting positions of MNC's subsidiaries and trace possible improvements in their mandates, productivity, technology and local networking. Based on their answers a unique database on foreign direct investment subsidiaries was created and used for the analysis. A so-called Apriori algorithm was used and tested in this application for identification of association rules based on this database. Our goal is to describe preprocessing steps, formulation of hypotheses, evaluation and interpretation techniques used in presented approach as well as comparison with classic statistical methods and suggestions for the further research.

1 Introduction

In general it is expected that economic activity of foreign investors in countries of CEE will accelerate technological development in the host economy. There are several ways how foreign direct investment (FDI) in less developed economies can do that – by means of transfer of foreign technology to the host economy (in the form of product and process technology), or in form of management practices, improved information on and access to foreign markets (provided by the foreign owner), but also thanks to intensified competition.
Research within the "ProductivityGap"[1] project is focused on identification of determinants of the intensity of actual technology transfer in the manufacturing sector of Central and Eastern Europe. We are investigating under what conditions (for example in terms of dividing responsibilities for business functions a between foreign owner

[1] The Project „EU Integration and the Prospects for Catch-Up Development in CEECs. The Determinants of the Productivity Gap (Productivity Gap)" is supported by European Commission, DG Research, within the FP5 Programme, Contract No. HPSE-CT-2001-00065

and a local subsidiary) the foreign investment enterprises (FIEs) in CEE are best equipped to (technological) development.
Research is based on a database which containing data on FDI subsidiaries in selected new EU member states in CEE. These data were collected within the ProductivityGap survey carried out in 2002. We received 458 filled in questionnaires from 5 CEE countries. Based on their answers a unique database on foreign direct investment subsidiaries was created and used for the analysis.
In the next chapters of this paper use of KDD approach especially association analyses will be presented as well as preprocessing steps, formulation of hypotheses, interpretation of results and comparison with classic statistical approach. Finally, some conclusions and suggestions for the further research will be presented in the end of the paper.

2 Knowledge Discovery Techniques

Knowledge discovery in databases (KDD) is a process of semi-automatic extraction of knowledge from databases. Extracted knowledge should be valid (in statistical interpretation), unknown (i.e. knowledge was not known before application of KDD) and potentially useful (for the given purpose). In the case of the ProductivityGap survey, a record is an answered questionnaire and attributes are questions or subquestions.
The process of KDD consists of several steps. It is worth mentioning that this process is usually iterative and interactive. Definition of individual steps of KDD depends on the level of detail. Key steps are as follows [1]:

1. Definition and analysis of the goal of the given task.
2. Acquisition of relevant data and their understanding.
3. Preprocessing of data – extraction of relevant data, data clearing, integration, and transformation of data to a representation suitable for the given goal.
4. Data mining (DM) – application of intelligent methods for identification of valid patterns in data. Task of DM can be defined in several ways:
    a) Descriptive DM (generalization and discrimination methods)
    b) Predictive DM (classification and prediction methods)
    c) Unsupervised DM (clustering and association analyses)
5. Evaluation of identified data and knowledge identification.
6. Application of the acquired knowledge and evaluation of its use.

2.1 Association Rules

Association rules [2] are used for example for analysis of shopping baskets. A record in this application typically consists of items bought in a transaction. The problem can be formally defined as follows [2]: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq Z$. We say that a transaction T contains X, a set of some items in Z, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X,Y \subset Z$, and $X \cap Y =$

$\varnothing$. The rule $X \Rightarrow Y$ holds in the transaction set D with *confidence* c if $c\%$ of transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has *support* s in the transaction set D if $s\%$ of transactions in D contains $X \cup Y$.

3 Association rules found in the ProductivityGap survey results

For identification of association rules the Apriori algorithm [2] implemented in Weka [3] has been used. In principle to would be possible to run the algorithm over the whole database (with answers to all questions), but then it would be very difficult to find among the huge set of generated association rules some "interesting" ones. For this reason, the problem was decomposed into smaller sub-problems and in most cases only associations between two questions in the questionnaire were investigated. The result of the data-mining algorithm was a set of identified association rules describing (probabilistic) dependencies among values of attributes (i.e. among answers to sub-questions). These associations can then serve for support of denial of formulated hypotheses. This approach was used for identification and analysis of associations among individual and 'group' business functions (operational, marketing, strategic), business functions and magnitude of change, etc.

3.1 Pre-processing

Also in our experiments some pre-processing was needed, e.g.:

- Aggregation of sub-questions: Business functions (question Q7) were aggregated into 'group business functions':
  a) **Q7_O** - Operational business function (BF) includes sub-questions: Q7b - Process engineering, Q7d - Supply and logistics, Q7e - Accounting and finance, Q7l - Operational management,
  b) **Q7_M –** Marketing BF includes: Q7g - Market research, Q7h - Distribution and sales, Q7i - After sales services, Q7j - Advertisement, Q7k – Marketing,
  c) **Q7_S –** Strategic BF includes: Q7a - Product development, Q7c - Determining the product price, Q7f - Investment finance, Q7m - Strategic management or planning.
- Aggregations of values of attributes – some values were aggregated with the aim to get larger sets of same values. Otherwise no (or very limited number of) association rules for given values could be identified. For example following aggregation have been made for question Q7:
  - Q7 = 0: "*Mainly or only your company"* (i.e. *local subsidiary*)
  - Q7 = 1: "*Mainly or only foreign owner"*

  This applies to all sub-questions of Q7 including Q7_O, Q7_M, Q7_S.
- Question 9 – Magnitude of change - Q9x, x=a,b,c,d,e (Volume of sales, Share of export, Productivity, Level of technology, Quality of produce; values: -2 considerable reduction, -1 reduction, 0 no change, 1 increase, 2 considerable increase)

3.2 Association rules among individual business functions (Question Q7)

Let us start with looking for associations among individual 13 BFs. The algorithm (Apriori) generates a list of identified association rules ordered by decreasing confidence. For illustration, first 5 association rules as generated by the Apriori algorithm are given in List 1 (where all BFs have value 0). For example the rule No.1 says:

IF *Q7_g=0 & Q7_k=0 (what is true in 269 cases)* THEN *Q7_j=0 in 267 out of these 269 cases, i.e. this rules holds with confidence 267/269 = 0.99*

Such a list, however, needs some further processing (post-processing) – many rules overlap; set of items contained in one rule can be a subset of a set of items contained in another rule etc. And what is the most important it is necessary to select "interesting" association rules.

1. Q7_g=0 Q7_k=0 269 ⇒ Q7_j=0 267 conf:(0.99)
2. Q7_g=0 Q7_h=0 Q7_k=0 264 ⇒ Q7_j=0 262 conf:(0.99)
3. Q7_g=0 Q7_i=0 Q7_k=0 263 ⇒ Q7_j=0 261 conf:(0.99)
4. Q7_e=0 Q7_g=0 Q7_k=0 261 ⇒ Q7_j=0 259 conf:(0.99)
5. Q7_g=0 Q7_h=0 Q7_i=0 Q7_k=0 260 ⇒ Q7_j=0 258 conf:(0.99)

List 1 Association rules among individual BFs (where 0 – "mainly or only local subsidiary is responsible for the given BF") generated by the Apriori algorithm.

Of course, such association rules can be represented in different formats, e.g. rules from List1 are represented in Table1. The association rules (AR) have the form $X \Rightarrow Y$, where X and Y consist of one or more items (answers to questions). Items (consequents) - members of the set Y (i.e. the right hand side of the association rule) are in tables below in bold with sign '⇒' before the given item. For each association rule figures like support and confidence are given in the table.

Table 1 Association rules from List 1 (O – operational, M – marketing, S – strategic business function).

S	O	S	O	O	S	M	M	M	M	M	O	S	SuppX (No.)	SuppX (%)	Conf.	SuppXY (No.)
7a	7b	7c	7d	7e	7f	7g	7h	7i	7j	7k	7l	7m				
						0			**⇒0**	0			269	61%	0,99	267
						0	0		**⇒0**	0			264	60%	0,99	262
						0		0	**⇒0**	0			263	60%	0,99	261
				0		0			**⇒0**	0			261	59%	0,99	259
						0	0	0	**⇒0**	0			260	59%	0,99	258

Columns of Table 1, corresponding to individual BFs belonging to the same group of BF, have the same background. It can be seen from this table that most of the associations are among marketing BFs. Before going into a more detailed analysis let us formulate a hypothesis.

Hypothesis 1: *Associations among BFs belonging to the same group (i.e. either to operational or marketing or strategic) are higher than associations among BFs belonging to different groups of BFs.*

Based on analysis of a larger set of association rules (association rules with confidence higher than 0.90 were analysed) the more tables like Table2 were produced, also for other BFs, types of rules (1⇒1, 0⇒0, etc.) and their combinations.

Table 2 Associations within the group of Marketing business functions (Q7g, Q7h, Q7i, Q7j, Q7k) of the type "0 => 0" (where 0 - "Mainly or only local subsidiary"). For example the first AR in table should read (Q7g = 0) & (Q7k = 0) ⇒ (Q7j = 0) with confidence 0.99.

0 ⇒ 0	SuppX (No)	SuppX (%)	Conf.	SuppXY (No)
g, k ⇒ j	269	60%	0,99	267
g, h, k ⇒ j	264	59%	0,99	262
g, i, k ⇒ j	263	58%	0,99	261
g, h, i, k ⇒ j	260	58%	0,99	258
g, i, k ⇒ h	263	58%	0,99	260

In all cases, excluding BF Q7m (Strategic management or planning), the number of answers with the value 0 is higher than the number of answers with the value 1. It means that in majority of cases responsibility is on the side of local subsidiary. However, the ratio between the number of 0's and 1's decreases in the order Operational – Marketing - Strategic BFs. Still, there are some exceptions (irregularities) in "linearity" of this decrease.

Some significant concluding comments based on analyses:

- There are strong associations among Marketing and Operational BFs
- Associations among Strategic BFs (conf. higher than 0.9) have not been found.
- Associations with confidence higher than 0.9 have been found also between Marketing and Strategic BFs.
- No associations between different values of BFs (i.e. between "Mainly or only local subsidiary" and "Mainly or only foreign owner") of that high confidence have been found.

Once the associations were identified, we used also statistical approach and calculated Spearman's correlation coefficients between individual business functions. These figures are in accordance with the association rules given above. Concluding this section, we cannot say that the Hypothesis 1 has been fully supported. Although associations between BFs belonging to the same groups are strong, confidence of associations between some BFs belonging to different groups is also high.

3.3 Association rules among groups of business functions

As mentioned above, the 13 business functions can be grouped into 'group' business functions Operational, Marketing and Strategic BFs as described above. Let us define an ordering of "group" BFs by defining a "level" of BF in the following way - Operational BF - level 0, Marketing - level 1 and Strategic - level 2.

Hypothesis 2: *Responsibility for business functions is delegated by foreign owner (FO) to local subsidiary (LS) in an ascending order, i.e. before LS overtakes responsibility for the level i, LS should have responsibility for level (i-1). It means, for ex-*

ample that LS should be first responsible for Operational BFs and only then can overtake responsibility for Marketing BFs.

If summarizing table for 'group' business functions were produced, it can be seen that the number of answers for the value 0 (i.e. "mainly or only local subsidiary") is decreasing with the level of BF (i.e. less FIEs are responsible for Marketing BFs than for Operational, and even less for Strategic BF) and, logically, the number of answers for the value 1 ("Mainly or only foreign owner") has an opposite, decreasing trend. To testify the Hypothesis 2 it is necessary to look at association rules among the values of BFs.

An association rule will be called "backward" if has the form "IF *higher level BF = value* THEN *lower level BF = value*" and an association rule will be called "forward" if it says "IF *lower level BF = value* THEN *higher level BF = value*".

In our case we need to analyse backward and forward association rules for different type of rules ($0\Rightarrow0$, $1\Rightarrow1$, $0\Rightarrow1$, $1\Rightarrow0$) between all groups of business functions.

After analyses we found that:

- If local subsidiary is responsible for higher level BF that with very high probability (higher than 0.9) is responsible also for lower level BF. For BFs differing by two levels (i.e. association S⇒O) confidence is higher than for BFs differing only by one level.
- Confidence of backward rules of $1 \Rightarrow 0$ type is on average lower than for backward rules of $0 \Rightarrow 0$ type.

Thus, in principle we can say that we have not found any association rules, which would be in contradiction with the Hypothesis 2. Trying to support the hypothesis we mostly used "atomic" association rules ($X \Rightarrow Y$, where both X and Y contain only one item, it is a unidirectional relation). Confidence of atomic AR corresponds to conditional probability.

For comparison with the associations defined in the form of atomic association rules, Spearman's correlation coefficients between group BFs were calculated ($\alpha = 0.05$). Correlation coefficients are by definition symmetrical. On the other hand, dependencies identified by the association rules show asymmetrical nature of these relations. At the same time, the association rules show dependencies between individual values of investigated variables (in this case BFs), it means they provide more detailed information.

