Július Štuller
Zdeňka Linková (Eds.)

Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu

Seminář projektu programu "Informační společnost"

5. 10. – 7. 10. 2006
hotel Olympia, Zadov

Ústav informatiky AV ČR
Praha, 2006

Volume Editors

Július Štuller
Zdeňka Linková (Eds.)

Ústav informatiky
Akademie věd České republiky
Pod Vodárenskou věží 2, 182 07 Praha 8

Print in Reprostředisko UI MFF, Sokolovská 83, 186 75 Praha 8

ISBN 80–903298–7–X
EAN 978–80–903298–7–4

Předmluva

Projekt *Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu* programu "Informační společnost" (Tématického programu II Národního programu výzkumu) je zaměřen na integraci progresivních teoretických disciplín informatiky a jejich praktických aplikací. Jeho cílem je ověřit a rozvíjet teoretické základy sémantického webu, s ohledem zejména na moderní metody umělé inteligence, popřípadě i metody "computing by nature", například umělé neuronové sítě, evoluční algoritmy nebo dolování z dat, dále speciální (deskripční) logiky, a aplikovat i verifikovat vybrané metody inteligentního zpracování dat a znalostí na vybrané oblasti.

Projekt začal 1. července 2004 a je plánován na dobu 4,5 roku. Na jeho řešení spolupracují vědečtí pracovníci a studenti působící ve třech institucích: Matematicko-fyzikální fakulta Univerzity Karlovy v Praze, Fakulta informatiky Masarykovy univerzity v Brně a Ústav informatiky Akademie věd České republiky v Praze.

Seminář shrnující uplynulé dva roky řešení projektu se uskutečnil 5. – 7. října 2006 v hotelu Olympia, Zadov, na Šumavě. Jeho náplní byla rekapitulace dosažených výsledků za jednotlivé instituce a pracovní skupiny i vystoupení jednotlivých členů řešitelských týmů. Z jejich příspěvků pak vznikl tento sborník.

Rádi bychom poděkovali všem, kteří vznik tohoto sborníku umožnili, především všem autorům. Poděkování patří také Evě Pospíšilové za rychlou kompletaci sborníku a Haně Klímové za organizaci vlastního semináře.

20. listopadu 2006

Július Štuller
koordinátor projektu

Zdeňka Linková
co-editor

Preface

The project *Intelligent Models, Algorithms, Methods and Tools for the Semantic Web realization* of the Program "Information Society" (of the Thematic Program II of the National Program of Research in the Czech Republic) aims at the integration of progressive theoretical disciplines of Computer Science with their practical applications. Its goal is the evaluation and further development of theoretical backgrounds of the semantic web, mainly from the point of view of modern AI methods and methods of computing by nature, as e.g. artificial neural networks, evolutionary algorithms, and data mining, further, for instance, special (description) logics, and the applications and the verification of selected methods of intelligent data and knowledge processing in the field.

The project has started on July 1, 2004 and is planned for 4.5 years. The participants in the project are researchers and students in the following 3 institutions: Faculty of Mathematics and Physics of the Charles University in Prague, Faculty of Informatics of the Masaryk University in Brno, and Institute of Computer Science of the Academy of Sciences of the Czech Republic in Prague.

The workshop dedicated to the results obtained during the first two years of the project was held on October 5 – 7, 2006 in the Hotel Olympia, Zadov, Šumava. The results obtained in each of the participating institutions, in several working groups and also results of the majority of team members were presented there. Based on these presentations papers were completed and are published in these post-workshop proceedings.

We would like to express our thanks to all those who contributed to the publishing of the proceedings, especially to all contributing authors. Our thanks go also naturally to Eva Pospíšilová for a rapid completion of the proceedings and to Hana Klímová for the workshop organization.

November 20, 2006

Július Štuller
Project Coordinator

Zdeňka Linková
Co-editor

Řešitelský tým

Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Vědečtí pracovníci:	Prof. RNDr. Jaroslav Pokorný, CSc. Prof. RNDr. Peter Vojtáš, DrSc. RNDr. Filip Zavoral, Ph.D. RNDr. David Bednárek RNDr. Jakub Yaghob, Ph.D. RNDr. Tomáš Skopal, Ph.D. RNDr. Leo Galamboš, Ph.D.
Doktorandi:	Mgr. Irena Mlýnková Mgr. Kamil Toman Mgr. Martin Nečaský
Odborní pracovníci:	RNDr. David Obdržálek

Fakulta informatiky
Masarykova univerzita

Vědečtí pracovníci:	doc. PhDr. Karel Pala, CSc. Mgr. Jan Pomikálek Mgr. Vlastislav Dohnal, Ph.D. Mgr. Pavel Rychlý, Ph.D. prof. Ing. Pavel Zezula, CSc. Mgr. Aleš Horák, PhD. RNDr. Petr Sojka, PhD. Patrick Hanks, PhD.
Doktorandi:	Mgr. Stanislav Bartoň Mgr. et Bc. Vít Nováček
Odborní pracovníci:	Mgr. Zuzana Nevěřilová Bc. Adam Rambousek Bc. Jana Subiková Viktor Švub Jaroslav Hlávka

Ústav informatiky
Akademie věd České republiky

Vědečtí pracovníci:	Ing. Július Štuller, CSc. – Koordinátor
	Prof. RNDr. Jiří Wiedermann, DrSc.
	Prof. Ing. Zdeněk Strakoš, DrSc.
	Prof. RNDr. Petr Hájek, DrSc
	Prof. RNDr. Peter Vojtáš, DrSc.
	Prof. Ing. Miroslav Tůma, CSc.
	doc. RNDr. Václav Snášel, CSc.
	RNDr. Věra Kůrková, DrSc.
	Mgr. Roman Neruda, CSc.
	Ing. RNDr. Martin Holeňa, CSc.
	Ing. Dušan Húsek, CSc.
	Ing. David Coufal, Ph.D.
	Ing. Petr Cintula, Ph.D.
	RNDr. Milan Daniel, CSc.
Doktorandi:	Ing. Martin Řimnáč
	Ing. Mgr. Radim Nedbal
	RNDr. Petra Kudová
	Ing. Zdeňka Linková
	Ing. Roman Špánek
	Mgr. Stanislav Slušný
	Ing. Zuzana Capeková
	Ing. Pavel Tyl
Odborní pracovníci:	RNDr. Nina Ramešová
	Hana Bílková
	Ludmila Nývltová
	Helena Zelenková
	Hana Klímová

Table of Contents

Mining Citation Graphs Employing an Index for Graph Structured Data

Stanislav Bartoň and Pavel Zezula

Faculty of Informatics, Masaryk University,
Brno, Czech Republic
{xbarton,zezula}@fi.muni.cz

Abstract. In this paper a correlation between ρ-operators and indirect relations in citation analysis is presented. The ρ-operators were defined to explore complex relationships in graph structured data. Various direct and indirect relations identified in citation analysis are used to study the semantics within the citation network. The ρ-index was used to implement the $\rho - path$ search in the citation network and gained results are evaluated and discussed.

1 Introduction

In the context of the Semantic Web, ρ-operators are proposed in [2] as a mean to explore complex relationships [16] between entities. The problem of searching for the complex relationships can be modeled as the process of searching paths in a graph where entities represent vertices and edges the relationships between them. The notion of complex relationships can be also identified in bibliographic digital libraries, where entities could represent publications and the relationship can represent references or citations between them.

As proposed in [16], we recognize two kinds of complex relationships. The first one is represented by *a path* lying between two inspected vertices. The second type of complex relationship is *a connection* between two inspected vertices. This relation is represented by a couple of paths that are both either originated or terminated in the pair of inspected vertices and have one common terminating or initial vertex - a connection. In recent years an approach for indexing generic graph structured data with an emphasis on implementing the search for complex relationships in the indexed graphs has been introduced in [3], [4] and [5]. The designed structure called ρ-index was formally described and experimentally evaluated in [6].

In the field of citation analysis the graph structure was identified when the citation network was firstly introduced by Garfiled in [11] to represent the evolution in science. Garfield believes that there is some semantics behind the citation association between two scientific papers, in other words "association-of-ideas". The citing relation ties the two publications together.

In this paper we would like to present a mapping of searching the complex relationship represented by ρ-operators in graph structured data to citation

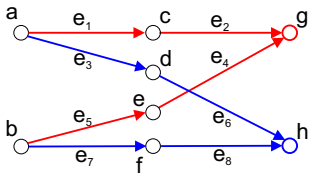


Fig. 1. An example of two connections between vertices a and b. One connection denoted by a pair of paths (e_1e_2, e_5e_4) terminated in vertex g and the another denoted by (e_3e_6, e_7e_8) terminated in vertex h

analysis and therefore usability of generic graph approaches to this problem in the citation analysis.

2 ρ-operators and Citation Analysis

In this section we provide the reader with the formal definitions of ρ-operators mentioned in the previous section. We also introduce the direct citation relationships in the citation analysis and their respective extensions to indirect citation relationships. Finally, we discuss the relationship of the relations studied in citation analysis and the ρ-operators.

2.1 Definition of ρ-operators

The complex relationships among vertices in a graph can be mined using ρ-operators. The first ρ-operator is $\rho - path$ and it is defined as follows:

$$\begin{aligned}\rho - path(x, y) = &\{p = (v_1e_1v_2e_2 \dots e_nv_{n+1}) | v_1 = x \wedge v_{n+1} = y \\ &\wedge e_1, \dots e_n \in E \wedge p \; is \; acyclic\}\end{aligned}$$

The $\rho - path$ operator returns a set of all paths between the two inspected vertices.

The second ρ-operator is $\rho - connection$. We recognize two types of $\rho - connections$. The first one is defined as follows:

$$\begin{aligned}\rho - connectionTo(x, y) = &\{(p_1, p_2) | p_1 = (v_1e_1v_2e_2 \dots e_nv_{n+1}), \\ &p_2 = w_1h_1w_2h_2 \dots h_mw_{m+1}) \wedge \\ &v_1 = x \wedge w_1 = y \wedge v_{n+1} = w_{m+1} \\ &\wedge e_1, \dots e_n, h_1, \dots h_m \in E \wedge p_1, p_2 \; are \; acyclic\}\end{aligned}$$

This type of $\rho - connection$ returns all pairs of paths for a couple of inspected vertices and the inspected vertices are origins for the respective paths from each pair. An illustration of this kind of operator is depicted in Fig. 1.