3.4 Associations between business functions (Question Q7) and volume of sales, share of export, level of productivity, technology and quality (Question Q9)

Hypothesis 3: *The higher the responsibility of foreign owner in business functions, the higher the increase in productivity, sales, share of export, level of technology and quality of produce.*

In this case the Apriori algorithm was applied to 'group' business functions Q7_O, Q7_M, Q7_S and sub-questions of Question 9. The following procedure was used: the DM algorithm was run on one sub-question of Q9 and Q7_O, Q7_M, Q7_S. Again, same type of tables as in previous chapters were produced.

It is noteworthy that absolute majority of values of Q7_O, Q7_M, Q7_S in generated rules (with high confidence) is 0 (i.e. mainly or only local subsidiary). That can be (at least partly) explained by the fact that value 0 is more frequent than 1.

For example result contains atomic AR: (Q9a = 2) ⇒ (Q7_O = 0), which hold with confidence = 0.88. It means that 88% of FIEs, which achieved considerable increase in volume of sales, are responsible for operational BFs.

ARs in opposite direction, i.e. for example (Q7_O = 0) ⇒ (Q9a = 2), have much lower confidence (0.49).

Based on the analysis we can formulate the following conclusions:

1. Majority of the association rules found are for Q7_O, Q7_M, Q7_S equal to 0 (i.e. local subsidiary is mainly or only responsible for the given BF).
2. For ARss *(Q9x = y) ⇒ (Q7_z = 0),* where x = a, b, c, d, e; y = 0, 1, 2; and z = O, M, S holds: the highest the level of group BF, the lowest the confidence.
3. Out of those FIEs which enjoyed considerable increase, only in those FIEs which enjoyed considerable increase in volume of sales is an average responsibility of local subsidiary for strategic BFs higher than average responsibility of foreign owner.

For comparison an average "autonomy index" (AI) is used, this index is in fact the number of companies with Q7 = 0 (i.e. LS is mainly or only responsible for the given BF) divided by the number of all FIEs answering this question (AI = 0 means that LS has no responsibility in the area, AI = 1 means that LS has all responsibility for the given BF). AI/Q9a = 2 is a 'conditional' AI - autonomy index of those FIEs which experienced considerable increase (i.e. Q9a = 2). Symbol Δ denotes a relative difference between the autonomy index and conditional autonomy index (in %).

Table 3. Comparison of the autonomy index of all FIEs and of FIEs with (Q9a = 2), (Q9a = 1), and (Q9a = 0), respectively AI – autonomy index, AI/Q9a=x – autonomy index for FIEs with Q9a = x, Δ - relative difference between AI/Q9a=x and AI (for x = 2, 1, 0).

	AI	**AI/Q9a=2**	**Δ**	**AI/Q9a=1**	**Δ**	**AI/Q9a=0**	**Δ**
Q7_O	0.86	0.88	2%	0.83	-3%	0.88	2%
Q7_M	0.72	0.70	-3%	0.79	9%	0.57	-21%
Q7_S	0.52	0.54	4%	0.52	-1%	0.47	-10%

It is interesting to note that FIEs, which experienced considerable increase in volume of sales actually as far as autonomy index is concerned do not differ from "average" FIEs (autonomy index calculated from all FIEs participating in the survey). On the hand, autonomy index in Marketing BFs of FIEs with unchanged volume of sales is by 21% below average. Similar tables for Q9b, Q9c, Q9d, and Q9e can be created.

Our results indicates that responsibility for BFs is a factor contributing to considerable increase in some areas defined in the Question 9 (Q9) of the questionnaire (Hypothesis 3). Differences in the average autonomy index are usually the largest for Strategic BFs and the lowest for Operational BFs.

Once these differences in business function autonomy were identified, we tested whether differences between sets of FIEs with Q9x = 0, Q9x = 1, and Q9x = 2. We tested differences between all three groups (using Kruskall-Wallis and Jonckheere-

Terpstra tests) as well as differences between pairs of sets (Mann-Whitney and Kolmogorov-Smirnov tests). Significant differences at the level of significance $\alpha = 0.05$ were identified. We found that confidence of association rules or respectively autonomy index AI/Q9x = k. Of course, it cannot replace statistical testing of a zero hypothesis (that the sets are not different) at the given level of significance.

4 Conclusions

A complementary approach to "classical" statistical approach using the data-mining algorithm was tested in this paper. We found that the achieved DM results correspond to the results of classical statistics. Further research can be focussed also on several other issues:

- Based on the results of the identified "interesting" bilateral associations to test also "multilateral" associations (for sub-questions from "interesting" ARs)
- To design and implement an algorithm for (semi-)automatic post-processing of data mining results for this kind of applications (results of questionnaire survey).
- To test a data mining software enabling some kind of specification of association rules we are looking for (e.g. LISPMiner software [4]) or to design a language for specification of such rules
- To test also other data mining algorithms on this set of data (e.g. cluster analysis)

Still one may ask what are advantages or value added of this approach. Potential advantages result from differences of association rules to correlation coefficients:

- Correlation coefficients define dependency between variables, association rules define dependencies among concrete values of variables
- Correlation coefficients are symmetrical, association rules are asymmetrical
- Association rules in general associate two or more variables.

Traditional approach is deductive – researcher formulates a hypothesis and then it is tested whether collected data are in harmony with the formulated hypothesis or not. Data mining techniques can be also used (as "another methodology") within this approach. However, data mining techniques can support also inductive approach – one would specify required minimum confidence and the data mining tool would generate all the association rules with confidence higher than the specify minimum confidence. And then, based on all the generated association rules, some conclusions could be formulated.

References

[1] Paralič, J.: Objavovanie znalostí v databázach. Elfa, Košice, ISBN 80-90066-60, 2002
[2] Agrrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Proc. Of 20th International Conference Very Large Data Bases, VLDB, 12-15, Morgan Kaufmann, ISBN 1-55860-153-8, 1994
[3] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco, ISBN 1-55860-552-5, 2000
[4] Rauch, J., Šimunek, M.: Systém LISp-Miner. In: Svátek, V.(ed.). Znalosti 2003. Ostrava: TU Ostrava, pp. 83–92, ISBN 80-248-0229-5, 2003

Refined Extension Principle for Multidimensional Dynamic Logic Programs

Jozef Šiška

Institute of Informatics, Faculty of Mathematics, Physics and Informatics,
Comenius University, Bratislava, Slovakia
siska@ii.fmph.uniba.sk

Abstract. Dynamic Logic Programming (DLP) and Multidimensional Dynamic Logic Programming (MDLP) represent a way to express changing knowledge using the language of logic programming and the idea of program updates. Many semantics were introduced for such programs, sharing common problems when handling updates with programs that intuitively do no carry any new knowledge, for example tautological updates. A *refined exension principle* for DLP was proposed by J.J.Alferes et. al. to formalize a certain type of updates, that should not change the stable model of a program, along with a new refined semantics. In this paper we extend the principle to the case of MDLP and identify the problems caused by nonlinearity that prevent a straighforward lifting of the semantics from linear case.

1 Introduction

Logic programming is one of the mostly used formalisms in knowledge representation and because of its foundations in classical logic it allows easy representation of static knowledge. Dynamic Logic Programming (DLP) and Multidimensional Dynamic Logic Programming (MDLP) represent a way to express changing knowledge using the same language and the idea of program updates.

Many semantics were defined for DLP. These semantics share different properties and it is hard to compare them or decide which are better. In [2] a refined extension principle was defined, which concerns the addition of rules that should not change the meaning of a program. Our contribution in this paper is the introduction of this principle into the MDLP – a generalization of DLP that allows for more freedom when expressing relations between logic programs and identification of the problems that prevent a straightforward lifting of the semantics from linear case.

2 Motivation

A number of semantics for Dynamic Logic Programming have been proposed. Many of these are based on rejection of rules and all suffer from tautological

* This work was partially supported by grant VEGA 1/0173/03

updates. By a tautological update we understand either update with a tautology or a rule belonging to a tautological cycle.

The danger of tautological rules lies in the fact that being satisfied by some model, they can reject rules with opposite literals in head. However being unsupported, a tautology does not actually say anything about its head. This is not a problem for most semantics. But when the rejected rule was in conflict with some other rule (not the tautology), thus generating an inconsistency, a new stable model can be achieved, because of the inconsistency being removed.

As this certainly presents a problem, it does not end with tautologies. Even rules which are a part of a cycle that is not supported from outside show similar behaviour. Although a simple solution would be to allow only acyclic programs besides the fact that detecting such cycles is not easy, it denies an easy way to restrict certain models as shown in [2].

3 Preliminaries

In this paper we will concentrate on a class of logic programs called *generalized logic programs* [13] which allow for default negation in heads and bodies of rules. Using the notion of updates of logic programs we define the class of Multidimensional Dynamic Logic Programs [8].

Let $\mathcal{A}$ be a set of propositional atoms. For an atom A, not A is called a *default literal*. We denote $\mathcal{L} = \{A, \text{not } A | A \in \mathcal{A}\}$ the set of all literals. For a literal $L \in \mathcal{L}$, we use not L to denote its *opposite* counterpart not A if $L = A$ or A if $L = \text{not } A$. We call L and not L *conflicting* literals and denote by $L \bowtie \text{not } L$. For a set of literals $M \subseteq \mathcal{L}$ we denote M^+ the set of all objective literals and M^- the set of all default literals from M. A set of literals M is called *consistent* if it does not contain two conflicting literals, otherwise it is *inconsistent*.

A *generalized logic program* P is a countable set of *rules* of the form $L \leftarrow L_1, L_2, \ldots, L_n$, where L and each of L_i are literals. When all literals are objective, P is called a *definite logic program*.

For a rule r of the form $L \leftarrow L_1, L_2, \ldots, L_n$ we call L the *head* of r and denote by $\text{head}(r)$. Similarly by $\text{body}(r)$ we denote the set $\{L_1, L_2, \ldots, L_n\}$ and call it the *body* of r. If $\text{body}(r) = \emptyset$ then r is called a *fact*. Two rules r and r' are called *conflicting* (opposite) if $\text{head}(r)$ and $\text{head}(r')$ are conflicting and we denote this by $r \bowtie r'$.

An *interpretation* is any consistent set of literals I. I is called a *total* interpretation if for each $A \in \mathcal{A}$ either $A \in I$ or not $A \in I$. A literal L is *satisfied* in an interpretation I ($I \models L$) if $L \in I$. A set of literals $S \subset \mathcal{L}$ is satisfied in I ($I \models S$) if each literal $L \in S$ is satisfied in I. A rule r is satisfied in I ($I \models r$) if $I \models \text{body}(r) \rightarrow I \models \text{head}(r)$. A total interpretation M is called a model of logic program P if for each rule $r \in P$ $M \models r$. We also say that M *models* P and write $M \models P$.

For a generalized logic program P we denote $\text{least}(P)$ the least model of the definite logic program P' obtained from P by changing each default literal not A into a new atom *not_A*. According to [3] we can define *stable models* of a

generalized logic program as the models M for which

$$M = \text{least}(P \cup M^{-}).$$

3.1 Multi-dimensional Dynamic Logic Programming

To express knowledge changing over a linear sequence of states, Dynamic Logic Programming[1] was introduced. It was further extended into Multidimensional Dynamic Logic Programming[9], a generalization with nonlinear orderings of states, which allows for more complex knowledge hierarchies, described by partial orderings or acyclic directed graphs (DAGs).

A *directed graph* is a pair $D = (V, E)$, $E \subseteq V \times V$. An edge $(u, v) \in E$ is considered to lead from u to v. A sequence of edges $(v_0, v_1), (v_1, v_2), \ldots (v_{n-2}, v_{n-1})$, (v_{n-1}, v_n) is called a *directed path* from v_0 to v_n. A *directed acyclic graph* (DAG) is a directed graph with no directed path from v to v for all vertices $v \in V$.

Every DAG $D = (V, E)$ defines partial orderings $<_D$ and $\leq_D$ on its set of vertices: $u <_D v$ if there exists a directed path from u to v and $\leq_D = <_D \cup \{(v, v) | v \in V\}$.

The *relevancy DAG* of D with respect to $v \in V$ is $D_v = (V_v, E_v)$ with:

$$V_v = \{u | u \in V, u \leq_D v\} \qquad E_v = E \cap (V_v \times V_v)$$

A *multidimensional dynamic logic program* (MDLP) $\mathcal{P}$ is a pair $(\mathcal{P}_D, D)$, where $D = (V, E)$ is an acyclic directed graph. Elements of V are called states and $\mathcal{P}_D = \{P_v | v \in V\}$ is a set of generalized logic programs corresponding to these states.

By $\rho(\mathcal{P})$ we denote the multiset of all rules in $P_v, v \in V$, and by $\mathcal{P} \cup \mathcal{P}'$ ($\mathcal{P}' = (\mathcal{P}'_D, D)$) the MDLP $(\{P_v \cup P'_v | v \in V\}, D)$.

Defined in this way MDLP allows for infinite DAGs. We will however restrict to DAGs for which the relevancy DAG for every vertex is finite. Semantics for MDLP is always defined at a certain state (or a set of states). We could consider semantics at the set of states $S = V$, having so a view of the whole program, but this may not be very intuitive in complex MDLPs.

When determining the semantics, rules from all relevant programs are taken into account and possible inconsistencies are resolved according to the principle that knowledge from more important program takes precedence that from less important programs.

Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a MDLP with $D = (V, E)$ and $P_D = \{P_v \mid v \in V\}$, $D_s = (V_s, E_s)$ is the relevancy graph of D with respect to $s \in V$. A total interpretation M_s is a stable model of P at state s iff

$$M_s = \text{least}\left([\rho(\mathcal{P})_s \setminus \text{Rej}(\mathcal{P}, s, M_s)] \cup \text{Def}(\rho(\mathcal{P})_s, M_s)\right)$$

where

$$\rho(\mathcal{P})_s = \bigcup_{v \in V_s} P_v$$

$$\text{Rej}(\mathcal{P}, s, M_s) = \{r \in P_i | \exists r' \in P_j : r \bowtie r', i <_D j \leq_D s, M_s \models \text{body}(r')\}$$

$$\text{Def}(\rho(\mathcal{P})_s, M_s) = \{\text{not } A | \nexists r \in \rho(\mathcal{P})_s : \text{head}(r) = A, M_s \models \text{body}(r)\}$$

The set $\text{Rej}(\mathcal{P}, s, M_s)$ contains all *rejected* rules from $\mathcal{P}$, i.e. rules for which there is a conflicting rule satisfied in M in more recent program. The set $\text{Def}(\rho(\mathcal{P})_s, M_s)$ contains default negations of all unsupported atoms – defaults.

The previous definition defines the *Dynamic Stable Model semantics* (DSM) for MDLP[8]. When requiring $<_D$ to be strict, finite ordering, we get an eqivalent definition of DLP [1, 8] which uses the notation $\rho(\mathcal{P})$, $\text{Rej}(\mathcal{P}, M)$, $\text{Def}(\rho(\mathcal{P}), M)$ for the respective sets. There are other semantics for DLP [5, 6, 10, 11] and their equivalents for MDLP [8, 7] based on causal rejection of rules. All of them can be achieved by corresponding definitions of the sets $\text{Rej}(\mathcal{P}, s, M_s)$ and $\text{Def}(\rho(\mathcal{P})_s, M_s)$.

A transformational semantics for MDLP that is equivalent to the DSM semantics is defined in [8]. Such semantics transforms the MDLP into a generalized logic program with a corresponding set of models. This allows for direct implementation of the DSM semantics for MDLP by using existing solutions for GLP[12, 4]. Ability to construct such transformational semantics will be an important aspect when looking for a refined semantics.

4 Refined Extensions

In order to distinguish between semantics that suffer from tautological updates and those that do not, refined extension principle [2] was defined. In this section we recall the notion of refined extension for DLP, which is a way of describing an addition of rules that do not change the meaning of logic program.

Using the notion of refined extension a *refined extension principle* was defined, which is used to distinguish between semantics that correctly deal with addition of tautological rules and those that do not. After that a short survey follows on how exactly the problem of tautologies arises and how it can be solved along with the definition of a complying semantics from [2].

We will now be interested in semantics for DLP based on causal rejection of rules[6, 8]. Models of such semantics can be characterized according to definition given in section 3.1, where we can get various semantics with various definitions of the sets Rej and Def.

First the *syntactic extension* is defined. Informally, $P \cup E$ is a syntactic extension of P if rules from E do not change the least Herbrand model of P.

Definition 1. *Let P and E be generalized logic programs. $P \cup E$ is a* **syntactic extension** *of P iff* $\text{least}(P) = \text{least}(P \cup E)$

Consider a semantics Sem. We can now use the notion of syntactic extension to verify whether addition of rules in $\mathcal{E}$ affects the model in Sem, because the model will be characterized by the least Herbrand model. When all models of $\mathcal{P} \cup \mathcal{E}$ are unaffected, we say that $\mathcal{P} \cup \mathcal{E}$ is a refined extension of $\mathcal{P}$.

Definition 2. *Let $\mathcal{P}$ and $\mathcal{E}$ be two dynamic logic programs, Sem a semantics and M an interpretation. We say that $\mathcal{P} \cup \mathcal{E}$ is an* **extension** *of $\mathcal{P}$ with respect to M iff*

$$[\rho(\mathcal{P}) - \text{Rej}(\mathcal{P} \cup \mathcal{E}, M)] \cup \text{Def}(\mathcal{P} \cup \mathcal{E}, M) \cup [\rho(\mathcal{E}) - \text{Rej}(\mathcal{P} \cup \mathcal{E}, M)]$$

is a syntactic extension of $[\rho(\mathcal{P}) - \mathrm{Rej}(\mathcal{P} \cup \mathcal{E}, M)] \cup \mathrm{Def}(\mathcal{P} \cup \mathcal{E}, M)$

We say that $\mathcal{P} \cup \mathcal{E}$ *is a* **refined extension** *of* $\mathcal{P}$ *iff* $\mathcal{P} \cup \mathcal{E}$ *is an extension of* $\mathcal{P}$ *with respect to all models in* $Sem(\mathcal{P} \cup \mathcal{E})$.

4.1 Refined Extension Principle and Refined Semantics

Using the definition of refined extension, J.J.Alferes et al. defined a *refined extension principle* [2].

Definition 3 (Refined extension principle for DLP). *A semantics Sem for dynamic logic programs complies with the refined extension principle iff for any dynamic logic programs* $\mathcal{P}$ *and* $\mathcal{P} \cup \mathcal{E}$*, if* $\mathcal{P} \cup \mathcal{E}$ *is a refined extension of* $\mathcal{P}$ *then*

$$Sem(\mathcal{P} \cup \mathcal{E}) \subseteq Sem(\mathcal{P})$$

Thus according to the principle, an addition of rules that do not change the meaning of the program (i.e. a refined extension) should not generate a new stable model. As shown in [2] the DSM semantics does not conform to this behaviour.

The problem arises, when there are two conflicting equally important rules with heads L and not L. Tautological rule r than can reject one of them, leaving the other to infer it's head and thus accepting a model that was not allowed before.

Example 1. Consider following DLPs

$$\mathcal{P} = \{P_1 = \{A \leftarrow;\ \text{not } A \leftarrow\}, P_2 = \{\}\}$$
$$\mathcal{E} = \{E_1 = \{\}, E_2 = \{A \leftarrow A\}\}$$

$\mathcal{P} \cup \mathcal{E}$ is a refined extension of $\mathcal{P}$. $\mathcal{P}$ has no stable models, however $\mathcal{P} \cup \mathcal{E}$ has a stable model $M = \{A\}$. This is because not $A \leftarrow$ is rejected by $A \leftarrow A$ and then

$$M = \mathrm{least}(\{A \leftarrow;\ A \leftarrow A\}).$$

Because none of the previous semantics for dynamic logic programs satisfied the refined extension principle, in [2] together with the principle also a semantics that complies with it is defined. In this semantics, both conflicting rules are removed.

Definition 4 (Refined stable model semantics). *(J.J.Alferes et al.) Let* $\mathcal{P}$ *be a dynamic logic program and* M *an interpretation.* M *is a* refined dynamic stable model *of* $\mathcal{P}$ *iff*

$$M = \mathrm{least}\left(\rho(\mathcal{P}) \setminus \mathrm{Rej}(\mathcal{P}, M) \cup \mathrm{Def}(\mathcal{P}, M)\right)$$

where $\mathrm{Rej}(\mathcal{P}, M) = \{r \mid r \in P_i, \exists r' \in P_j : i \leq j, r \bowtie r', M \models \mathrm{body}(r')\}$ *and* $\mathrm{Def}(\mathcal{P}, M)$ *is defined in the same way as in DSM semantics for DLP.*

The only actual change is in replacing $i < j$ with $i \leq j$. This allows conflicting rules in single program to reject each other. Instead of generating an inconsistency which denied that model, there is now no rule that could infer either an atom or its negation (only the tautology has such a head, but that is not able to infer it) and thus the model is not a stable model.

5 Refined Extension for MDLP

The idea of *Dynamic Logic Programs* was extended into *Multidimensional Dynamic Logic Programs* to allow better expressiveness in the area of preferences between programs. We now would like to define the refined extension and refined extension principle for this class of programs in a way that will not conflict with the linear case. After this we will investigate the reasons that prevent the refined semantics to be lifted from the linear case.

5.1 Refined Extension Principle for MDLP

The basic differences between semantics for MDLPs and for DLPs are only in the definition of Rej (which actually has only $<$ changed to $<_D$) and in the fact that a stable model is always considered at a certain state s, which can be either a vertex or a set of vertices of the digraph. We therefore change the definitions to reflect this.

The same MDLP can however behave as different programs when considering different states. Because of this we need to check the MDLP at all states when we want to distinguish a refined extension. This leads us to a definition of refined extension at a certain state, which is similar to the DLP case:

Definition 5. (Refined Extension at a state - MDLP) *Let $\mathcal{P}$ and $\mathcal{E}$ be two multidimensional dynamic logic programs over the same graph D,* Sem *a semantics for MDLP, M_s an interpretation and $s \in V_D$ a state. We say that $\mathcal{P} \cup \mathcal{E}$ is an* **extension** *of $\mathcal{P}$ with respect to M_s at state s iff*

$$[\rho(\mathcal{P})_s - \mathrm{Rej}(\mathcal{P} \cup \mathcal{E}, s, M_s)] \cup \mathrm{Def}(\rho(\mathcal{P} \cup \mathcal{E})_s, M_s) \cup [\rho(\mathcal{E})_s - \mathrm{Rej}(\mathcal{P} \cup \mathcal{E}, s, M_s)]$$

is a syntactic extension of

$$[\rho(\mathcal{P})_s - \mathrm{Rej}(\mathcal{P} \cup \mathcal{E}, s, M_s)] \cup \mathrm{Def}(\rho(\mathcal{P} \cup \mathcal{E})_s, M_s)$$

We say that $\mathcal{P} \cup \mathcal{E}$ is a **refined extension** *of $\mathcal{P}$ at state s iff $\mathcal{P} \cup \mathcal{E}$ is an extension of $\mathcal{P}$ with respect to all models in* $\mathrm{Sem}(\mathcal{P} \cup \mathcal{E})$.

Except the problem with the state s the previous definition is just a simple modification of the linear case. The Rej and Def sets for DLPs have been replaced by appropriate ones for multi-dimensional case.

Definition 6. (Refined Extension Principle - MDLP) *A semantics* Sem *for multi-dimensional dynamic logic programs complies with the refined extension principle iff for any two MDLPs $\mathcal{P}$ and $\mathcal{P} \cup \mathcal{E}$ over the same DAG D, if $\mathcal{P} \cup \mathcal{E}$ is a refined extension of P at all states then*

$$\mathrm{Sem}(\mathcal{P} \cup \mathcal{E}) \subset \mathrm{Sem}(\mathcal{P})$$

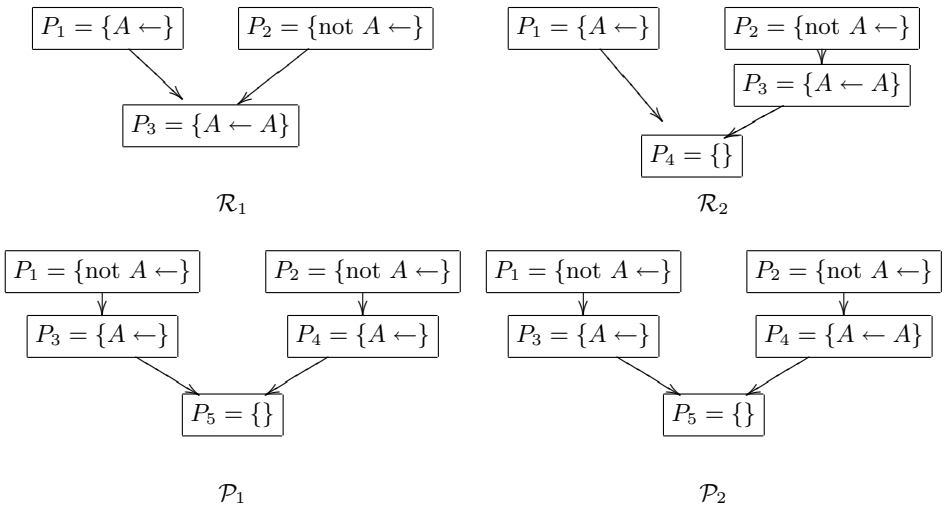
As may be expected, none of existing semantics complies with this principle and as we will see in the next section, the refined semantics for DLPs cannot be easily lifted to this case.

5.2 Tautologies in MDLP

In section 4.1 we have seen how the problem in linear case arises and how the refined dynamic stable model semantics handles it.

The basic problem in nonlinear case is the fact that we now have pairs of rules that are incomparable with respect to the ordering (preferences) of the programs – the DAG. In linear case all incomparable rules were in one program P_i, all other rules were either less important and rejected or more important.