The other type is very similar, the formal definition is:

$$\begin{aligned}\rho - connectionFrom(x,y) = \{(p_1,p_2) | & p_1 = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}),\\ & p_2 = w_1 h_1 w_2 h_2 \dots h_m w_{m+1}) \wedge \\ & v_1 = w_1 \wedge v_{n+1} = x \wedge w_{m+1} = y \\ & \wedge e_1, \dots e_n, h_1, \dots h_m \in E \wedge p_1, p_2 \; are \; acyclic\}\end{aligned}$$

and the only difference between the former kind of $\rho - connection$ is that the inspected vertices are always terminals of the respective paths from each returned pair.

In certain situations, when the importance of the information decreases with the increasing length of the paths in the graph, all introduced ρ-operators can be limited to certain path length they retrieve from the graph. We denote this as the limit l.

2.2 Direct and Indirect Relationships in Citation Analysis

The first citation relationship identified by Garfield in [11] is the basic citation or reference relationship. One publication cites another. Some other relations were introduced as an alternative to the direct citation relation. First of them is co-citation (Small [14]). It binds together two publications that are directly cited by at least one common paper. Basically, the co-cited pair appears in the reference list of at least one publication. Speaking in terms of graph theory, in the citation network exists at least one direct common predecessor.

The other alternative relation is bibliographic coupling (R. M. Fano [10], M. M. Kessler [13]). It binds two papers together if they both cite at least one common publication. Again speaking in terms of graph theory, in the citation network from the two vertices points an edge to one common vertex. The strength of such relation is then the number of references they have in common.

Another coupling method is called longitudal coupling [15] which connects two publications, older and younger, together taking two steps in the same direction along the citation relation. In another words it puts together two publications between which is an indirect citation relation represented by a path of a length two.

Following the notion of longitudal coupling that extends the direct citation relation to two step indirect citation relation, the work presented in [9] and [8] enhances generally the notions of direct relationships in citation analysis to indirect citation relationships and couplings.

2.3 ρ-operators in Citation Analysis

Considering the indirect relationships in the citation analysis, there is obvious correlation between them and the ρ-operators defined for directed graphs. The $\rho - path$ operator represents all paths forming the indirect citation relationship between two inspected publications – vertices in the citation network.

The indirect co-citation relationship can be implemented by $\rho-connectionTo$ operator since the indirect coupling is defined as a pair of chains of publications originated in the two inspected publications and having one common terminal vertex – a *connection* that represents a common successor, a publication that is indirectly co-cited by the two inspected publications.

Lastly, the bibliographic coupling is a reversed form of the indirect co-citation coupling and can be implemented by the $\rho-connectionFrom$ operator. It returns all pairs of paths that are originated in the common connection and terminated in the two inspected vertices, respectively. The semantic representation of the connection is the publication that indirectly cites the two inspected publications.

The difference between the result of the citing relationships in citation analysis and the ρ-operators for graph structured data is that the citing relationship creates pairs of publications that are in the particular relation. The result of the ρ-operators is a set of paths respectively pairs of paths that represent the particular relationship found. This enables the user with a possibility of further examination of the quality of the relationship found between the pair of investigated vertices – publications. In next section we provide an example of a $\rho-path$ operator applied to a citation network.

The key idea presented in this paper is to apply the indexing techniques developed for general graph structured data on citation network and study the semantics of the retrieved paths and publications. In next section we present a result of a $\rho-path$ operator applied to a graph representing the citation network.

3 $\rho - path$ Results

The set of data representing the citation network is a piece of the CiteSeer [12] database of scientific publications and citations among them. Our data set was created taking one publication and deploying the breadth first search for all weakly accessible publications from our starting one until we got a certain amount of vertices in the built data set. Weak accessibility ignores the orientation of the edge between two vertices. The amount of publications in our data set we set to be 30, 000. The amount of edges acquired among the vertices in the data set was 63, 584. The distribution of the degrees of the vertices in this citation graph is demonstrated in Fig. 2. The x-axis represents the particular vertex degree – respectively the amount of edges initiated, terminated and a total number of both in the particular vertex – and the y-axis then represents an amount of vertices having this degree. The x-axis is drawn using logarithmic scale to make a clearer view of the curve's progress. Also to the values on the y-axis the logarithm was applied to achieve better readability of the demonstrated distribution.

The indegree represents the number of citations of each particular publication. Notice that this distribution follows the power law that states that in the testing citation graph is a small number of vertices that have large indegree and a large number of vertices which's indegree is very small. This exactly conforms with the reality where most of the publications receive a small number of cita-

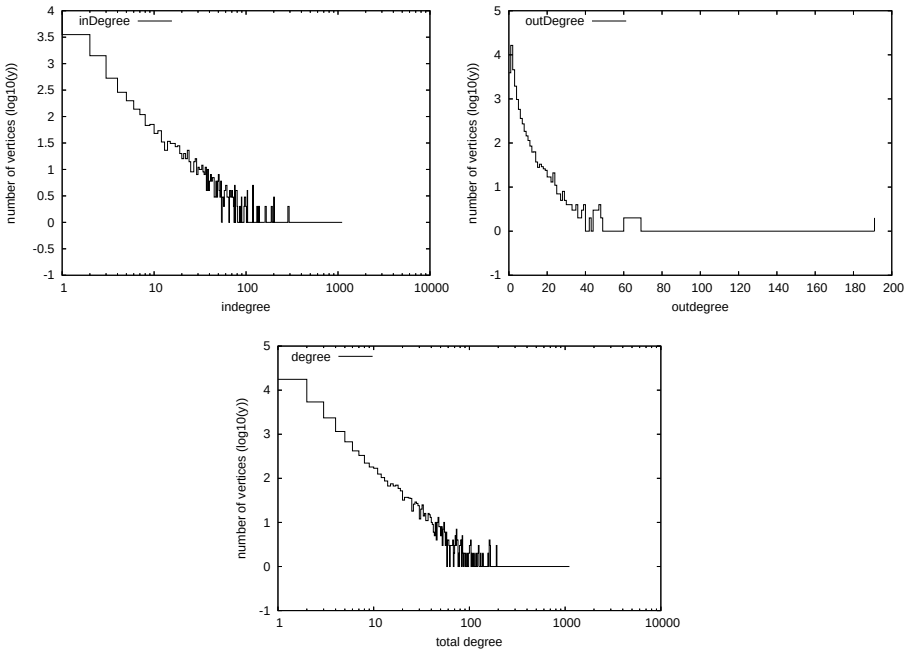
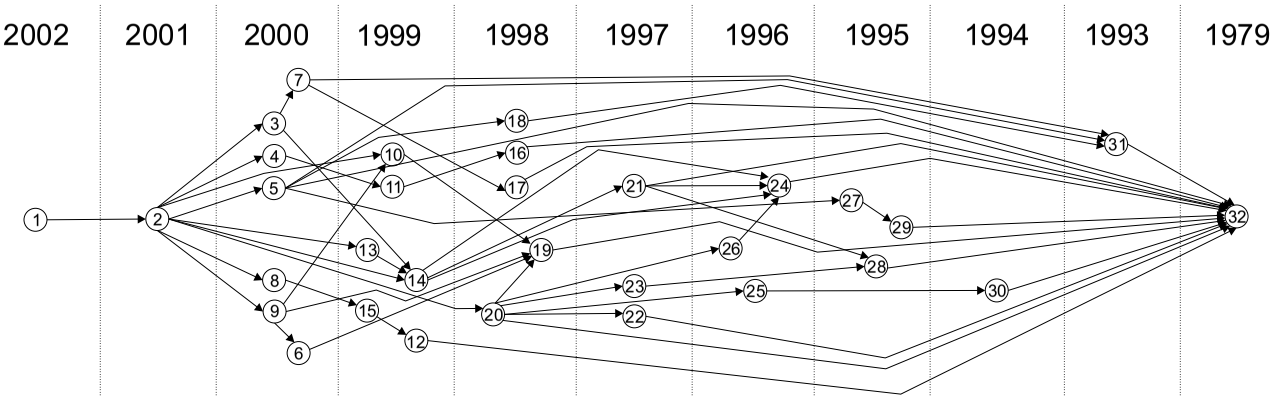


Fig. 2. Vertex degree distribution in the citation graph

tions and was also presented in [1]. To the contrary, the outdegree represents the number of references that the particular publication refers to. This number is not always accurate since CiteSeer does not contain all the references for each publication in its database.

Our testing citation graph was built around Van Rijsbergen's *Information Retrieval* [17] which was identified as a very important publication in the IR field for it's high number of citations by other publications. We followed an idea that if we come across some newer publication that we consider interesting to our research that falls into the same scientific field then there is a high probability that there exists either direct or indirect citation of our core book. If there exists an indirect citation then there is also a possibility that more then one indirect citation paths can be found. In this case we would like to study all paths to certain length lying between our recent – reference – publication and the core book. The vertices on these paths form a set of publications that deserve a further study of their importance by the user.