Fig. 1. MDLPs with tautologies



In MDLP all incomparable rules from different branches of the graph are taken as equally important and thus can produce an inconsistency. Consider the MDLP $\mathcal{R}_1$ in figure 1. Rules in P_1 and P_2 are conflicting in model $M = \{A\}$ and none is more important. With $P_3 = \{\}$ M would not be a stable model because of inconsistency. But the tautology $A \leftarrow A$ rejects the rule in P_2 and so M is a stable model according to existing semantics. In $\mathcal{R}_2$ the situation is just slightly altered to show that the tautology does not have to be more important than all involved rules – it just needs to rejects all opposite rules.

The solution for DLP was to let incomparable (i.e. those in the same program) conflicting rules to reject themselves. Now it is even hard to identify the conflicting rules that should create the inconsistency. There are two MDLPs in figure 1. They differ in the rule in P_4. In $\mathcal{P}_1$ it is a fact and in $\mathcal{P}_2$ it is a tautology. Intuitively $\mathcal{P}_1$ should have a stable model $M_s = \{A\}$ at state 5 but $\mathcal{P}_2$ should have no models at that state. This is because the tautology in P_4 should not reject and allow rules in P_2 and P_3 to conflict and create the inconsistency. Clearly we cannot allow incomparable conflicting rules to reject themselves as there would be no rules left in $\mathcal{P}_s$. The decision whether to reject or not would

therefore be based on fact whether some rule is tautological or not. This is not easy do decide in general and although it could lead to a definition of complying semantics, it would deny an easy construction of a transformational semantics.

6 Conclusion

We extended the Refined Extension Principle into the class of Multidimensional Dynamic Logic Programs. After studying the construction of Refined Stable Model Semantics for DLP, we argued why such a construction could not be easily lifted to the nonlinear case. A way was proposed in which a complying semantics could be defined. This semantics would however require to identify tautological rules which would deny the possibility of easy definition of a transformation semantics which would lead to an implementation of the semantics using existing systems for logic programming.

References

1. J.J.Alferes, J.A.Leite, L.M.Pereira, H.Przymusinska, and T.C.Przymusinski. Dynamic logic programming. In Procs. of KR'98. Morgan Kaufmann, 1998.
2. J.J.Alferes, F.Banti, A.Brogi, and J.A.Leite. Semantics for Dynamic Logic Programming: a principled based approach. In Procs. of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7), Springer-Verlag, LNAI, 2004
3. J.J.Alferes, J.A.Leite, L.M.Pereira, H.Przymusinska, and T.C.Przymusinsky. Dynamic updates of non-monotonic knowledge bases. The Journal of Logic Programming, 45(1-3):43-70, September/October 2000
4. A disjunctive datalog system. `http://www.dbai.tuwien.ac.at/proj/dlv`.
5. T.Eiter, M.Fink,G.Sabbatini, and H.Tompits. On Updates of Logic Programs: Semantics and Properties. INFSYS Research Report 1843-00-08, 2001.
6. T.Eiter, M.Fink,G.Sabbatini, and H.Tompits. On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming, 2(6), 2002.
7. Martin Homola, On relating the Various Semantic Approaches in Multidimensional Dynamic Logic Programming, Master's thesis, Comenius University, Faculty of Mathematics, Physics and Informatics, Bratislava, June 2004
8. J.A.Leite. Evolving Knowledge Bases, volume 81 of Frontiers in Artifical Inteligence and Applications. IOS Press, 2003.
9. J.A.Leite, J.J.Alferes, and L.M.Pereira. Mutli-dimensional Dynamic Knowledge Representation. In Procs. of LPNMR'01, volume 2173 of LNAI, Springer, 2001
10. J.A.Leite, L.M.Pereira. Generalizing updates: From models to programs. In Procs. of LPKR'97, volume 1471 of LNAI. Springer Verlag, 1997.
11. J.A.Leite, L.M.Pereira. Iterated logic program updates. In Procs. of JICSLP'98. MIT Press, 1998.
12. The SMODELS system. `http://www.tcs.hut.fi/Software/smodels/`.
13. V.Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In Procs. of KR'92. Morgan-Kaufman, 1992.

Modifikace bayesovského disambiguátoru

Michal Toman, Karel Ježek

Katedra informatiky, FAV, Západočeská univerzita v Plzni,
Univerzitní 8, 306 14, Plzeň

{mtoman, jezek_ka}@kiv.zcu.cz

Abstrakt. Rozlišení významů slov (disambiguace) je jednou z aktuálních úloh zpracování přirozeného jazyka a nachází uplatnění v oblastech od automatického překladu až po extrakci znalostí z textu. Cílem disambiguace je víceznačnému slovu přiřadit značku odpovídající správnému významu slova. Příspěvek se zabývá jednou z metod využitelných pro disambiguaci – bayesovským disambiguátorem. Tato metoda rozlišuje významy slov na základě kontextu, ve kterém se vyskytují. Ve standardní verzi se kontext považuje za neuspořádanou množinu slov. Tento přístup jsme se snažili vylepšit uplatněním dalších vlastností využitelných pro disambiguaci. Příspěvek obsahuje popis bayesovské disambiguace a navrhuje několik heuristických úprav zlepšujících její přesnost. Podle dosud provedených testů poskytují modifikované algoritmy slibné výsledy, přesnost se pohybuje v závislosti na tématické podobnosti trénovacího a testovacího korpusu v rozmezí 50 % - 90 %.

Klíčová slova: disambiguace, víceznačnost, EuroWordNet, přirozený jazyk, kolekce, synset.

1 Úloha disambiguace v NLP

Rozlišení významů slova je nezbytným krokem pro většinu aplikací zpracovávajících přirozený jazyk (Natural Language Processing, NLP). Jedná se o klíčovou úlohu pro správné porozumění sdělení, uplatňuje se v komunikaci člověk-počítač. Jako příklad lze uvést automatický překlad, kde se disambiguace využívá pro nalezení správné interpretace víceznačného slova. Mějme anglické slovo *bank*, jenž lze přeložit mimo jiné jako *břeh* nebo *banka*. Správný překlad vyplývá z kontextu, ve kterém je slovo použito a je zřejmé, že překlady nelze zaměňovat.

Disambiguaci v tomto článku chápeme jako klasifikaci víceznačného slova do tříd, které představují vždy jeden význam slova. Možné významy jsou typicky vyjmenovány ve slovníku, kde mohou být uvedeny i doplňující atributy pomáhající disambiguaci (např. synonyma, vztahy k ostatním slovům, slovní druh, apod.). Zařazení slova do třídy je ovlivněno jeho kontextem, případně další informací získanou ze slovníku, tezauru, encyklopedie, či jiného lexikálního zdroje. Často je jako zdroj informací použitý tezaurus EuroWordNet (EWN), který je blíže popsaný v kap. 2, využití pro disambiguaci je popsáno také v [2] a [3].

Z výsledků analýzy textů jsme zjistili, že téměř 20% slov je víceznačných. To poukazuje na důležitost disambiguace při zpracování přirozeného textu prakticky ve všech oblastech NLP.

2 Tezaurus EuroWordNet

Tezaurus EuroWordNet (EWN) obsahuje slova uspořádaná do množin synonym (tzv. synsetů). V každém synsetu jsou slova podobného významu a mají přiřazenou unikátní značku (index), která je shodná ve všech jazycích EuroWordNetu. Tezaurus obsahuje následující evropské jazyky: angličtina, dánština, italština, španělština, němčina, francouzština, čeština, estonština. EWN je strukturovaný podobně jako původní Wordnet vytvořený Princetonskou univerzitou.

Jelikož jsou značky pro jednotlivá slova shodné v různých jazycích, lze vhodně navrženým disambiguátorem provádět také křížovou disambiguaci – tzn. trénování provést na kolekci v jednom jazyce a rozlišovat významy slov v jiném jazyce. Taková vlastnost je výhodná především v případě jazyků, pro které nemáme dostatek trénovacích dat.

V současné době nejsou jednotlivé wordnety stejně rozsáhlé. Některé synsety nemají v určitých jazycích odpovídající ekvivalent. To vede k situaci, kdy se slovo spojí se značkou, která nemá v jiném jazyku překlad, což by činilo problém zejména při výše zmíněné křížové disambiguaci.

3 Disambiguační metody

Tyto metody lze dělit podle způsobu trénování. V případě, že máme k dispozici označkovaný korpus, mluvíme o metodách s učitelem. V označkovaném korpusu má každé víceznačné slovo přidruženou značku, která určuje jeho význam. Korpus se ve většině případů značkuje ručně.

Druhou skupinou disambiguačních algoritmů jsou metody bez učitele. Není nutná žádná apriorní informace o významech jednotlivých slov, tedy odpadá nutnost značkování trénovacího korpusu. Takovou disambiguaci lze považovat za úlohu shlukovaní víceznačných slov podle významů.

Rozhodli jsme se zaměřit především na metody s učitelem. Cílem bylo vytvořit disambiguátor, který dokáže víceznačným slovům přiřadit značku (index) uvedenou v tezauru EWN. V případě použití metod bez učitele není možné takového výsledku dosáhnout, jelikož není zřejmé, jaké jsou vazby mezi shluky získanými při disambiguaci a jejich indexy v EWN.

4 Bayesovská disambiguace

V případě použití metody patřící do skupiny disambiguace s učitelem je nutné poskytnout algoritmu trénovací data, nejčastěji ve formě označkovaného korpusu. Každý výskyt víceznačného slova w je označkován příslušným významem s_i^w (reprezentovaný indexem EWN), který nazveme sémantickou značkou (pro zjednodušení zápisu budeme dále psát pouze s_i). Tento přístup převádí problém disambiguace na problém klasifikace do k tříd, kde k je počet významů slova. Každá třída odpovídá jednomu významu s_i slova w, kde i nabývá hodnot $1,2, ..., k$.

Bayesovská disambiguace vytváří množinu slov *c* (obklopující víceznačné slovo) bez vnitřní struktury a bez rozlišení důležitosti, či vzájemných vztahů mezi nimi. Následně danou množinu využívá pro rozlišení významu slova ve fázi klasifikace. Absenci vztahů mezi slovy a malou volnost při volbě paramertů jsme považovali za limitující a navrhli jsme několik modifikací popsaných v kap. 5.

Základní verze bayesovského disambiguátoru (více viz [1], [4]) aplikuje Bayesovo pravidlo pro výběr správného významu *s'*:

$$P(s'|c) = \max_{s_i} P(s_i|c) \tag{1}$$

Bayesovo pravidlo minimalizuje pravděpodobnost výskytu chyb. Toto tvrzení je pravdivé, jelikož pro každé nejednoznačné slovo vybere význam s nejvyšší podmíněnou pravděpodobností.

Postupnými úpravami a aplikováním tzv. „Laplace smoothing" dojdeme ke vzorci (2):

$$P(v_j|s_i) = \frac{C(v_j, s_i) + mp}{C(s_i) + m}, \tag{2}$$

kde v_j jsou slova v množině kontextu *c*, s_i je jeden z významů víceznačného slova *s*, $C(v_j, s_i)$ je počet výskytů slova v_j v množině *c* s významem s_i a $C(s_i)$ je počet výskytů významu s_i v celém textu.

Laplace smoothing lze chápat jako přidání dalších *m* vzorků k existujícím hodnotám výskytů podle apriorního rozdělení odhadu pravděpodobnosti *p*. Konstantu *m* nazýváme ekvivalentní velikost vzorku a určuje váhu *p* vzhledem k trénovacím datům. Protože nemáme dostatek informací o datech, budeme volit $m = k$, $p = 1/k$, kde *k* představuje počet slov v množině *c*.

Dosazením *m* a *k* do vzorce (2) dojdeme k následujícímu vztahu:

$$P(v_j|s_i) = \frac{C(v_j, s_i) + 1}{C(s_i) + C(v_j)}, \tag{3}$$

kde $C(v_j)$ je počet různých slov v_j v množině *c*.

5 Modifikace bayesovské disambiguace

Výše uvedené vzorce budeme modifikovat tak, aby byly minimalizovány nedostatky bayesovské disambiguace. Především se budeme snažit zvýhodnit některá perspektivní slova (z hlediska disambiguace) a naopak potlačit taková, která nepřináší žádnou informaci použitelnou pro rozlišení významu víceznačných slov.

5.1 Prahové hodnoty

Malá četnost výskytu některého kontextového slova v_j z okolí víceznačného slova může vést k systematickému ovlivnění hodnoty výrazu (2). Taková slova se z lexikálního hlediska podílejí na disambiguaci minimálně a pouze zanáší šum do výpočtu. Proto jsme definovali prahové hodnoty k odfiltrování slov s nízkou hodnotou podmíněné pravděpodobnosti (2). Uvažujeme kontextová slova, která splňují:

$$c \in \{v_j \mid P(v_j \mid s_i) > threshold\} \tag{4}$$

5.2 Kontextové okénko

Zavedením metriky *M* dokážeme omezit kontext pouze na určitý rozsah textu napravo a nalevo od víceznačného slova *w*. Prahová hodnota vzdálenosti od slova *w* určuje počet objektů zahrnutých do kontextu *c*. Metrikou *M* chápeme vzdálenost slova od jiného slova, věty či odstavce.

$$c \in \{v_j \mid \|v_j, w\|_M < threshold\} \tag{5}$$

Dále je možné omezit kontextové okénko pouze na větu či odstavec. Vycházíme z předpokladu, že silné kontextové vztahy jsou v rámci jedné věty, případně odstavce.

5.3 Pružné kontextové okénko

Pro účely disambiguace není vždy vhodné provádět odstranění nevýznamových slov (stopslov). Taková slova (např. předložky) často dobře rozlišují význam slova, se kterým se pojí. Přesto není vhodné mít v okénku pouze nevýznamová slova. Proto jsme zavedli tzv. pružné kontextové okénko, které zvětší svou velikost tak, aby obsahovalo alespoň *n* významových slov. V případě n=1 platí:

$$c \in \{v_j \mid \|v_j, w\|_M < \min_{v'_j} \|v'_j, w\|_M\} \tag{6}$$

v_j jsou slova z okolí víceznačného slova *w*, v'_j jsou významová slova z okolí.

5.4 Váhová funkce

Zavedení váhové funkce předpokládá, že slova vzdálenější od disambiguovaného slova na něj mají menší vazbu. Taková závislost se zavede upravením členu $C(v_j, s_i)$ vzorce (3) vynásobením váhovým koeficientem *q*:

$$q = r^{\|v_j, w\|_M}, \; r \in (0,1) \tag{7}$$

5.5 Syntaktická analýza

Díky značkám v trénovacím korpusu, které kromě významu slova obsahují také označení slovního druhu, je možné provádět syntaktickou analýzu. Podle slovního druhu víceznačného slova je přiřazena určitým druhům kontextových slov v_j větší váha. Zvýhodnění takových slov se provádí vynásobením vzorce (2) váhovým koeficientem (viz tab. 8). V případě disambiguace slovesa se zvýhodňuje podstatné jméno, přídavné jméno a jiné sloveso, u podstatného jména se zdůrazňuje váha slovesa, podstatného jména a zájmena a u přídavného jména se provádí zvýhodnění podstatných jmen. Syntaktické vztahy se určují automaticky a uplatňují se v rámci jedné věty nebo celého kontextového okénka.

6 Datové kolekce

Pro natrénování disambiguátoru jsme použili textový korpus semcor [5] (Semantic concordance). Věty jsou označkované indexy WordNetu.

Tabulka 1. Parametry semcor korpusu

Slova	198796
Slova obsahující sémantickou značku	106724
podstatná jména obsahující sémantickou značku	48835
podstatná jména s různými významy	11399
slovesa obsahující sémantickou značku	26686
slovesa s různými významy	5334
přídavná jména obsahující sémantickou značku	19856
přídavná jména s různými významy	5205
příslovce obsahující sémantickou značku	11347
příslovce s různými významy	1455

Pro lemmatizaci byl použit anglický slovník čítající 119486 slov a stoplist obsahující 48 nevýznamových slov.

7 Výsledky testů

V testech bylo sledováno působení lemmatizátoru, stoplistu a pružného okénka (viz odst. 5.3) na přesnost disambiguace.

Použité zkratky v tabulkách

Zkratka	Význam
-	základní klasifikátor
L	klasifikátor používá lemmatizátor
S	klasifikátor vynechává nevýznamová slova
D	použito pružné kontextové okénko
P	přesnost disambiguace
cT	velikost kontextového okénka při trénování
cD	velikost kontextového okénka při disambiguaci

7.1 Velikost kontextového okénka

Testovali jsme vliv velikosti kontextového okénka na přesnost disambiguace. V tab. 2 jsou uvedeny dosažené přesnosti pro případy použití/nepoužití lemmatizace, stop slov, pružného okénka a jejich vybrané kombinace v závislosti na velikosti okénka při trénování a testování.

Tabulka 2. Velikost okénka – jednotka jsou slova

	-	L	S	SD	LS	LSD
P	87,55	85,49	95,43	**95,56**	93,61	93,77
cT	5	5	10	5	10	10
cD	5	5	10	12	10	10

Další výsledky udávají přesnost pro případ, že trénovaní okénko má velikost jedné věty (tab. 3) a velikost jednoho odstavce (tab. 4).

Tabulka 3. Velikost okénka – jednotka jsou věty (trénování) a slova (testování)

	-	L	S	LS
P	91,35	88,38	**95,95**	94,36
cD	35	17	17	17

Tabulka 4. Velikost okénka – metrika *M* jsou odstavce (trénování), slova (testování)

	-	L	S	LS
P	78,55	76,04	**91,22**	88,48
cD	40	35	40	40

V tab. 5 jsme uvažovali délku okénka ve slovech, avšak okénko nesmělo přesáhnout do vedlejší věty, případně vedlejšího odstavce (tab. 6). Z tabulek je zřejmé, že výsledná přesnost disambiguace se zlepšila zhruba o 1 %, což se na první pohled může zdát málo, ovšem chybovost metody tím byla snížena o 20 % (z 5 na 4 %).

Tabulka 5. Velikost okénka – jednotka je slovo, omezení kontextu na větu

	-	L	S	SD	LS	LSD
P	88,46	86,48	96,02	**96,37**	94,60	94,97
cT	5	5	5	5	5	10
cD	5	5	12	10	12	12

Tabulka 6. Velikost okénka – jednotka je slovo, omezení kontextu na odstavec

	-	L	S	SD	LS	LSD
P	87,78	85,72	95,61	**95,79**	93,85	94,07
cT	5	5	10	5	10	5
cD	5	5	10	12	10	10

7.2 Zvýhodňování blízkých slov

Byla použita váhová funkce z odst. 5.4 s koeficienty uvedenými v tabulce 7. Zvýhodňování blízkých slov nepřineslo žádné zlepšení. Naopak se ukázalo, že pokud se nezvýhodňuje v závislosti na vzdálenosti od víceznačného slova, poskytuje algoritmus nejlepší výsledky (pro q=1). Pro test byla použita stop slova a pružné kontextové okénko.

Tabulka 7. Úspěšnost v závislosti na zvýhodnění

koeficient q	0,1	0,25	0,5	0,75	1,0
P	95,23	95,48	95,71	95,77	**95,78**

7.3 Syntaktické vztahy v textu

Tímto testem jsme ověřili přínos modifikace využívající syntaktických vztahů ve větách. Modifikace je popsaná v odst. 5.5. Podle výsledků testů poskytuje v závislosti na datech zlepšení přesnosti o 1 – 2 %, což představuje snížení chyby o téměř 40 %.

Tabulka 8. Úspěšnost v závislosti na uvažování syntaktických vazeb

Zvýhodnění	1,0	2,0	10,0	50,0
P	96,37	96,70	**97,00**	96,91

8 Závěr

Výsledky testů ukázaly, že bayesovská disambiguace poskytuje slibné výsledky s přesností pohybující se přes 90 %. Navržené modifikace přinesly zlepšení od 1 do 5 %, což představuje přínos až 50 % z hlediska snížení chybovosti disambiguátoru.

Předmětem dalšího zkoumání bude využití tezauru EWN pro disambiguační úlohu bez učitele. Chceme také realizovat disambiguaci s využitím paralelních dvoujazyčných textů. V následujících měsících hodláme zahrnout navrženou disambiguaci do systému vyhledávání v multilingválních kolekcích.

Reference

1. Manning, C. D.., Hinrich S., *Foundations of Statistical Natural Language Processing*, The MIT Press 1999, ISBN: 0262133601
2. Resnik, P., *Disambiguating Noun Groupings with Respekt to WordNet Senes*, Proceedings of 4th Workshop on Very Large Corpora, Copenhagen 1996
3. Resnik, P., *Selectional Preference and Sense Disambiguation*, University of Maryland, Annual Meeting of the Association for Computational Linguists 1997.
4. Veronis, J., Ide, N., *Word Sense Disambiguation, State of the Art*, Computional Linguistics 1998.
5. Cognitive Science Laboratory Princeton, http://www.cogsci.princeton.edu/~wn/doc/man/semcor.htm

Annotation:

The Bayesian Disambiguation Modifications

Word Sense Disambiguation is a major sub task of many Natural Language Processing (NLP) tasks, ranging from machine translation to information retrieval. In this paper we present some modifications of Bayessian disambiguation algorithm. We introduce the modified method and its quite promising results. The main advantage lies in better choise of context window and observing more relevant attributes than the standard method does. The precision of the method presented in this paper is about 90 – 95 %.

Tractable Construction of Relational Features

Filip Železný

České vysoké učení v Praze, Fakulta elektrotechnická, Katedra kybernetiky

zelezny@fel.cvut.cz

Abstract. A popular technique for converting multi-relational data into a single-relational form is based on constructing truth-valued relational features of the data instances, where the features play the role of binary attributes in the resulting representation. Here I consider a simple relational feature language whose formulas correspond to conjunctions of first-order atoms where arguments are only variables and respect user-defined constraints on types and input/output modes. I show a sufficient condition for polynomial time construction of such formulas and give preliminary results on tractable enumeration of complete sets of such formulas.

1 Introduction

This paper is concerned with efficient construction of expressions such as

`hasCar(C),hasLoad(C,L1),triangle(L1),hasLoad(C,L2),box(L2)`

corresponding to conjunctions of constant-free, function-free, non-negated atoms and subject to some predefined syntactic constraints. The expression is an example of a *feature* related to an *individual* (here a train, refer to Fig. 1) meaning that the train has a car carrying a triangle-shaped load and a box-shaped load. Note, so far informally, that all variables appearing as *outputs*, eg. `C` in `hasCar(C)`, also appear as *inputs*, eg. `C` in `hasLoad(C,L1)` and vice versa. This will be a general requirement on correct features.

I am not concerned here with the semantics of features and do not treat the problem of determining their truth value for a specific individual. Note however, that for purposes of generating a single-relational representation of a database, a set of features is evaluated with respect to a given individual at a time, and this procedure is then conducted for all available individuals. Due to this item-wise process of feature evaluation, no special, 'key variable' is needed which would link the feature to some individual in the database. Below I will discuss more on the consequences of ignoring the key variable on the syntactic construction of features.

Let me define an atom and an expression (features will be searched among expressions) formally. I assume there are some infinite countable sets S ('predicate symbols') and V ('symbols of variables'). N stands for the set of naturals numbers.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 314–321, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

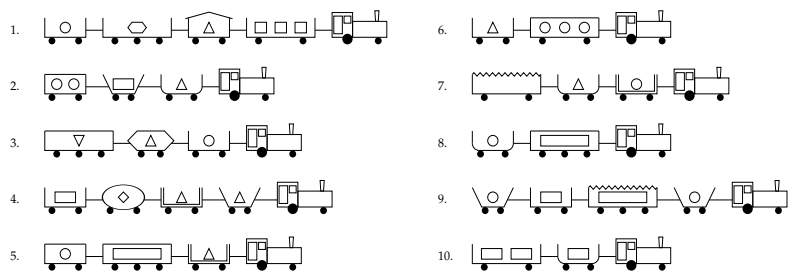


Fig. 1. A data base containing structured items.

Definition 1. $x = s_x(v_{x,1}, v_{x,2}, \ldots, v_{x,a_x})$ *is called an* atom *if* $s_x \in S$, $a_x \in N$ *and* $v_{x,i} \in V$ $(1 \leq i \leq a_x)$. $\mathbf{L}$ *denotes the set of all atoms. Any finite* $e \subseteq \mathbf{L}$ *is called an* expression. *Denote* $Arg(e) = \{(x,i) \mid x \in e, 1 \leq i \leq a_x\}$ *and* $Var(e) = \{v_a \mid a \in Arg(e)\}$. $\mathbf{E}$ *denotes the set of all expressions.*

From the semantic point of view, the atom order is irrelevant. This allows to view expressions simply as *sets* of atoms. Note that the indexation in $v_{x,i}$ and a_x should be understood as a functional notation, ie. $v_{l,j}$ (v_{l,a_l}) will always represent the variable at the j-th (last, respectively) argument in atom l (a_l is called the *arity* of l).

The concept of *substitution* also acquires a simple meaning in this constrained framework.

Definition 2. *Let* $e \in \mathbf{E}$. *A* substitution *is a mapping* $\theta : V \to V$. *For* $x \in e$, $x\theta = s_x(\theta(v_{x,1}), \theta(v_{x,2}), \ldots, \theta(v_{x,a_x}))$ *and* $e\theta = \{x\theta \mid x \in e\}$. *Expression* e θ-subsumes *expression* f *(denoted as* $e \preceq_\theta f$*) iff there is a substitution* θ *such that* $e\theta \subseteq f$. *Finally,* e *is* equivalent *to* f *(written* $e \approx f$*) iff* θ *is bijective and* $e\theta = f$.

A specific feature language is defined by a *template*, which is made of

- an expression t, determining the argument types and modes of all atoms which may appear in a feature,
- a subset M of argument places in the above expression, which are assigned the input-mode.

The role of typing will simply be that no variable can appear in a feature at two argument places with different types. The reason I use the expression t for specifying types is that any type-compliant expression will then θ-subsume t, as I will exemplify later.