For our experiment we have chosen [7] as the reference publication. We deployed the ρ-index on our testing citation graph and searched for all paths to length 10. From the nature of the ρ-index that we discussed earlier in this paper we got all the paths to the length of 10 and some longer. Table 1 summarizes the amounts of paths found according to their length and a total number of distinct



1 Predicting Query Performance
2 Relevance-Based Language Models
3 Relating the New Language Models of Information Retrieval to the Traditional Retrieval Models
4 Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding
5 Improving the Effectiveness of Informational Retrieval with Local Context Analysis
6 Statistical Models for Tracking and Detection
7 The Mirror DBMS at TREC-8
8 OCELOT: A system for summarizing web pages
9 Dragon's Tracking and Detection Systems for the TDT2000 Evaluation
10 Topic Tracking in a News Stream
11 Probabilistic Latent Semantic Indexing
12 Automated Text Summarization in SUMMARIST
13 A General Language Model for Information Retrieval
14 A Hidden Markov Model Information Retrieval System
15 Summarizing Text Documents: Sentence Selection and Evaluation Metrics
16 Unsupervised Learning from Dyadic Data
17 MiRRor: Multimedia Query Processing in Extensible Databases
18 Effective Retrieval with Distributed Collections
19 Topic Detection and Tracking Pilot Study Final Report
20 On-line New Event Detection and Tracking
21 Boosting and Rocchio Applied to Text Filtering
22 Study on Retrospective and On-Line Event Detection
23 The TREC-5 Filtering Track
24 Context-Sensitive Learning Methods for Text Categorization
25 Text-Based Information Retrieval Using Exponentiated Gradient Descent
26 Training Algorithms for Linear Text Classifiers
27 Providing Government Information on the Internet: Experiences with THOMAS
28 Evaluating and Optimizing Autonomous Text Classification Systems
29 Corpus-Specific Stemming using Word Form Co-occurrence
30 Optimizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval
31 Using Statistical Testing in the Evaluation of Retrieval Experiments
32 Information Retrieval

Fig. 3. Publication search result visualization

Table 1. A summary of paths found between the reference and the core publication

Path length	**Amount of paths**	Distinct vertices
4	2	5
5	7	13
6	17	32
7	27	51
8	33	58
9	48	59
10	62	60
11	46	61
12	29	61
13	22	61
14	11	61
15	2	61

vertices of all paths up to that length. Fig. 3 demonstrates the network of the paths up to length 6 – as a length of a path we mean the number of vertices in a path. In that figure, the vertices represent publications that are placed on the background of a timeline to make the result more readable. Although, the ρ-index was created to index all the paths up to the length of 10 and as Table 1 shows ρ-index does index also some more, Fig. 3 demonstrates only the paths to the length 6 since it would get very hard to follow when it has contained all the paths got from the ρ-index .

As can be seen in Fig. 3, the result of the search is a network with one source which is our reference publication and one sink which represents our core publication. The result presents a chosen set of publications from the citation graph which relate to the reference publication because the reference publication indirectly cites them and they relate to the field of information retrieval since they indirectly cite our core publication. The resulting publications are ordered according to the year of their publication. Yet another ranking technique could be used to study the relevancy between the reference publication and the publication found using ρ-index but that is beyond the scope of this paper.

The approach proposed used one core publication and as we seen we got for this one a fair amount of publications from the citation graph because the core publication is well known. If we used as a core publication not so well-known publication the system would not be able to retrieve a reasonably big set of publications. For this reason the approach could be improved to carry out the search with not only one core publication but with a set of core publications. Consequently, ρ-index would find all the paths to each particular publication from the core set and put the result together. This improvement brings another interesting issue since the retrieved networks can overlap and that information can be also used for further recommendation process.

4 Concluding Remarks & Future Work

The correlation between the ρ-operators and the relations identified in the citation analysis was presented and ρ-index , the indexing structure that is designed to ease the search for complex relationships in the graph structured data was used to implement the search for indirect citation relation. The results gained by this search were evaluated and discussed. The main contribution of applying ρ-operators to implement various relations in citation analysis we see is the possibility of a further study of the paths that form the particular relation.

Our future work has two main aims. Firstly, to further study the semantics of the retrieved results by the ρ-operators from the citation network. Secondly, to study also the results of the $\rho - connection$ operator in the citation analysis.

References

1. An Y., Janssen J. and Milios E. E.: Characterizing and mining the citation graph of the computer science literature. In Knowledge Information Systems, New York, NY, USA, Springer-Verlag New York, Inc. **6** (2004) 664–678
2. Anyanwu K. and Sheth A.: The ρ-operator: Enabling querying for semantic associations on the semantic web. In Proceedings of the twelfth international conference on World Wide Web, ACM Press (2003) 690–699
3. Bartoň S.: Designing indexing structure for discovering relationships in RDF graphs. In Proceedings of the Dateso 2004 Annual International Workshop on DAtabases, TExts, Specifications and Objects (2004) 1–11
4. Bartoň S.: Indexing structure for discovering relationships in RDF graph recursively applying tree transformation. In Proceedings of the Semantic Web Workshop at 27th Annual International ACM SIGIR Conference (2004) 58–68
5. Bartoň S. and Zezula P.: Rho-index - an index for graph structured data. In 8th International Workshop of the DELOS Network of Excellence on Digital Libraries (2005) 57–64
6. Bartoň S. and Zezula P.: rhoIndex – designing and evaluating an indexing structure for graph structured data. Technical Report FIMU-RS-2006-07, Faculty of Informatics, Masaryk University (2006)
7. Cronen-Townsend S., Zhou Y. and Croft W. B.: Predicting query performance. In Proceedings of the 25th Annual International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2002) (August 2002) 299–306
8. Egghe L. and Rousseau R.: Co-citation, bibliographic coupling and a characterization of lattice citation networks. In Scientometrics **55** (2002) 349–361
9. Fang Y. and Rousseau R.: Lattices in citation networks: An investigation into the structure of citation graphs. In Scientometrics **50** (2001) 273–287
10. Fano R. M.: Information theory and the retrieval of recorded information. In Documentation in action (1956) 238–244
11. Garfield E.: Citation indexes for science: A new dimension in documentation through association of ideas. In Science **122** (1955) 108–111
12. Giles C. L., Bollacker K. and Lawrence S.: CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, Digital Libraries 98 – The Third ACM Conference on Digital Libraries, Pittsburgh, PA, ACM Press, June 23–26 (1998) 89–98

13. Kessler M. M.: Bibliographic coupling between scientific papers. In American Documentaion **14** (1963) 10–15
14. Small H.: Co-citation in the scientific literature: A new measure of the relationship between two documents. In Journal of the American Society of Information Science **24** (1973) 265–269
15. Small H.: Navigating the citation network. In Proceedings of the 58th Annual Meeting of the American Society for Information Science: Forging new partnership in information **50** (1999) 118–126
16. Thacker S., Sheth A. and Patel S.: Complex relationships for the semantic web. In D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, editors, *Spinning the Semantic Web*. MIT Press (2002)
17. Van Rijsbergen C. J.: Information Retrieval, 2nd edition. Dept. of Computer Science, University of Glasgow (1979)

Infrastruktura pro dotazování nad semantickými daty*

Jiří Dokulil, Jakub Yaghob a Filip Zavoral

Katedra softwarového inženýrství, MFF UK Praha
{Jiri.Dokulil,Jakub.Yaghob,Filip.Zavoral}@mff.cuni.cz

Abstrakt Idea sémantického webu je široce diskutována mezi odbornou veřejností již mnoho let. Přestože je vyvinuta řada technologií, jazyků, prostředků a dokonce i softwarových nástrojů, málokdo někdy nějaký reálný sémantický web viděl. Za jeden z hlavních důvodů tohoto stavu považujeme neexistenci potřebné infrastruktury pro provoz sémantického webu. V našem článku popisujeme návrh takové infrastruktury, která je založena na využití a rozšíření technologie datového stohu a nástrojích pro něj vyvinutých a jejich kombinaci s webovými vyhledávači a dalšími nástroji a prostředky.

1 Úvod – současný stav sémantického webu

Kdyby sémantický web byl alespoň z poloviny tak dobrý, jak se snaží proklamovat jeho vizionáři, jistě by se ho chopila komerce a každý by se s ním denně setkával, podobně jako dnes s emailem nebo webovými stránkami. Realita současnosti je však zcela jiná – asi jen málokdo někdy viděl nebo používal něco, co by se dalo nazvat sémantickým webem. Jedním z hlavních důvodů je neexistence nějaké jednotné infrastruktury, na které by bylo možné sémantický web efektivně provozovat.

Tento problém lze nejlépe demonstrovat srovnáním s 'obyčejným' webem. Zde je infrastruktura jasná a dlouhodobě stabilně používaná. Webové servery (nebo farmy serverů) mají na svých discích uložené stránky a zdrojové texty webových aplikací, server data poskytuje typicky protokolem http nebo https klientskému prohlížeči. Data jsou nejčastěji ve formáltu html doplněném případně o další aktivní prvky. Tato data prohlížeč zobrazí nebo interpretuje a umožní uživateli další navigaci.

Sémantický web takovou 'standardní' infrastrukturu nemá. Jsou sice vyvinuty a relativně stabilizovány různé popisné prostředky pro zaznamenávání ontologií (RDF, RDFS, OWL), navrženy a pilotně implementovány specializované dotazovací jazyky (SPARQL [4], RQL [8], SeRQL [9] nebo RDQL [10]), avšak kde a jak jsou data a metadata ukládána (RDF a RDFS jsou sice vhodné prostředky pro uchavávání relativně malých objemů dat, avšak pro velmi velké

* Tato práce byla částečně podporována projektem 1ET100300419 Programu Informační společnost Tématického programu II Národního programu výzkumu České republiky.

datové objemy samy o sobě příliš vhodné nejsou), jak se plní daty, jak jsou data vázána na metadata, čím se na ně lze dotazovat, kdo a jak zpracovává odpovědi, jaké protokoly se používají pro vzájemnou komunikaci – to jsou všechno technické detaily, kterým doposud byla věnována pouze marginální pozornost, a to zejména vzájemné komplexní provázanosti jednotlivých otázek. Nedořešenost těchto technických otázek je jednou z příčin reálné neexistence sémantického webu.

Další kapitoly tohoto článku jsou organizovány následujícím způsobem: v kap. 2 je popsána použitelnost stohových systémů pro datové úložiště sémantického webu, kap. 3 popisuje způsoby transformace mezi RDF daty a stohem, v kap. 4 jsou rozebrány jednotlivé moduly infrastruktury, kap. 5 se zabývá vyhodnocováním SPARQL dotazů, v kap. 6 jsou analyzovány možné optimalizace dotazování pomocí indexů a konečně kap. 7 shrnuje současný stav a nastiňuje další vývoj.

2 Stohové systémy a jejich vztah k sémantickému webu

2.1 Stoh

V rámci vývoje konkrétního informačního systému jsme vyvinuli datovou strukturu pro centrální úložiště dat odpovídající požadavkům na systém kladeným – stoh [1,2].