The M subset will simply list the argument places which have to hold an input variable. Although the input-output moding of arguments has a clear

intuitive role (refer to the initial train example), my only formal requirement will be that each variable present in a feature appears at both some input and some output argument place. This requirement leads to closed-form, well-interpretable features [3].

The following definition formalizes the above. Further it defines when a variable is *proper*, ie. when it has both an input and an output role in an expression, and defines which expression is a feature.

Definition 3. *A* template τ *is a pair* (t_τ, M_τ) *where* t_τ *is an expression and* $M_\tau \subseteq Arg(t_\tau)$. $\mathbf{T}$ *denotes the set of all templates. Let* $e \preceq_\theta t_\tau$. *Denote* $Arg^+_\tau(e) = \{(x,i) \in Arg(e) \mid (x\theta, i) \in M_\tau\}$ *and* $Arg^-_\tau(e) = Arg(e) \setminus Arg^+_\tau(e)$. *If some* $v \in V$ *satisfies the equivalence* $(v = v_{a^+}, a^+ \in Arg^+_\tau(e)) \Leftrightarrow (v = v_{a^-}, a^- \in Arg^-_\tau(e))$, *then* v *is said to be* τ-proper *in* e . *A non-empty expression* f *is a* τ-feature *iff* $f \preceq_\theta t_\tau$ *and each* $v \in Var(f)$ *is* τ*-proper in* f.

Example 1. Consider a template (t_τ, M_τ) suitable for the initial train example

$$t_\tau = \texttt{hasCar(C)}, \texttt{hasLoad(C,L)}, \texttt{triangle(L)}, \texttt{box(L)}$$
$$M_\tau = \{(\texttt{hasLoad(C,L)}, 1), (\texttt{triangle(L)}, 1), (\texttt{box(L)}, 1)\}$$

The typing constraint is here defined by the t_τ expression, isolated from the *moding* constraint. This makes it explicit that verifying compliance to a typing constraint corresponds to a subsumption check. Indeed, consider an expression

$$e = \texttt{hasCar(X)}, \texttt{hasLoad(X,Y)}, \texttt{triangle(Y)}, \texttt{hasLoad(X,Z)}$$

e complies to the typing specified by t_τ because $e \preceq_\theta t_\tau$.[1]

Each variable of a feature must occur at both an argument contained in M_τ and an argument not in M_τ. Therefore, e above is not a feature, since `Z` is at no argument in M_τ.

Note that every template τ has a dual template τ^{-1} with inverse moding $M^{-1}_\tau = Arg(t_\tau) \setminus M_\tau$ and every τ-feature is also a τ^{-1}-feature. Through the inversion, every 'primary structural' (such as `hasCar(C)`) becomes a 'property' and every property (such as `triangle(L)`) becomes a primary structural. A theoretical consequence is that if a feature class (say features where atoms chained by variable sharing form a 'tree') can be efficiently enumerated, then the inverse class (here atoms forming a 'root') can also be efficiently enumerated. The symmetric properties commented above are a result of disregarding the key (individual-linking) variable, which would occur only at some 'input' arguments (those in M_τ). In a sense, the sole role of the key type lies in setting the *orientation* of the otherwise symmetric features, whereas the orientation is irrelevant for sakes of feature construction.

Note that also the notation (t_τ, M_τ) above is understood functionally, ie. for any template $\rho \in \mathbf{T}$, t_ρ represents the prescribed typing of ρ and M_ρ its moding. Let me now specify the main problem treated in the remainder of this paper.

[1] In general, there may be more than one substitution θ such that $e \preceq_\theta t_\tau$. A straightforward way to avoid this ambiguity is to constraint oneself, quite naturally, to templates where t_τ contains each symbol $s \in S$ (such as `hasCar`) at most once.

Definition 4. *Let $T \subseteq \mathbf{T}$ and $E : T \to 2^{\mathbf{E}}$. The* feature existence problem *for T and E is defined as follows. The problem instance is the tuple $n \in N$, $\tau \in T$. The instance size is n. The instance solution is a τ-feature f such that $|f| \leq n$ and $f \approx f'$ for some $f' \in E(\tau)$ (if such f exists), or "NO" otherwise.*

The function E takes a template $\tau \in T$ and produces a set of expressions in which τ-features are searched (due to the $f \approx f' \in E(\tau)$ requirement, a solution may be not be in $E(\tau)$ but must be equivalent to some expression in $E(\tau)$; this avoids dependency on variable naming). The reason for specifying the problem class this way is that different complexity results can be proved for different functions E's. For example, $E(\tau)$ may consist of *connected* expressions (where all atoms are linked via the transitive closure of the variable sharing relation) and then be independent of τ. Less trivially, $E(\tau)$ may be such that all τ-features therein are *loop-free* (edges between atoms given by variable sharing, orientation given by moding) and thus obviously depend on the moding of the specific τ.

2 The Bottom Set Theorem

I will first show that the problem of constructing an expression out of a finite set of available atoms, such that a given variable is proper (has both an input and an output occurrence) in that expression, is equivalent to a problem of satisfying a set of propositional Horn clauses.

Lemma 1. *Let τ be a template, $e = \{l_1, l_2, \ldots, l_p\}$ and $e' \subseteq e$ two expressions, $P = \{P_1, P_2, \ldots P_p\}$ a set of propositional variables and $v \in Var(e)$. Further let*

$$\{in_1, in_2, \ldots, in_r\} = \{1 \leq in \leq p \mid \exists i \ (l_{in}, i) \in Arg^+_\tau(e), v_{l_{in},i} = v\} \quad (1)$$

$$\{out_1, out_2, \ldots, out_s\} = \{1 \leq out \leq p \mid \exists i \ (l_{out}, i) \in Arg^-_\tau(e), v_{l_{out},i} = v\} \quad (2)$$

be two index sets.[2] *Let further $C_{in}(v)$ denote the following set of Horn clauses*

$$P_{in_1} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \quad (3)$$
$$P_{in_2} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \quad (4)$$
$$\vdots \quad (5)$$
$$P_{in_r} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \quad (6)$$

Similarly, let $C_{out}(v)$ be the following Horn clause set

$$P_{out_1} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \quad (7)$$
$$P_{out_2} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \quad (8)$$
$$\vdots \quad (9)$$
$$P_{out_s} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \quad (10)$$

[2] The former index set thus addresses those of atoms in e, which contain v at some input argument while the latter set corresponds to atoms with v acting as an output.

Let $C(v) = C_{in}(v) \cup C_{out}(v)$ *and* $\xi_{e'} : P \rightarrow \{true, false\}$ *be a truth assignment*

$$\xi_{e'}(P_i) = \begin{cases} false, \text{ if } l_i \in e'; \\ true, \quad \text{if } l_i \notin e'. \end{cases} \tag{11}$$

Then v *is* τ*-proper in* e' *iff* $\xi_{e'}$ *satisfies all clauses in* $C(v)$*.*

Proof. See `http://labe.felk.cvut.cz/~zelezny/pubs/znalosti05.pdf`.

It is now straightforward to extend the previous lemma to the problem of finding a non-empty expression with all variables proper.

Lemma 2. *Let all assumptions of Lemma 1 hold. Let further* $C = \bigcup_{v \in Var(e)} C(v)$ *and* $C_\emptyset = \{\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_p\}$ *Then the following assertions are equivalent:*

1. *Expression* e' *is non-empty and each* $v \in Var(e)$ *is* τ*-proper in* e'*.*
2. *Assignment* $\xi_{e'}$ *satisfies all clauses in the Horn clause set* $C \cup C_\emptyset$*.*

Proof. See `http://labe.felk.cvut.cz/~zelezny/pubs/znalosti05.pdf`.

Lemma 3. *Let all assumptions of Lemma 2 hold. A maximal assignment (ie. one assigning the* false *value to the smallest number of variables) satisfying all clauses in* C *can be found (or decided that no satisfying assignment exists) in time polynomial in* r *and* s*.*

Proof. See `http://labe.felk.cvut.cz/~zelezny/pubs/znalosti05.pdf`.

I am finally in the position to show the main result of this paper, which is informally as follows. If a polynomial set $\perp$ can be constructed such that all acceptable features (up to variable renaming) are subsets thereof (let me call $\perp$ a *bottom set*), one can find a feature (if it exists) in polynomial time. This is despite the fact that there is of course an exponential number of subsets of the bottom set.

Theorem 1. *Let* $T \subseteq \mathbf{T}$*,* $E : T \rightarrow 2^{\mathbf{E}}$ *and let there be* $\perp : T \times N \rightarrow \mathbf{E}$ *such that for all* $\tau \in T$*,* $n \in N$*:* $\perp(\tau, n)$ *is computable in time polynomial in* n*,* $\perp(\tau, n) \preceq_\theta t_\tau$*, and for all* τ*-features* f *it holds that* $f \approx f' \in E(\tau)$ *iff* $f \subseteq \perp(\tau, |f|)$*. Then the feature existence problem for* T *and* E *can be solved in polynomial time.*

Proof. See `http://labe.felk.cvut.cz/~zelezny/pubs/znalosti05.pdf`.

Example 2. The bottom set for τ specified in Example 1 and $n = 3$ is

$$\perp(\tau, n) = \texttt{hasCar(C),hasLoad(C,L),triangle(L),box(L).}$$

Clearly, all τ-features (up to variable renaming) of length up to 3 atoms are subsets of this bottom set (incidently equivalent to t_τ). Let the propositional variables assigned to the bottom atoms (in the order of their appearance) be P_1, P_2, P_3, P_4. The corresponding HORNSAT instance is the union of the following Horn clause sets

$C_{in}(\texttt{C}) = \{P_2 \vee \neg P_1\}$,
$C_{out}(\texttt{C}) = \{P_1 \vee \neg P_2\}$,
$C_{in}(\texttt{L}) = \{P_3 \vee \neg P_2, P_4 \vee \neg P_2\}$,
$C_{out}(\texttt{L}) = \{P_2 \vee \neg P_3 \vee \neg P_4\}$
$C_{\emptyset} = \{\neg P_1 \vee \neg P_2 \vee \neg P_2 \vee \neg P_4\}$.

One of the maximal solutions assigns the *false* value to P_1, P_2 and P_3 (the reader will check that all mentioned clauses are satisfied), which corresponds to the set `hasCar(C),hasLoad(C,L),triangle(L)`, which therefore forms a correct feature.

3 Bottom Set Existence and Feature Enumeration

The bottom set theorem leaves two crucial questions open:

- When can a polynomial bottom set be constructed?
- When does tractability of the feature existence problem entail tractability of the enumeration of *all* τ-features in $E(\tau)$?

The following results are given without proofs, which will appear in an extended version of the paper.

3.1 Bottom Set Existence

Let me first formalize a few structural notions about templates and features.

Definition 5. *Let $T \subseteq \mathbf{T}$, $E : T \to 2^{\mathbf{E}}$, $\tau \in \mathbf{T}$ and $e \in \mathbf{E}(\tau)$. $x, y \in e$ are said to be* connected *in e iff they share a variable or some $z \in e$ is connected with both x and y. e is is said to be* connected *(or* undecomposable*) iff any atom in e is connected to all other atoms in e. There is a* path *in a τ-feature f from $x \in f$ to $y \in f$ of length 1, iff for some $a, b \in Arg(f)$ it holds $v_a = v_b$, $a \notin M_\tau$, $b \in M_\tau$, or a path of length $l + 1$, iff for some $z \in f$ there is a path from x to z of length l and a path from z to y of length 1. The* distance *$\delta_\tau(x, y)$ is the length of the shortest path from x to y, if one exists. The* depth *of f is defined as $\Delta_\tau(f) = \max_{u,v \in f} \delta_\tau(u, v)$. f is said to be* loop-free *if for no x there is a path in f from x to x. A loop-free τ-feature f is called a* semi-root (semi-tree) *iff no variable has two input (output) occurrences in f w.r.t τ; f is called a* root (tree) *iff it is a semi-root (semi-tree) and no atom in f has two output (input) arguments w.r.t τ. Further, f is called a* semi-chain (chain) *iff it is simultaneously a semi-root (root) and a semi-tree (tree). Finally, τ is said to be hierarchical iff there is a partial irreflexive order $\prec$ on $Var(t_\tau)$ such that $v_{l,i} \prec v_{l,j}$ whenever there are l, i, j such that $(l, i) \in Arg^+_\tau(t_\tau)$ and $(l, j) \in Arg^-_\tau(t_\tau)$.*

The next theorem gives three sufficient conditions, each of which allows for the construction of a polynomial-size bottom set, and therefore also the polynomial-time construction of a feature.

Theorem 2. *Assumptions of Theorem 1 are satisfied if for each $\tau \in T$: every $e \in E(\tau)$ is connected and any of the following holds:*

1. *each τ-feature in $E(\tau)$ is either a tree, root or chain,*
2. *there is a $\Delta_{max} \in N$ such that every τ-feature f in $E(\tau)$ is loop-free and $\Delta_\tau(f) \leq \Delta_{max}$,*
3. *τ is hierarchical (this implies Cond. 2).*

Let me illustrate the spirit of the proof with an example.