Základní ideou stohových systémů je vertikalizace dat, tj. nevyužívá se tradiční horizontální pojetí databázové tabulky, kdy jedna řádka představuje nějakou množinu spolu souvisejících atributů nějaké entity. Místo toho je každý atribut 'tradiční' řádky představován jedním řádkem datového stohu a odpovídající si atributy jsou pak spojeny identifikací entity, které tyto atributy náleží. Celé datové schéma všech zúčastněných aplikací je nahrazeno dvěma základními tabulkami - hodnotami atributů a strukturou entit.

Během práce na grantu Sémantického webu jsme ukázali [3], že tato datová struktura je ve skutečnosti velmi dobře použitelná i pro sémantický web.

Původní základní požadavky na použití stohu jakožto centrálního datového úložiště pro integraci byly:

1. Zachování většího množství stávajících provozních aplikací
2. Sjednocení dat z různých pracovišť
3. Aktivní distribuce změn údajů
4. Relativně snadná možnost přidání dalších sbíraných a skladovaných informací
5. Plná informace o změnách dat na časové ose

Tyto požadavky velmi dobře odpovídají i požadavkům na datové úložiště sémantického webu:

1. Zachování zdrojů zdroje dat jsou rozmístěny po celém webu a nejsou pod správou nikoho konkrétního, tudíž nelze ovlivnit jejich datové schéma.

2. Data uchovávaná v jedné instanci stohu odpovídají jedné ontologii. Ta však nemusí odpovídat ontologiím jiných zdrojů. Při importu dat z jiných zdrojů se tato data převádějí na námi zvolenou a udržovanou ontologii.
3. Stoh je schopen zajistit export dat včetně aktivního exportu (push).
4. Lze poměrně snadno měnit strukturu dat ve stohu, neboť struktura dat není určena přímo databázovým schématem, ale obsahem metatabulek uchovávajících strukturu dat. Změna struktury dat odpovídá změnám v udržované ontologii.
5. Stoh je navíc schopen udržet informace o časovém průběhu dat, takže jsme schopni zjistit informace o stavu světa vzhledem k nějakému časovému bodu.

2.2 Uložení dat ve stohu a vztah k RDF

Data jsou ve stohu aktuálně uložena v jedné tabulce, která obsahuje pro každou hodnotu atributu entity v daném časovém úseku jednu řádku. V každé této řádce jsou uloženy následující položky: číslo entity, typ atributu a hodnota atributu, zdroj dat, validitu a relevanci. První tři položky velmi dobře modelují RDF model dat, kde entita představuje subjekt, typ atributu je predikát a hodnota atributu je objekt. Zbývající atributy jsou vhodné pro implementaci reifikací.

Tím, že datová tabulka stohu odpovídá RDF, jsme získali v rámci výše uvedeného projektu velkou databázi (řádově desítky miliónů záznamů) reálných RDF dat. Drobnou nevýhodou je fakt, že některá data jsou privátní a nemohou být zveřejněna, což se dá napravit vhodným anonymizováním těchto dat.

2.3 Reifikace

Atribut zdroj dat zmiňovaný v předešlé podkapitole určuje odkud data pocházejí, relevance reprezentuje dohad důvěryhodnosti zdroje dat a dat samotných a validita určuje časový rozsah platnosti dat. V kontextu sémantického webu lze tyto údaje o každé datové položce považovat za reifikace příslušné RDF trojice. Explicitním modelováním těchto vztahů pomocí čistého RDF bychom dostali několikanásobně větší data a dotazování nad takovými daty by bylo znatelně pomalejší. Jednoduchým využitím základních vlastností stohu můžeme tyto vztahy velmi jednoduše a přitom efektivně použít.

2.4 Kontextová ontologie a mapování ontologií

Ontologie uchovávaná v metatabulkách stohu je ontologií kontextovou, tj. popisuje pouze data uložená ve stohu. Data časem přibývají a mění se i ontologie typicky zvětšováním (přidáváním dalších oblastí), někdy i změnou. Metatabulky stohu pak musejí zvládnout tyto změny ontologií beze změny obsahu datové části stohu.

Velkým problémem, který brání celosvětovému rozšíření sémantického webu, je mapování různých ontologií na sebe. Různí autoři se snaží tento problém různými prostředky řešit, bohužel v současné době problém není uspokojivě vyřešen.

Pokud použijeme kontextovou ontologii, pak musíme i zajistit mapování s jinými ontologiemi při přijetí nových dat a metadat. Zde se nabízejí tři různé metody:

- Jednou z možných metod je metoda Rosetské desky, tj. existuje nějaký spolehlivý, dobře známý zdroj, který zajišťuje alespoň částečný překlad mezi různými ontologiemi.
- Druhá metoda je mapování map na sebe. Mějme mapy nějakého území z různých časových období, takže zobrazují trochu jinou situaci. Pokud se nám podaří na mapách najít několik málo styčných bodů (např. význačná města), pak už jsme schopni zbytek map na sebe také namapovat.
- Poslední metodou je domluva dvou lidí, kteří mluví jiným jazykem. S využitím neverbální komunikace si vytvoří základní slovník, pomocí kterého pak vytváří další bohatší slovník, čímž mapují postupně jazyky na sebe.

První dvě metody vyžadují nějaký lidský zásah nalezení nebo vybudování Rosetské desky, v druhém případě typicky lidská obsluha najde styčné body na mapě. Třetí metodu lze nejspíše využít pro čistě strojové mapování. Bude nutné navrhnout nějaký protokol, který nahradí neverbální prostředky lidské komunikace nějakými jinými prostředky dostupnými ve světě počítačů.

3 Transformace

Při zpracování RDF dat jsou zapotřebí transformace mezi stohovou formou a čistou RDF formou (trojicemi).

Základní myšlenka celé transformace je, že pokud uděláme projekci tabulky STOH na sloupce obsahující identifikátor entity, atribut a hodnotu atributu, dostaneme množinu trojic, které jsou velmi podobné RDF trojicím.

3.1 Identifikátor entity

Všechny entity ve Stohu mají přidělen unikátní osmnáctimístný identifikátor identifikátor entity. V RDF grafu potřebujeme vytvořit uzly, kterými budeme reprezentovat entity ze Stohu. K těmto uzlům potřebujeme jednoznačné identifikátory. Identifikátor entity je pro tento účel ideální. Můžeme jej použít buď jako součást URI pro uzel a nebo jako jméno anonymního uzlu v serializaci grafu. V následujícím textu popisujeme druhou možnost, protože anonymní uzly jsou významný koncept v RDF a proto chceme, aby je výsledná data obsahovala ve velkém množství. V případě, že by byl potřeba přímý přístup k uzlům, je změna celé transformace na první možnost triviální.

3.2 Metadata

Zpracování dat ve Stohu je řízeno metadaty, která jsou uložena v tabulkách. Aby bylo možné provést transformaci dat do RDF, je nutné napřed transformovat

alespoň část metadat. Nejdůležitější část metadat jsou v tomto případě atributy, protože poslouží jako predikáty (hrany grafu) v RDF. Nejjednodušší reprezentace jednoho atributu vypadá takto (notace TURTLE [3]).

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix mt:  <http://example.org/stoh/metadata/> .

mt:person__name rdf:type rdf:Property .
```

Tímto definujeme http://example.org/stoh/metadata/person__name jako atribut. Jeho předloha ve stohu byl atribut "name" patřící k typu entity "person". I tuto informaci můžeme uložit v RDF.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix mt:  <http://example.org/stoh/metadata/> .

mt:person rdf:type rdfs:Class .
mt:person__name rdf:type rdf:Property .
mt:person__name rdfs:domain mt:person .
```

Jako alternativní název atributu by bylo možné použít pouze název atributu ve Stohu a nekombinovat jej se jménem typu entity. Pak by bylo možné pokládat dotazy jako třeba: "Chci jméno všeho, co jméno má." Na druhou stranu by se zkomplikovala typová kontrola. Z tohoto důvodu jsme zvolili delší jména. S těmito RDF daty už můžeme přiřadit entitě její typ.

```
_:568421369754123695 rdf:type mt:person .
```

Subjektem trojice je anonymní uzel s identifikátorem shodným s identifikátorem entity ve Stohu.

3.3 Datové typy

V následující tabulce jsou uvedeny datové typy používané ve Stohu a jajich ekvivalenty v RDF. Typ BLOB byl vynechán. Transformace binárních dat by sice byla možná pomocí například Base64 kódování, ale taková data by nebyla příliš užitečná a navíc by zbytečně narůstala velikost výsledku transformace.

string	xsd:string
number	xsd:decimal
timestamp	xsd:dateTime
entity reference	odkaz na anonymní uzel

S těmito typy můžeme rozšířit transformaci metadat:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix mt:  <http://example.org/stoh/metadata/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

mt:person rdf:type rdfs:Class .
mt:person__name rdf:type rdf:Property .
mt:person__name rdfs:domain mt:person .
mt:person__name rdfs:range xsd:string .
```

Odkazy na entity ve Stohu jsou typované. Jeden atribut lze použít pouze jako odkaz na entity jednoho, předem zvoleného, typu. To v RDF vyjádříme specifikací třídy odpovídající odkazovanému typu entit jako oboru hodnot predikátu reprezentujícího odkaz.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix mt:  <http://example.org/stoh/metadata/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

mt:person rdf:type rdfs:Class .
mt:address rdf:type rdfs:Class .
mt:person__address rdf:type rdf:Property .
mt:person__address rdfs:domain mt:person .
mt:person__address rdfs:range mt:address .
```

3.4 Transformace dat

Teď je již možné transformovat data za použití transformovaných metadat. V úvodu zmíněnou projekcí tabulky STOH na sloupce obsahující identifikátor entity, atribut a hodnotu atributu získáme řádky a z každého z nich vytvoříme jednu RDF trojici. Příklad jedné transformované entity by mohl vypadat takto:

```
_:568421369754123695 mt:person__name "John Smith" .
_:568421369754123695 mt:person__date_of_birth
          "1980-08-14T00:00:00"^^xsd:dateTime .
_:568421369754123695 mt:person__height "1.82"^^xsd:decimal .
_:568421369754123695 mt:person__father _:684258941535789524 .
```

Poslední trojice je odkaz na jinou entitu (cizí klíč).

3.5 Vícejazyčné atributy

Ačkoliv popsaná transformace je dostatečně obecná, aby bylo možné zpracovat libovolná data uložená ve Stohu, existuje jeden případ, který si zaslouží zvláštní pozornost a zacházení. Při praktickém nasazení Stohu se ukázalo, že je někdy nezbytné vyjádřit jeden řetězcový atribut ve více jazycích nebo více tvarech v

rámci jednoho jazyka. Za použití prostředků Stohu bylo nutné vytvořit dva typy entit, což velmi znesnadnilo použití této funkce. RDF nabízí jednodušší prostředek, jak dosáhnout stejného výsledku. K literálu je možné přidat jazykovou značku, která vyjadřuje v jakém jazyce je daná hodnota. Tyto značky jsou definovány standardem RFC 3066, který je dostatečně flexibilní a dovoluje nejen uvést ve kterém jazyce, ale i vlastní třídy, například číslo a pád. Dva příklady jejich použití:

```
_:469751359754692454 rdf:type mt:department .
_:469751359754692454 mt:department__name
   "Katedra softwarového inženýrství"@cs .
_:469751359754692454 mt:department__name
 "Department of Software Engineering"@en .
_:954783125769542934 rdf:type mt:place .
_:954783125769542934 mt:place__name
     "Praha"@cs-CZ-singular-nominative .
_:954783125769542934 mt:place__name
     "v Praze"@cs-CZ-singular-locative .
```

3.6 Reifikace

Stohové reifikace je možné vyjádřit v RDF. Pro každý řádek stohu založíme jeden anonymní uzel, který odpovídá reifikovanému tvrzení.

```
_:568421369754123695 mt:person__name "John Smith" .
_:r65413 rdf:type rdf:Statement .
_:r65413 rdf:subject _:568421369754123695 .
_:r65413 rdf:predicate mt:person__name .
_:r65413 rdf:object "John Smith" .
_:r65413 mt:valid_from "20050703T15:21:49" .
_:r65413 mt:valid_to "20050821T09:35:12" .
```

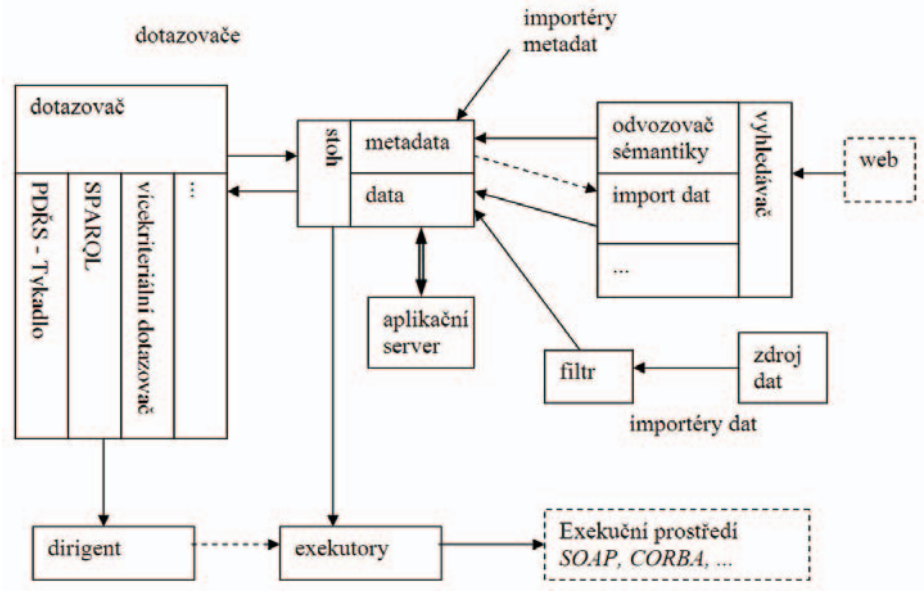
4 Infrastruktura pro provoz sémantického webu

Základem navrhované infrastruktury pro sémantický web je stoh, kde jsou uložena všechna metadata a data na ně vázaná. Stoh poskytuje čtyři základní druhy rozhraní – pro import dat, import a aktualizaci metadat, dotazování a exekutory.

4.1 Importéry

Rozhraní pro import dat umožňuje libovolnému modulu doplňovat do stohu data. Typickým představitelem importérů jsou filtry, které data z libovolného zdroje (databáze, XML, web, ...) konvertují do fyzické podoby zpracovatelné stohem a do logického tvaru odpovídajícímu metadatům, na která jsou tato data navázána.

Důležitou součástí importérů je schopnost detekovat již existující data a tato aktualizovat. K tomu slouží rozhraní pro unifikce, kde na základě určujících a



Obrázek 1. Infrastruktura pro provoz sémantického webu

relevantních atributů lze pomocí unifikačních algoritmů na sebe vázat existující a nově importovaná data.

Zvláštní význam mezi importéry mají vyhledávače - ty spojují sémantický web s webem. Pro naše účely jsme použili systém Egothor [6], který svojí modulární koncepcí umožňuje komfortně doplnit příslušné moduly pro spolupráci se stohem. V původní podobě Egothor na základě stažených dat vytváří záznam webu v inverzní vektorové podobě. Doplněním extrakčních modulů umožní vybraná data ukládat do stohu. Vzhledem k obrovskému množství těchto dat je konverze zprostředkovávána pomocí specializovaného kompresního modulu [12].

Pouhý přísun samotných dat by v dlouhodobějším provozu sémantického webu nedostačoval - svět sémantického webu je velmi dynamický a jakákoliv struktura dat již v okamžiku jejího zaznamenání může být zastaralá. Proto důležitou úlohu mají importéry metadat. Jejich cílem je aktualizovat metadata tak, aby co nejvěrněji odpovídala aktuálnímu stavu. Podobně jako importéry dat mohou být importéry metadat libovolné komponenty s libovolnou logikou, jedinou podmínkou je implementace vyhovující definovanému rozhraní.

Importéry metadat lze rozdělit na manuální a automatické. Typickým představitelem manuálních importérů metadat jsou filtry exportů různých datových modelů, XML Schemat apod., které umožní přímý import z takto popsaných zdrojů dat. Vystavět sémantický web pouze nad těmito relativně pevně strukturovanými daty by však bylo příliš omezující, současný web obsahuje o mnoho řádů více dat nestrukturovaných.

Připravený framework pro automatické importéry metadat umožňuje vytvářet samostatné moduly, které na základě různých algoritmů založených heuristických, statistických a pravděpodobnostních metodách a v neposlední řadě také na umělé inteligenci mohou automaticky generovat metadata ze zpracovávaných dat. Jedním z příkladů automatických importérů je vyhledávač Egothor doplněný o modul pro automatické odvozování sémantiky na základě stažených dat.

4.2 Dotazovače

Zřejmě nejdůležitějším účelem uložených dat a metadat sémantického webu je možnost dotazování. Na rozdíl od tradičních relačních databází s pevným datovým schématem a standardizovanými dotazovacími nástroji je dotazování nad daty sémantického webu zatím ve stádiu návrhů a pilotních implementací. Tomu odpovídá i navrhovaná infrastruktura. Základem je opět definované rozhraní, jehož prostřednictvím lze na stoh klást dotazy a získávat odpovědi. Tento popis je záměrně velmi široký, neboť obě jeho části (kladení dotazů a získávání odpovědí) mohou nabývat mnoha podob.

Vlastní dotazování je komplikováno tím, že uživatel typicky nezná strukturu dat, která je navíc rozsáhlá a v čase dynamická. Proto jedním z modulů dotazovače je prohlížeč dat řízený sémantikou (PDŘS) pracovně nazvaný Tykadlo, který umožňuje vyhledávat a prohlížet metadata, filtrovat data vztažená k těmto metadatům, a přes datové vazby prohlížet další data na tato vázaná.

Modul SPARQL [3,4] přeloží dotaz zapsaný v jazyce SPARQL do SQL a nechá ho vyhodnotit databázový stroj. Tento způsob dotazování je vhodný pro jednodušší dotazy, složitější dotazy s velkým počtem spojení jsou zatím výkonnostně nedostačující.

Dalším modulem je vícekriteriální dotazovač [5], který umožňuje specifikovat několik vyhledávacích kritérií, přičemž výsledek je nějakou obecnou (monotónní) funkcí jednotlivých kritérií [7]. Uživatel má vlastní preference pro jednotlivé atributy i pro jejich celkovou integraci. Úlohou modulu je najít nejlepší odpověď, případně k-nejlepších odpovědí. V dynamické verzi může být modul rozšířen o použití výsledků předešlých dotazů, a to jak vlastních tak i jiných uživatelů.

Ve fázi vizí je doplnění dotazovače o moduly umožňující formulaci dotazů v přirozeném jazyce a zapojení lingvistických metod, případně použití pokročilých metod umělé inteligence a dolování dat ve stylu 'ukažte, data, co je na vás zajímavého'. Přestože přiblížení těchto vizí je hudbou vzdálenější budoucnosti, infrastruktura je na tyto moduly připravena.

4.3 Exekutory

Doposud jen volně zmiňovaný 'výsledek dotazu' lze interpretovat mnoha různými způsoby. Od relačně orientovaného data-setu přes seznam odkazů známý z webových vyhledávačů nebo seznam entit doplněný jejich vazbami až po aplikační funkčnost typu zavolání vhodné služby SOA.

Tradiční způsob reprezentace výsledků dotazování je pevně vázaný na použitý dotazovač. Tykadlo zobrazuje vzájemně propojené html stránky se zobrazenými vyhledanými daty a jejich vazbami, vyhledávač zobrazuje webové odkazy s případným podrobnějším popisem, SPARQL vrací řádky n-tic vyhledaných atributů.