Example 3. Consider again the continued train example. For each template τ, let $E(\tau)$ be the set of all *connected* τ-features, ie. expressions such as

$$\texttt{hasCar(C),triangle(C),hasCar(D),triangle(D)}$$

are disallowed. Let further T consist of *hierarchical* templates $\tau = (t_\tau, M_\tau)$, such as

$$t_\tau = \texttt{hasCar(C)}, \texttt{hasRoof(C)}, \texttt{hasLoad(C,L)}, \texttt{triangle(L)}, \texttt{box(L)} \quad (12)$$

$$M_\tau = \{(\texttt{hasLoad(C,L)}, 1), (\texttt{triangle(L)}, 1), (\texttt{box(L)}, 1)\} \quad (13)$$

There is of course no τ-feature of size $n = 1$. For $n = 2$, a bottom set (coinciding with the only correct τ-feature) is

$$\bot(\tau, 2) = \texttt{hasCar(C),hasRoof(C)}$$

For $n = 3$, a bottom set is

$$\bot(\tau, 3) = \texttt{hasCar(C),hasRoof(C),hasLoad(C,L),box(L),triangle(L)}$$

In general, for

$$3 + 3p \leq n < 3 + 3(p + 1),\ p = 1, 2, \ldots \quad (14)$$

$\bot(\tau, n)$ can be obtained by adding the atoms

$$\texttt{hasLoad(C,L}_{p+1}\texttt{),box(L}_{p+1}\texttt{),triangle(L}_{p+1}\texttt{)}$$

to $\bot(\tau, n-1)$. This is better seen from a *directed graph* representation of a bottom set (below), where the nodes correspond to atoms and left-to-right edges correspond one variable being an output in the left node and an input in the right node.[3]

$$\text{branching factor} \leq n \left\{ \begin{array}{llll} \texttt{hasCar(C)} & -\texttt{hasRoof(C)} & & \\ & -\texttt{hasLoad(C,L1)} & -\texttt{box(L1)} & \\ & & -\texttt{triangle(L2)} & \\ & -\texttt{hasLoad(C,L2)} & -\texttt{box(L2)} & \\ & & -\texttt{triangle(L2)} & \\ \ldots & \ldots & \ldots & \end{array} \right.$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\Delta \leq \Delta_{max}}$$

[3] Formally, there is an edge between nodes of atoms x and y in this representation of $\bot(\tau, n)$, if $v_1 \in Var(x)$ and $v_2 \in Var(y)$ and $\theta(v_1) \prec \theta(v_2)$ where $\prec$ is the assumed hierarchy-defining order, and θ is a substitution mapping the bottom set variables to the variables in t_τ (types) such that $\bot(\tau, n)\theta \subseteq t_\tau$.

Here every $\tau-$feature is some connected subgraph of the above graph without orientation. Under the specific choice of τ (Eq. 12), the graph is a tree, which is because all atoms in t_τ have at most one input variable. Due to the assumed hierarchical ordering of types $\prec$, the depth of the graph is bounded by a constant[4] Δ_{max}. Also its branching factor can be upper-bounded by n (eg. no feature of size at most n can address more loads than n). The number of nodes, ie. the size of the bottom set is thus of order $n^{\Delta_{max}}$, that is, polynomial in n. Due to Theorem 2, this guarantees a polynomial-time construction of features.

However, a different choice of τ in Eq. 12 may produce a non-tree graph if atoms with multiple inputs appear in t_τ. Without elaborating the detailed proof, the atoms can still be organized in 'layers' (like above) using the assumed type hierarchy $\prec$. The branching factor is then of order n^{InpAr} where $InpAr$ is the maximum number of inputs in an atom, among atoms in t_τ. In this case $|\bot(\tau, n)|$ is of order $n^{InpAr \times \Delta_{max}}$ which again preserves polynomial-time constructibility of features.

3.2 Enumeration of Feature Sets

The *feature enumeration problem* is the same as the problem of feature existence, except that a solution to the former problem is the set of all distinct (ie. mutually non-equivalent) solutions to the latter problem. An auxiliary, yet important lemma for this problem is as follows.

Lemma 4. *Let the feature existence problem for T and E be decidable in polynomial time and let there be a polynomial number of distinct solutions to every instance thereof. Then the feature enumeration problem for T and E can be decided in polynomial time.*

Proof. Direct consequence of the enumeration-tractability result in [1]. □

A preliminary analysis indicates that satisfying the conditions stipulated by Theorem 2 implies a polynomial number of solutions to the existence problem and hence the polynomial decidability of the enumeration problem.

References

1. R. Dechter and A. Itai. Finding All Solutions if You can Find One *AAAI-92 Workshop on Tractable Reasoning*, 1992
2. W. F. Dowling and J. H. Gallier. Linear time algorithm for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267-284, 1984.
3. N. Lavrač and P. A. Flach. An extended transformation approach to inductive logic programming *ACM Transactions on Computational Logic* 2:4, 2001
4. N. Lavrač, F. Železný and P. A. Flach. RSD: Relational Subgroup Discovery through First-Order Feature Construction *12th Int. Conf. on Inductive Logic Programming*, Springer 2002
5. Thomas J. Schaefer. The complexity of satisfiability problems. *Tenth Annual Symposium on Theory of Computing*, 1978.

[4] assuming implicitly a fixed size of t_τ

Clementine a text mining

Ondřej Háva

SPSS CR, Krakovská 7, 110 00, Praha 1

Hava@spss.cz

Abstrakt. Primární potřebou data minera je efektivně připravit řešení celého projetu od definování úlohy až po jeho uvedení do praxe tak, jak je obecně popsáno metodologií CRISP-DM. Vhodným nástrojem pro splnění těchto cílů je data miningový software Clementine. Efektivita práce je v Clementine podporována především zdařilým grafickým rozhraním pro tvorbu vizuálních programů a integrací všech důležitých nástrojů do tohoto rozhraní. Nástroj umožňuje i hladkou integraci externích algoritmů. Příkladem může být nástroj pro zpracování nestrukturovaných data LexiQuest. Jeho přidáním do Clementine získá uživatel výhodu jedného zpracování a spojení strukturovaných a nestrukturovaných dat.

Klíčová slova: data mining, text mining, extrakce konceptů

1 Úvod

Význam pojmu data mining je u nás chápán značně rozdílně. Mnoho lidí hovoří o data miningu v rozdílných souvislostech. Ještě zkreslenější představy má česká veřejnost o data miningovém softwaru. Důsledkem toho je často neefektivní práce s nástroji, které buď jsou úplně nevhodné nebo se hodí pro data mining jen částečně a je třeba je kombinovat s dalšími produkty, což bývá práce navíc, která odvádí pozornost od vlastního projektu.

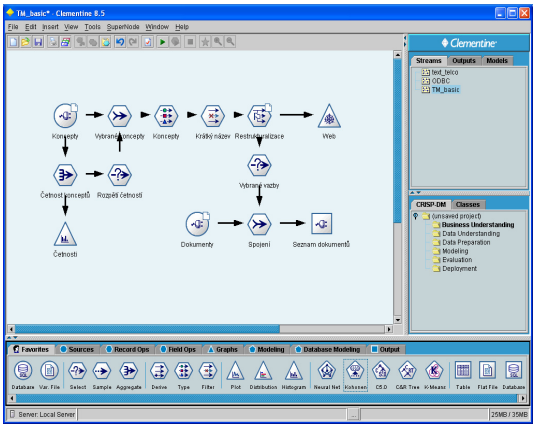
Pro efektivní práci na data miningovém projektu je důležité mít dobrý přehled o celém řešení. Ten by mělo zajistit názorné a praktické grafické rozhraní. Ukazuje se například, že zadávání úloh pomocí programového kódu je častým zdrojem chyb a neumožňuje dostatečnou orientaci ostatním uživatelům. A pro kvalitní data miningový projekt je klíčové, aby na něm pracoval tým složený z různých expertů.

Software představený v příspěvku je již po mnoho let zřejmě nejoblíbenějším data miningovým programem (podle pravidelné ankety na nezávislých stránkách o data miningu http://www.kdnudgets.com) právě pro svou názornost a uživatelskou příjemnost zajišťovanou unikátním grafickým rozhraním. Mezi jeho další nesporné výhody patří integrace potřebných metod a postupů, které pokrývají všechny fáze standardní DM metodologie CRISP-DM.

2 Clementine

Data miningová řešení společnosti SPSS jsou standardně založena na programu Clementine. Tento software není nadstavbou žádné konkrétní databázové aplikace, naopak je otevřený spolupráci s jakoukoli databází a umožňuje práci se všemi standardními datovými formáty včetně například konkurenčního SASu..
Program se ovládá metodou tzv. vizuálního programování. To spočívá ve vytváření názorného diagramu, který popisuje a zadává jednotlivé kroky řešení. Diagram budeme nazývat proudem. Proud můžeme chápat jako model datového procesu, lterý lze snadno modifikovat. Každý proud v Clementine se sestává z funkčních uzlů a jejich spojnic. Uzly reprezentují datové manipulace, spojnice naznačují předávání dat a metadat mezi uzly. Celý proud se konstruuje jednoduše myší, k přesnější specifikaci datových manipulací slouží specifické dialogy, kterými jsou vybaveny všechny uzly.

Příklad proudu v grafickém rozhraní Clementine je uveden na *obrázku 1*. Uživatel má k dispozici uzly umístěné v dolní části rozhraní, které klade na pracovní plochu, kde vytváří vlastní proud. V našem případě se jedná o jednoduchý proud pro vizualizaci vztahů mezi koncepty, které byly extrahovány z předložené množiny dokumentů. Na základě zobrazené mapy konceptů byly vybrány potřebné dokumenty a zobrazen jejich hypertextový seznam.



Obr. 1. Grafické rozhraní Clementine

Uvedený příklad se týká pouze části celého procesu CRISP-DM. Tento proud bychom mohli zařadit do fází přípravy nebo analýzy dat. Celé data miningové řešení by se zřejmě sestávalo z více rozdílných proudů. Výstupem z proudů by byly různé grafy, reporty, modely apod. Clementine podporuje efektivní management všech objektů a proudů v rámci data miningového projektu. Všechny výstupy lze velmi prakticky strukturovat podle členění metodologie CRISP-DM a uložit jako ucelený projekt.

Do vytvářených proudů může uživatel zařadit jednak standardní uzly Clementine a také uzly reprezentující funkcionalitu přidanou pomocí externích programů. Takové uzly se definují pomocí rozhraní CEMI (Clementine External Module Interface), které slouží k začlenění libovolných algoritmů do Clementine. Nyní jsou například oblíbené

uzly pro aplikaci modelovacích algoritmů implementovaných v databázích (MS SQL, Oracle). V našem příkladě je to třeba vstupní uzel pro extrakci konceptů.

Dalším podstatným a praktickým rysem Clementine je její připravenost ke spolupráci s jinými aplikacemi především s databázemi. Clementine může (ale nemusí) delegovat přípravu rozsáhlých dat do databázového stroje. Uživatel nemusí ovlaádat SQL jazyk, stačí opět pouze vytvořit příslušný proud. Příprava dat přímo v databázi je zřejmě rychlejší než v jakémkoli jiném nástroji, a přitom může být z Clementine prováděna stejnými prostředky jako třeba modelování či tvorba grafů.

Stejně snadno jako příprava dat a modelování se provádí i export řešení. Export (ale i import) samotných modelů je realizován standardem PMML. Daleko důležitější než sdílení predikčních modelů je však pro praktické aplikace export celého řešení, kde predikční model tvoří jen jeho část. Celé řešení se pak dá vyjádřit opět jako proud. Existují speciální aplikace z dílny SPSS, které realizují grafické rozhraní nad exportovaným řešením. Rozhraní může být zpřístupněno velkému množství uživatelů pomocí internetového prohlížeče.

3 Text Mining for Clementine

Zpracování nestrukturovaných dat se provádí v Clementine opět metodou vizuálního programování, tedy konstruováním proudů. Výhodou řešení je mimo jiné možnost snadného spojení heterogenních datových zdrojů a modelování na základě strukturovaných i nestrukturovaných dat současně.

Pro extrakci konceptů z textových dokumentů slouží komponenta Text Mining for Clementine realizovaná přes rozhraní CEMI. Využívá extraktor vytvořený společností LexiQuest, která má s počítačovým zpracováním textů více než dvacetileté zkušenosti. V roce 2002 se LexiQuest stal součástí společnosti SPSS.

Transformace souboru dokumentů do konceptové reprezentace probíhá v několika fázích. Celý proces se nazývá přirozené zpracování jazyka (NLP), protože převádí textovou informaci do podoby vhodné pro počítačové zpracování. Na dokumenty však není pohlíženo pouze jako na množiny slov jako při běžném statistickém zpracování, ale jsou reprezentovány v sémantické rovině svými koncepty.