Technika exekutorů zavádí do infrastruktury procesní modely. Úkolem exekutoru je provést sémantickou akci, tj. interakci dat získaných dotazovačem s ostatním světem, a to nejen světem sémantického webu. Tyto atomické exekutory lze složit a vytvořit exekutory složené. Orchestraci, tj vzájemné propojení exekutorů za účelem dosažení požadované komplexnější funkčnosti, provádí modul dirigent.

Ideu exekutorů lze ilustrovat následujícím příkladem. Onemocní-li někdo, potřebuje lék. Dotazovač nalezne nejbližší lékarnu nabízející vhodný lék. Jeden exekutor je zodpovědný za nákup léku zatímco druhý zařídí jeho dodávku až domů. Modul dirigent orchestruje tyto exekutory tak, aby byly vzájemně synchronizované, aby vzájemně spolupracovaly a předávaly si relevantní data.

5 Vyhodnocování SPARQL dotazů

Jeden z dotazovačů, který je již plně implementován, je dotazovač SPARQL.

V této kapitole ukážeme několik způsobů dotazování spolu s analýzou rychlosti jejich vyhodnocení. Pro každý dotaz jsme otestovali čtyři metody uložení a indexace trojic. Naším cílem bylo nalezení nejvhodnějšího způsobu indexace pro výběr trojic se zadaným predikátem.

5.1 Testovací prostředí

Databáze běžela na stroji se dvěma procesory XEON 3.06GHz, 6GB RAM a SCSI diskovým polem se čtrnácti 144GB 10k RPM disky pro uložení dat a indexů. Pro RDF databázi bylo vyhrazeno 600MB RAM.

5.2 Měření dotazů

K dotazům uvádíme i tabulky s výsledky experimentálního měření rychlosti. Při měření jsme postupovali následujícím způsobem.

1. SPARQL dotaz převedeme do SQL.
2. SQL dotaz vložíme do měřicího systému. Ten automaticky provede kroky 3 až 8.
3. Proběhne restart databáze (vyprázdnění cache pamětí)
4. Pauza 10 vteřin, aby databáze dokončila start.
5. Spuštění SQL dotazu.
6. Změření času do vrácení první řádky.
7. Změření celkového času dotazu a celkového počtu vrácených řádek.
8. Druhé měření bez restartu databáze (body 4 až 7).

Výsledky ukázaly, že vyprázdnění cache v databázi a opakování měření má smysl, protože časy prvního a druhého pokusu se výrazně liší. Další iterace však již zrychlení nepřinášejí.

5.3 Jednoduché dotazy

Napřed se budeme zabývat dotazy, které jsou zaměřeny na jednotlivé konstrukce jazyka SPARQL.

Následující dotaz vrací seznam všech tříd v databázi.

```
select ?trida
where { ?trida a rdfs:Class }
```

	První spuštění		Opakované spuštění	
	celkem	první řádek	celkem	první řádek
základní	91ms	90ms	20ms	19ms
B-strom	63ms	63ms	19ms	19ms
paralelní	38ms	37ms	21ms	20ms
bitmapa	73ms	72ms	21ms	20ms
	Počet řádků: 226			

Ukážeme si ještě dotaz, který vrací velký seznam, pro jehož získání je třeba spojenit velké množství trojic.

```
select ?jmeno ?prijmeni
where { ?osoba mt:ot_osoba__jmeno ?jmeno .
  ?osoba mt:ot_osoba__prijmeni ?prijmeni }
```

	První spuštění		Opakované spuštění	
	celkem	první řádek	celkem	první řádek
základní	20s	2920ms	11s	914ms
B-strom	14s	878ms	13s	292ms
paralelní	13s	369ms	12s	315ms
bitmapa	19s	4634ms	11s	902ms
	Počet řádků: 91166			

Jako poslední ukážeme dotaz, kterým hledáme všechny objekty, jejichž hodnota je číslo 1.

```
select ?s ?p
where { ?s ?p 1 }
```

	První spuštění		Opakované spuštění	
	celkem	první řádek	celkem	první řádek
základní	101s	274ms	83s	134ms
B-strom	87s	159ms	85s	196ms
paralelní	87s	177ms	84s	181ms
bitmapa	81s	234ms	80s	160ms
	Počet řádků: 1987905			

Celkově lze říct, že dosahované časy odpovídají očekávání.

5.4 Složitější dotazy

V této části se budeme zabývat dotazy, které mohou kombinovat i více základních konstrukcí SPARQL a měly by spíše odpovídat realným dotazům, které by mohl uživatel pokládat nad uloženými daty.

Napřed uvažme dotaz, který na základě několika zadaných hodnot testuje, jestli taková osoba je nebo není v databázi.

```
select ?osoba
where {
  ?osoba mt:ot_osoba__jmeno 'Josef' .
  ?osoba mt:ot_osoba__prijmeni 'Dvořák' .
  ?osoba mt:ot_osoba__datum_narozeni
  '1968-04-06T00:00:00' .
  ?osoba mt:ot_osoba__rok_maturity '1987' .
  ?osoba mt:ot_osoba__trvaly_pobyt_v_cr 'A' .
  ?osoba mt:ot_osoba__pohlavi ?pohlavi .
  ?pohlavi mt:cht_ciselnik_plochy__kod '1'
}
```

V relační databázi s rozumně definovanými indexy by byl dotaz vyhodnocen téměř okamžitě. Protože některé z omezení (například na datum narození) mají velmi dobrou selektivitu (vyberou obvykle 0 až 1 trojici), bylo by možné vyhodnotit i tento dotaz v RDF velmi rychle díky tomu, že toto omezení nechá velmi málo možných ohodnocení proměnné osoba a ověřit existenci požadovaných hodnot pro tento subjekt lze díky indexu nad subjektem snadno.

Reálná měření však dopadla špatně. I v případě, že taková osoba v databázi neexistuje, trvá vyhodnocení dlouho.

	První spuštění		Opakované spuštění	
	celkem	první řádek	celkem	první řádek
základní	8129ms		1154ms	
B-strom	826ms		117ms	
paralelní	4805ms		118ms	
bitmapa	5225ms		1147ms	
	Počet řádků: 0			

Následující dotaz vrací dvojice rodné číslo a email pro osoby, které mají oba údaje zadány.

```
select ?rc ?email
where {
  ?kontakt mt:ot_kontakt__id_osoba ?osoba .
  ?kontakt mt:ot_kontakt__druh_kontaktu ?dkon .
  ?kontakt mt:ot_kontakt__email ?email .
  ?dkon mt:cht_ciselnik_plochy__kod "2" .
  ?ident mt:ot_identifikace__id_osoba ?osoba .
  ?ident mt:ot_identifikace__druh ?rckod .
  ?rckod mt:cht_ciselnik_plochy__kod "2" .
  ?ident mt:ot_identifikace__identifikace ?rc
}
```

Dosahované rychlosti ukazuje následující tabulka:

	První spuštění		Opakované spuštění	
	celkem	první řádek	celkem	první řádek
základní	54s	44s	54s	44s
B-strom	717s	716s	745s	743s
paralelní	16s	11s	14s	10s
bitmapa	275s	101s	286s	105s
	Počet řádků: 7861			

U tohoto dotazu se ukázala nevýhoda námi zvoleného přístupu, kdy veškerá data jsou uložena v jediné tabulce (to je však vynuceno obecností SPARQL dotazů) a optimalizace je ponechána na optimalizátoru relační databáze. Tento optimalizátor vychází ze statistik, které si o tabulkách udržuje.

Již jsme však zmínili, že naše data (ačkoliv mnoho pozorování by bylo možné zobecnit na většinu RDF dat) vykazují některé nezvyklé vlastnosti, co se týče celkových statistik. Důležité je například pozorování, že trojice s konkrétním predikátem představují pouze zlomek databáze a přesto jde až o stovky tisíc záznamů.

Tato vlastnost je pravděpodobná příčina toho, že optimalizátor dělá špatné odhady velikosti mezivýsledků při vyhodnocení dotazu. Po spojení několika tabulek s trojicemi dojde k odhadu, že výsledek spojení bude obsahovat pouze jeden řádek. Na základě tohoto předpokladu zvolí metodu spojení. Při skutečném vyhodnocení dotazu se často stane, že oproti jednomu předpokládanému řádku se do mezivýsledku dostanou stovky tisíc řádků, na což zvolená metoda spojení není vhodná.

Tím lze vysvětlit, proč vyhodnocení dotazu trvá nepřiměřeně dlouho.

Je také vidět, že mezi rychlostmi při různých metodách indexace jsou velké rozdíly. Obzváště zajímavý je velký rozdíl mezi paralelní a neparalelní verzí B-stromu. Mnohonásobný nárůst rychlosti v paralelní verzi nemůže být způsoben jen větším dostupným výpočetním výkonem, protože databázový stroj měl

k dispozici pouze dva procesory. Pravděpodobná příčina je, že různé plány vyhodnocení byly kvůli špatným odhadům optimalizátoru ohodnoceny podobnou celkovou cenou. Proto mohla drobná změna parametrů na vstupu optimalizátoru (například míra paralelizace) vést k výběru jiného plánu s podobnou odhadovanou cenou, který se při skutečném spuštění dotazu ukázal mnohem vhodnější.

Navíc dosahované časy se při opakovaném měření mění. Pravděpodobný zdroj tohoto chování je fakt, že Oracle při optimalizaci využívá i znalostí o předchozích vyhodnoceních stejného dotazu. Protože se všechny navržené plány vyhodnocení dotazu ukazují jako velmi neoptimální, snaží se optimalizátor (neúspěšně) o jejich zlepšení drobnou změnou plánu.

K řešení tohoto problému je možné přistupovat ze dvou směrů. Těmito směry se zabývají následující dvě kapitoly.

6 RDF indexy

Problémy s optimalizací jsou částečné způsobeny velkým množstvím spojení, které je potřeba pro vyhodnocení jednoho dotazu. Snížením tohoto počtu by se vyhodnocení mohlo urychlit.

Naše řešní (tzv. *RDF indexy*) spočívá v předvyhodnocení a uložení výsledků vhodně zvolených dotazů v databází. Tyto dotazy musí v současné době definovat uživatel databáze, i když by jistě bylo možné je generovat automaticky na základě sledování dotazů pokládaných do databáze.

Pokud se takovýto předvyhodnocený dotaz objeví jako součást jiného dotazu, pak je možné jeho vyhodnocení spojením tabulek trojic a literálů nahradit přímo tabulkou s výsledkem předvyhodnoceného dotazu.

Odstranění několika spojení znamená nejen eliminaci jejich výopčtu, ale také zjednodušení vyhodnocovaného SQL dotazu.

6.1 Implementace RDF indexů

Vytvořili jsme implementaci omezené verze navrhovaných RDF indexů. Tato implemetace podporuje jen indexy, které jsou definovány dotazem, který vyhovuje následujícím omezením:

- predikáty nejsou proměnné
- všechny predikáty jsou různé
- dotaz neobsahuje OPTIONAL, UNION a FILTER
- dotaz neobsahuje anonymní uzly

Pro vytváření indexů jsme rozšířili syntax jazyka SPARQL o klauzili CREATE INDEX.

```
CREATE INDEX jméno_indexu
AS ?proměnná1 ?proměnná2 ...
WHERE grafový_dotaz
```

Jako příklad ukážeme index nad jmény osob:

```
create index osobajmena as ?osoba ?jmeno ?prijmeni
where { ?osoba mt:ot_osoba__jmeno ?jmeno .
  ?osoba mt:ot_osoba__prijmeni ?prijmeni }
```

K výběru indexů, které budou použity pro vyhodnocení dotazu q, používáme jednoduchý hladový algoritmus, který postupně zkouší všechny indexy a pokud je možné některý použít, tak jej použije.

Tento cyklus se opakuje, dokud je v jeho průběhu nalezen alespoň jeden použitelný index. Pokud je takový index nalezen, pak jsou z q odstaněny trojice, které odpovídají indexu, a jejich použití je nahrazeno použitím vyhodnoceného indexu. Protože při každém použití indexu je z dotazu q odstraněna alespoň jedna trojice a nahrazena indexem, tento algoritmus vždy končí.

Snadno lze nalézt protipříklad, který ukazuje, že algoritmus není optimální vzhledem k počtu nahrazených trojic v dotaze q.

6.2 Rychlost RDF indexů

Vytvořili jsme index na základě tohoto dotazu:

```
select ?p ?q ?x ?y
where { ?x ns:p1 ?p . ?x ns:p2 ?y . ?y ns:p3 '2' . ?x ns:p4 ?q }
```

Tento index jsme použili pro vyhodnocení složitějšího dotazu, který vypadá takto:

```
select ?q ?r
where{ ?u ns:p10 ?p . ?u ns:p11 ?v . ?u ns:p12 ?r . ?v ns:p13 '3'.
?x ns:p1 ?p . ?x ns:p2 ?y . ?y ns:p3 '2' . ?x ns:p4 ?q }
```

Druhá čtveřice trojic může být nahrazena indexem. Rychlost vyhodnocení se tím výrazně zlepší. Bez indexu trvalo vyhodnocení dotazu 92 vteřin, s indexem se čas zkrátil na 18 vteřin.

7 Závěr

Předkládané řešení infrastruktury pro sémantický web je svým zaměřením otevřený framework, nikoliv jedno uzavřené řešení. To je podle nás zcela nezbytné vzhledem k rozmanitosti sémantického webu, jeho současné nevyzrálosti a potřebě velmi flexibilní budoucí rozšiřitelnosti.

Jednotlivé části infrastruktury jsou v různých stádiích dokončenosti. Centrální databáze založená na technologii stohu je hotová a funkční, v reálném projektu byla pilotně otestována na řádově desítkách miliónů záznamů. Stejně tak základní nástroje spolupracující se stohem, zejména Tykadlo a unifikační algoritmy. Pro plnohodnotné použití stohu pro infrastrukturu sémantického webu

je však vhodné rozšířit strukturu metadat tak, aby umožňovala pojmout komplexnější ontologie.

Technika filtrů jakožto základních prostředků pro importéry dat i metadat je také převzata z pilotního nasazení [11], konkrétní importéry jsou však ve fázi implementace. Vyhladávač Egothor je plně funkční, avšak nezaintegrován do infrastruktury. Navrhované moduly importu dat a odvozovače sémantiky jsou nyní ve fázi specifikace.

Z dotazovačů je Tykadlo implementované a funkční, SPARQL a vícekriteriální dotazovače jsou pilotně implementovány, avšak doposud nezaintegrovány do zbytku prostředí. Formáty exekutorů a přesná funkčnost dirigentů jsou ve fázi specifikace.

Budoucí práce bude spočívat především v implementaci zbývajících komponent, jejich integraci a následném experimentálním zkoumání vlastností jak celkové architektury tak i jednotlivých modulů. Předpokládáme, že na základě získaných výsledků a zkušeností bude v tomto článku popisovaná otevřená infrastruktura doplněná o další moduly a datové toky.

Reference

[1] Bednárek, D., Obdržálek, D., Yaghob, J. and Zavoral F.: Data Integration Using DataPile Structure, ADBIS 2005, Proceedings of the 9th East-European Conference on Advances in Databases and Information Systems, Tallinn (2005)

[2] Bednárek, D., Obdržálek, D., Yaghob, J. and Zavoral F.: Synchronisation of Large Heterogenous Data Using DataPile Structure, ITAT 2003, Information Technologies – Applications and Theory, University of P. J. Safarik Kosice (2003)

[3] Dokulil, J.: Transforming Data from DataPile Structure into RDF, in Proceedings of the Dateso 2006 Workshop, Desna, Czech Republic (2006) 54-62, http://sunsite.informatik.rwth-aachen.de/Publications/CEURWS, Vol-176

[4] Dokulil, J.: Použití relačních databází pro vyhodnocení SPARQL dotazů. ITAT 2006, Information Technologies – Applications and Theory, University of P. J. Safarik Kosice (2006)

[5] Eckhardt, A.: Metody pro nalezení nejlepší odpovědi s různými uživatelskými preferencemi, Diplomová práce MFF UK Praha (2006)

[6] Galambos, L. and Sherlock W.: Getting Started with ::egothor, Practical Usage and Theory Behind the Java Search Engine (2004) http://www.egothor.org/

[7] Pokorný, J. and Vojtáš, P.: A data model for flexible querying. In Proc. ADBIS'01, A. Caplinskas and J. Eder eds. Lecture Notes in Computer Science 2151, Springer Verlag, Berlin (2001) 280-293

[8] Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D. and Schol, D.: RQL: A Declarative Query Language for RDF. In the Proceedings of the Eleventh International World Wide Web Conference, USA (2002)

[9] Broekstra, J. and Kampman, A.: SeRQL: A Second Generation RDF Query Language, SWAD-Europe, Workshop on Semantic Web Storage and Retrieval, Netherlands (2003)

[10] Seaborne, A.: RDQL – A Query Language for RDF, W3C Member Submission (2004) http://www.w3.org/Submission/2004/SUBM-RDQL-20040109

[11] Kulhánek, J. and Obdržálek, D.: Generating and handling of differential data in DataPileoriented systems, IASTED 2006, Database and Applications DBA (2006) 503-045

[12] Chernik, K., Lánský, J. and Galamboš, L.: Syllable-based Compression for XML Documents. In: Snášel, V., Richta, K., and Pokorný, J.: Proceedings of the Dateso 2006 Annual International Workshop on Databases, Texts, Specifications and Objects. CEUR-WS, **176** (2006) 21-31.

Inverted Index Maintenance

Leo Galambos

Department of Software Engineering
Charles University in Prague, Czech republic
leo.galambos@mff.cuni.cz

Abstract. This paper presents a method for dynamization which may be used for fast and effective inverted index maintenance. Experimental results show that the dynamization process is possible and that it guarantees the response time for the query operation and index actualization.

1 Introduction

The method of this paper addresses how to incrementally update an index in place while still guaranteeing the response time for querying and index actualization. Second, the method uses a native solution which can be easily implemented without complex data structures or extra data space. Therefore, the method can be used in cases where it is not possible to use new advanced methods (see Chapter 2). The paper also shows that some theoretical issues can be eliminated in a practical configuration of a web search engine.

Our goal is the algorithm which will keep a full-text index in a good shape. What is meant by "good shape" is that the data structure is not significantly slower than the fully-optimized index structure.

This paper is organized as follows. The current methods are summarized in Chapter 2. The actualization algorithm, based on the dynamization, is presented and discussed in Chapter 3. Chapter 4 presents the extension for batch processing. The impact on searching phase is discussed in Chapter 5. Issues related to the Web are discussed in Chapter 6.

1.1 Vector space model

In the Vector space model [14, 15], the query (Q) and document (D) are represented as vectors. The vectors are indexed by terms (t_i), or rather, by their numbers ($i = 1 \dots terms$). A document vector is defined as $\overrightarrow{D_i} = (w_{i,j})_{j=1\dots m}$, where $w_{i,j}$ is the weight of the j-th term in the i-th document and m is the number of different terms in a corpus. Generally, terms that are absent from a document are given zero weight. A query vector is defined in the same way ($\overrightarrow{Q} = (q_j)_{j=1\dots m} = (w_{q,j})_{j=1\dots m}$). The document-term matrix $DT = (w_{i,j})_{i,j}$ represents the index. The columns of that matrix (without zero-cells) represent the inverted lists [6]. The number of rows (number of indexed documents) is also termed "the size of the index". The index may contain other values, for instance,

positions of words (tokens) in documents. For purposes of this paper an item in an inverted list is termed a "tuple".

We will assume that inverted lists are stored in one file (inverted file) in an order that reflects the order of their terms. That is, the inverted list of term t is stored before the lists of terms $t'_1 \dots t'_s$, if and only if the term t is (alphabetically) lower than any of $t'_1 \dots t'_s$ terms. This format ensures that two inverted files T and U can be merged by reading them sequentially. Assuming that the inverted file X is built up for the document collection C_X, and its length is L_X, the merge operation then produces a new inverted file V, and it holds: $C_V = C_T \cup C_U$, $L_V = L_T + L_U$, with the operation taking $O(L_V)$ instructions. When a set of inverted files is merged, the process uses $O(L_V \log u)$ instructions, where u is the number of merged files and L_V is the length of the original inverted files.