Hlavní fáze celého procesu jsou:

- **Konverze textových formátů a rozpoznání jazyka.** Extraktor umí zpracovávat dokumenty ve všech standardních formátech jako jsou HTML, PDF, XML, ASCII či formáty MS Office. Pro výběr slovníků, které budou potřeba v následujících fázích, musí LexiQuest správně rozpoznat jazyk, pokud není explicitně zadán uživatelem. Rozpoznání se provádí pomocí tzv. *n-gramů*, což jsou kombinace slov nebo písmen délky *n*. Typický počet n-gramů pro rozpoznání jednoho jazyka je čtyři sta. V současnosti podporuje extraktor angličtinu, francouzštinu, němčinu, španělštinu, holandštinu, italštinu a japonštinu.
- **Identifikace kandidátů na termy.** Termy jsou jednak slova, která se nepodařilo najít ve slovníku, ale především skupiny slov, u nichž byl na základě slovníků určen slovní druh. Termy jsou identifikovány na základě předdefinovaných kombinací slovních druhů. Pro každý jazyk je připraveno zhruba 30 kombinací.

- **Rozpoznání tříd ekvivalentních termů a zahrnutí synonym.** Termy extrahované v předchozí fázi jsou sloučeny do skupin se stejným nebo podobným významem. Při slučování je použito několik algoritmů, jejichž popis překračuje rámec tohoto příspěvku. V této fázi se též využívají slovníky synonym. Každá skupina je nakonec reprezentována jediným vhodným termem.
- **Přiřazení typů.** Každému termu je přiřazena kategorie. Extraktor rozpoznává organizace, osoby, produkty a místa. Navíc je možné definovat vlastní typy přidáním vhodných slovníků.
- **Indexace dokumentů.** Každý rozpoznaný term, reprezentovaný identifikátorem skupiny, je zahrnut do indexace. Při indexaci se vytváří databáze pro záznam pozic a četností termů v kolekci dokumentů.
- **Přiřazení ke vzorům a extrakce událostí.** Poslední fáze, implementovaná do nejnovějších verzí extraktorů, umožňuje vyhledávat kombinace termů na základě zvolených vzorů. Vzory popisují struktury pro rozpoznávání vztahů mezi termy a nabízí tak další hlubší reprezentaci obsahu dokumentů.

Uživatel může proces extrahování modifikovat editací a přidáváním specifických externích slovníků. V grafickém rozhraní Clementine je komponenta Text Mining for Clementine reprezentovaná třemi uzly, které mohou být standardním způsobem umisťovány do vytvářených proudů.

První zdrojový uzel vytváří indexační databázi kolekce dokumentů. Uživatel tak získá strukturovanou reprezentaci předložených dokumentů. Data lze dále zpracovat standardními nástroji Clementine.

Někdy jsou analyzované texty uloženy v databázi přímo jako textové proměnné. Může se jednat například o poznámky či slovní popis jednání se zákazníkem. Extrakci termů z textových proměnných realizuje další speciální uzel. Tentokrát se zřejmě nejedná o zdrojový, nýbrž o manipulační uzel pro zpracování textových proměnných.

Třetí výstupní uzel slouží k vytvoření hypertextového seznamu dokumentů. Zpravidla se používá pro zobrazení výsledků text miningového vyhledávání.

4 Závěr

Komponenta Text Mining for Clementine je relativně nový produkt, který však využívá prověřený extraktor LexiQuest. Do nové verze Clementine jsou opět plánována jeho další vylepšení. Jedná se především o interaktivní rozhraní pro zobrazování extrahovaných termů a pro editaci externích slovníků.

Annotation:

The Clementine and text mining

Data mining software Clementine is a comprehensive tool that enables users to implement all phases of standard CRISP-DM methodology. Clementine is open product for external algorithms. Text mining software LexiQuest takes advantege of its openness and it has become the popular module called Text Mining for Clementine. The module is used to transform unstructured data to tabular form.

Górnik System Implementation of Meta Modeling Approach to Data Mining

Peter Zvirinský

StatConsulting, Warsaw, Poland

peter.zvirinsky@statconsulting.com.pl

Abstract. Solving a data analysis task is an iterative process of searching through a vast space of possible solutions supported by data mining algorithms. To bring some order to this process, there is an attempt to introduce a certain level of abstraction. Abstract classification is suggested for different data mining tasks, algorithms or other data objects. These meta modeling concepts are presented in CWM standard (Common Warehouse Metamodel) and they are ready to be used in data mining applications in order to speed up the work of analyst, to ease the implementation of data mining algorithms and to unify modeling tools provided by various vendors. This article presents a data mining suite called Górnik System that implements a meta modeling approach and follows the CWM standard.

1 Introduction

Increasing amount of information processed in business nowadays led to development of data storage and data analysis techniques. There emerged a need for modern analytical tools supporting the complex work of an analyst. Solving a data analysis task is an iterative process of searching through a vast space of possible solutions supported by data mining algorithms. Tasks of data analyst include: understanding the business problem and formulating it as a analytical task, constructing the data model, choosing from the set of algorithms and their parameter settings. He can use several alternative algorithms simultaneously to solve the specific problem. Furthermore, the same algorithm can be used to solve several different tasks. All this illustrates that the process of data analysis is a non-trivial issue. To bring some standardization to this process, there is an attempt to introduce a certain level of abstraction that will be described in the article. There is a standard called CWM (Common Warehouse Metamodel) that attempts to cover all processes in so-called information supply chain, including data analysis.

In the rest of article we will give an overview of meta modeling approach. We will also present our implementation of system for data analysis based on CWM.

© L. Popelínský, M. Krátký (Eds.): Znalosti 2005, pp. 326–330, ISBN 80–248–0755–6.
VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2005.

2 Meta Modeling Approach

Automated process of extracting and analyzing data is complex and it faces many problems of software development in general, such as proliferation of tools and technologies and high level of complication. As in other fields of engineering customers expect high quality stable and configurable solutions. One of possible obstacles on the way towards building such solutions was a lack of uniform computer systems and software modeling approach that can help to model business processes (data warehousing and analysis in our case). Such modeling approach could be used by software user as missing link that enables transformation of business processes into computer code supporting these processes.

In recent years new software modeling technologies emerged. One of mostly known is UML (Unified Modeling Language) capable of representing models diagrammatically. UML is standard of OMG (Object Management Group) and this standard is broadly adopted for modeling object oriented computer programs. The other OMG's specifications are CWM (Common Warehouse Metamodel) - our focus here, and MOF (Meta Object Facility) - defining concepts in the language and how models of those concepts are stored in XML. As UML is general language for modeling object oriented software and processes, the CWM is one for domain specific languages.

Development of domain specific modeling languages is often seen as one of ways for filling a gap between software system and domain specialist. If domain specific language (a meta model in OMG terminology) is used, a software system user can use terminology and classes from his field. In the scenario of data mining an analyst is using concepts like algorithm, statistical model, model building, model testing, data transforming etc. Models consist of sets of interconnected objects. These objects follow design from a metamodel (CWM in our case) and are usually stored in so-called metadata repositories. Objects from metadata repository can be accessed by all parts of software system including humans interacting with the system. In this context humans are modeling some data analysis processes by manipulating metadata in the repository. This model is then used by software system as a data mining task specification. Results produced by data mining task also follow metamodel definition and are stored in metadata repository.

A typical data analysis process involves some core activities. Usually there are sequences of tasks that are performed repeatedly with changing only some parameters of data analysis process definition. Typical data analysis sequence includes:

- data preprocessing tasks,
- defining a data representations for modeling process,
- definition of data analysis task to be performed over a data,
- building a model - selection of data analysis method to perform specified tasks,
- testing of the model on test data set,
- eventual modification of the model in order to improve it,
- application of the best model to new data.

All this processes can be expressed in CWM modeling language.

Advantages of Meta Modeling. Abstract and formal way for defining a modeling language in the context of data mining and data analysis is an important step forward. Its main advantages are:

- uniform object oriented approach for mining processes metadata,
- easy cross-system interoperability through metadata interchange,
- formal, manageable definition of language for data mining process description and modeling that is easily extendible.

Object oriented approach for metadata gives a chance for clean and easy interaction with system end user. Reality of developing software systems deals with constant change and new challenges. Data mining knowledge discovery and information processing is rapidly advancing and establishing new concepts, algorithms and strategies. Using abstract way for defining descriptive language of such knowledge discovery processes in the software system allows painless adopting of new elements originating from new business requirements and advances in the field of data mining and statistical modeling. These new elements can be added to the system effortlessly by expanding data mining metamodel.

Such approach has some obvious advantages over traditional practice of adding new modules to the system since it can reuse present metamodel concepts. Such reuse makes the adding of new modules easier and, what is the most important, it supports end user in understanding and using new functionality due to use of many original concepts already known by user.

3 Implementation of Meta Modeling Approach – Górnik System

Based on our practical experience with data modeling we recognized the strength of CWM approach. We developed a data mining platform called Górnik System that contains a wide set of tools for data analysis and data modeling such as data processing language, set of statistical tools and data mining algorithms and advanced visualization of data or models. All is located in one integrated environment enabling the effective work with data models.

The system is based on the CWM standard. Thanks to implementation of this meta modeling approach, Górnik System offers a different nature to the whole process of data analysis and modeling. In this system a user deals with abstract data mining objects defined by CWM. This object oriented data modeling seems to be much more intuitive for the analysts, since the concept of objects is simple to understand and operate with. Thanks to CWM standard the constructed models gained on uniformity and are easy to reuse.

3.1 Górnik System Tools

Górnik System offers a wide range of tools that support the analyst's work enabling data transformation, visualization and modeling. Our system enables metadata manipulation and control of analytical processes.

The system is composed of set of components that are built around its basic element - a *Metadata Repository* that stores objects representing various elements of data modeling process such as data models, attribute descriptions, algorithm settings and results. Other components interact with the metadata repository according to tasks they perform.

Because construction and optimization of data models usually requires intense computations, the system uses client-server architecture, in order to ensure the computation effectiveness. Furthermore, we decided to apply a grid computing concept by performing computations on a distributed server. Due to this the system can effectively use available resources by automatic distribution of calculations over different computers, which significantly speeds up the data analysis process.

Main analytic activities covered by system's tools are:

Data Transformations. Górnik System can perform various data transformations by using special language structures. Specification of data transformation can be a part of script language or may be created automatically on the basis of high level specification of the transformation e.g. standardization, normalization, missing values replacement, principal component analysis. The transformations are available for further modifications in form of generated code.

Data Modeling with Data Mining Modules. Data models can be divided into several types concerning what dependencies they describe. Górnik System recognizes a set of model types listed bellow. It can be also easily extended by new model types, if needed. CWM defines such ways for meta model extensions. In the standard Górnik System a five main model types are present:

- classification - each observation is assigned to one of defined classes
- approximation - a continuous value is assigned to each observation
- association rules - the result is a set of associations among data attributes
- survival analysis – a survival function is assigned to each observation
- clustering - each observation is assigned to a certain group in the way ensuring high homogeneity among observations from the same group and low homogeneity among observations from different groups.

Górnik System offers a set of algorithms for construction of a diverse data models, each working with data objects from Metadata Repository e.g. Logistic and Linear Regression, Classification Trees, Self Organized Maps, Survival Model Algorithms, SVM, Backpropagation etc. The algorithms are available as independent modules.

Górnik System allows also to use libraries for mathematical and statistical calculations covering: linear algebra, linear and non-linear optimization, Monte Carlo simulations (MCMC), Principal Component Analysis.

Visualization. In order to make analyst's work easier Górnik System offers several advanced ways of visualization and visual interaction with the constructed models. There are specialized visualizations for data models such as Self Organized Maps, Classification Trees, Association Rules; for model quality measures Lift/ROC and for 2D/3D charts.

4 Conclusions

When developing Górnik System we set a goal to create an integrated environment for modeling with intuitive modeling interface. Using a CWM standards in our system helped us significantly to fulfil this goal and allowed us to reach some other attractive characteristics such as:

- Intuitive use of different methods that saves valuable time of analyst's work thanks to uniform object oriented approach.
- Easy modification, extension and comparison of different methods constructing a data model.
- Possibility to adapt easily available modules to client's needs or to construct new specialized solutions by modification of data mining process metamodel.
- Simplified possible integration with other systems.

Using metamodel like CWM can bring significant benefits in solving various data analysis tasks. It is useful for end users when interacting with the systems based on CWM because it defines a unified language that is easy to learn and use by analysts. The metamodel is helpful also in case of developing new domain specific data analysis methods. It outlines a general language for describing also new methods and tasks and therefore making the extensions of the metamodel not a troublesome issue.

If you wish to obtain more information on Górnik System or data analysis services, feel free to contact the author.

Rejstřík autorů (Author Index)

Editoři:	Lubomír Popelínský, Michal Krátký
Název:	Znalosti 2005, sborník příspěvků
Místo, rok, vydání:	Ostrava, 2005, 1.
Počet stran:	344
Vydavatel:	VŠB–Technická univerzita Ostrava, 17. listopadu 15, 708 33 Ostrava–Poruba
Tisk:	Vydavatelství Univerzity Palackého, Biskupské náměstí 1, 771 11 Olomouc
Náklad:	140 kusů