2 Existing solutions

In this section, the methods for modification of inverted indices are discussed briefly. We assume that the indexed collection of documents has the same properties as WWW, where the number of inserted or deleted documents is smaller than the number of modified documents [8]. Moreover, the number of changes is rather small when held against the whole collection of documents.

There are several solutions for such a configuration, we mention just some in the following overview.

Rebuild. This method replaces an obsolete index with a new one which is built from scratch. Obviously, this way is not effective because it always re-scans all documents in a collection. The method could be improved by distributed processing, but it does not change the fact that this method is wasteful for the document collection assumed herein.

Delta change-append only. Some improvement is achieved when just the modified documents are processed. This can be implemented effectively using the standard indexing technique (known as merging [2], pp 198). For this, the index is merged with an index built for new documents. When a document should be removed, it is only denoted as "deleted". Modification of a document is then realized via Delete+Insert operations. This implementation is very popular and can be found, for instance, in the Lucene engine [1].

Unfortunately, one serious drawback exists - if we execute many Delete operations (also part of the Modification operation), the index structure stops working effectively, and what is more important, it is not possible to easily detect this situation. Therefore, from time to time one must optimize the index. Such an optimization phase may take a lot of time, which again makes this method less effective. The issue was studied by many researchers, for instance [4]. Unfortunately, their solution was demonstrated with simplified conditions; it would be interesting to see whether these conditions hold in a heavy load system.

Forward index. Another method was proposed by Brin and Page for the Google search engine [3]. This method constructs an auxiliary data structure (forward index) which speeds up the modifications in the main (inverted) index.

The term "modifications" means the real changes of the existing values stored in the index. Other modifications (insertion or removal of values to/from the index) are realized using other approaches, e.g. Btree [5].

Landmark. Research in this area continues and new approaches are still developed, for instance, a landmark method [9] which introduces special marks in the index. The marks are used as initial points from which some values in the index are derived. Therefore, if one modifies the mark, all values dependent on the mark are also shifted. It was shown in the cited paper that such a case often happens in an index built for WWW and, as a result, the landmark method was faster than the forward index.

This paper. We will try a straightforward way that is based on the dynamization of a static index structure. The method solves the issue (where the index structure can stop working effectively) of the "Delta change-append only" approach. Moreover, the method does not need extra data space (as does a "Forward index"), and even works when the inverted lists are compressed and not based on the index structure supposed for the "landmark" method, i.e., when positions of words in a document are not stored. Last, but not least, the method economizes a file system structure, because it may keep all the index files defragmented.

3 Dynamization

The dynamization we use is inspired by Mehlhorn's algorithm [11–13] for common data structures, e.g., hash tables. The algorithm will be modified and reformulated for an IR system. Let us agree upon the following terminology: *a Barrel is an autonomous index that is often static (it represents the inverted file above); a Tanker is an index that is decomposed to smaller barrels, and it disposes of dynamization as the actualization algorithm.* The term (Barrel) was used in many previous papers, e.g. [3], but it is used in a different meaning in this paper.

Example 1. One can build a simple index (of size 1) for any document. These indices are always barrels, because the index is static (for a given document). Moreover, the index can be searched, therefore it is also autonomous. Later we will show how these simple barrels are organized into tankers, and how the larger barrels are constructed.

We already know that a document can be easily transformed to a simple barrel, thus barrels are considered instead of documents in this paper. Firstly, let us agree upon the following notation:

Notation 1. *Let B be the barrel that is built for the collection of documents C_B. The barrel B offers the query operation $search_B(q, C_B)$ (for a query q). It does not matter how the search is implemented. If n is the number of documents in the collection C_B, then the size of barrel B is $B.size() = n$; the time we need to build B over $|C_B|$ is $Ti_B(n)$; the space we need for the structure B is $Sp_B(n)$; the time we need to compute $search_B(q, C_B)$ is denoted $Q_B(n)$.*

The document removal operation is implemented using a bit array which sets the bit related to the document to 1, if and only if the document is removed from the collection. Obviously, such documents must be also filtered out of the hit lists prepared by $search_B$. This operation can be easily implemented, however.

Notation 2. *Let B be the barrel. The number of documents denoted as removed in the barrel is $B.deleted()$. The number of live documents in the barrel is $|B| = B.size() - B.deleted()$.*

Similarly, we define the notation in the case of tanker T – $Ti_T(n)$, $Sp_T(n)$ and $Q_T(n)$. Moreover, we must require a condition which is described in the following definition:

Definition 1. *Let T be the tanker containing barrels $B_0, B_1, \ldots B_r$ (also accessed as $T[i]$ in the algorithm below). Next, let C_i be a short form of C_{B_i}. We request that $\forall i \neq j : C_i \cap C_j = \emptyset$. If and only if a position j is not occupied in the tanker, we define: $B_j = \emptyset$, $C_j = \emptyset$. When the position j is not occupied, we define $search_{B_j}(q, C_j) = \emptyset$.*

Let us denote the tanker collection $C_T = \bigcup_{i=0}^{r} C_i$. Tanker T is able to solve the query operation (for a query q) as $search_T(q, C_T) = search_{B_0}(q, C_0) \oplus search_{B_1}(q, C_1) \oplus \cdots \oplus search_{B_r}(q, C_r)$, where $\oplus$ denotes the composition of partial results. This operation is computable in constant time because we only compute hit lists of limited length.

Moreover, we request:

$$\forall i : B_i \neq \emptyset : 2^{i-3} < |B_i| \text{ and } B.size() \leq 2^i \tag{1}$$

This condition ensures that no part of a tanker is degenerated after a number of *Delete* operations. If such a situation did happen, we would execute a reorganization which would repair the tanker structure (see below). After this operation, the condition would again hold.

Algorithm 1 Algorithm of dynamization – Tanker, routine insert(B:barrel)

1: $k := \min\left\{x; |B| + \sum_{i=0}^{x} |B_i| \leq 2^x\right\}$;
2: $S := \{B_i; i < k \wedge B_i \neq \emptyset\} \cup \{B\}$;
3: $\forall i < k : B_i := \emptyset$;
4: $B_k := Merge(S)$;{Merge barrels of set S; values of removed documents are left out}

The actualization in a tanker after barrel insertion can be achieved by Alg. 1. The algorithm ensures that the tanker will reorganize its inner structure and that the condition (1) always holds.

The tests of the algorithm revealed that we can assume that $Ti_B(2n) = 2Ti_B(n) = Ti_B(n) + Ti_B(\frac{n}{2}) + Ti_B(\frac{n}{4}) + \cdots + Ti_B(1)$. Moreover, the algorithm works as fast as merging with a merge factor equal to 2. In other words, the

dynamization should not be used for the index construction from scratch, because we can easily replace it with the merging process of higher factor. On the other hand, in a real system the index must also reflect other modifications than simple insertions, for instance, when a document is changed or removed. That is the point discussed next.

4 Batch updates

Until now we assumed that the index is modified by insertions. This chapter introduces batch processing, when a set of insertions or removals is executed at once. For clarity's sake we will present a solution to a case where the number of removals is almost the same as number of insertions. All these changes come from modifications of documents, because we use the Delete+Insert approach. It is also assumed that the index is already huge and the number of removals or insertions is rather small (less than 30%).

The algorithm 2 repairs a tanker which reflects changes in documents D. It transforms an existing tanker's structure b to a new one – n (both are arrays of barrels, so that B_i is represented as $b[i]$ or $n[i]$). All merge operations denoted by the *merge* method are delayed until we really need the product of the merge. The empty barrel is denoted as *empty*. We also note that the $merge(B_i)$ operation creates a copy, not containing any values of documents which are denoted as removed, of barrel B_i.

For a simulation of our algorithm, we utilize the formula $Ti_B(2n) = 2Ti_B(n)$, which holds in our real IR system. Then, we can estimate the time needed to realize the steps of the Alg. 2.

The existing methods introduced in the Chapter 2 are often compared with the method labeled as "rebuild from scratch" (RFS). We will apply the same approach.

The simulation is organized as follows: the original tanker has 2^N documents and it consists of one barrel B without deleted documents $B.deleted() = 0$. The barrel is placed at position N. The value N is placed on the X-axis in the figure. The updater always removes and inserts *chg* percent of the index size (Y-axis). We measure the number of read and write operations S needed to realize the Alg. 2. The Z-axis then presents the ratio $\frac{S}{M}$, where M is the number of read and write operations needed by the merging RFS strategy of factor 100. The measured values are the average of 10000 reiterated runs. The 3-d figure (Fig. 1) is equivalently represented as a contour figure (see Fig. 2).

It should be noted that we can simulate the algorithm for collections which are almost unrealistic for evaluation. A collection of 2^{46} documents is not easy to obtain, and 10000 repetitions of the experiment are pretty unrealistic on the hardware available.

The simulation was verified on a collection of size 2^{21}. When the test was repeated with 10% of the documents randomly changed, it was saved about 86% of time comparing to the complete rebuild. Our theoretical assumption of 88% was not achieved, but the difference is small, such that it can be rooted in the

Algorithm 2 (Algorithm of dynamization) Tanker, routine **modify(D:documents)**.

```
1: {step1: remove all obsolete data}
2: mark existing documents D as deleted/invalid in the index using bit vector
3: {r is the number of cells in this tanker}
4: for all i=0...r do
5:    if |b[i]| == 0 then
6:       n[i]=empty
7:    else
8:       if |b[i]| > 2^{i-3} > 0 then
9:          n[i]=b[i]
10:      else if i==0 then
11:         n[i]=b[i]
12:      else if |n[i-1]| >= 2^{i-2} then
13:         n[i]=merge(n[i-1]);n[i-1]=merge(b[i])
14:      else
15:         y=merge(b[i],n[i-1])
16:         if |y| >= 2^{i-2} then
17:            n[i]=y;n[i-1]=empty
18:         else
19:            n[i]=empty;n[i-1]=y
20:         end if
21:      end if
22:   end if
23: end for
24: {step2: if a barrel is still small then eliminate it}
25: zombie = {n[i]; n[i].size() > 0 ∧ |n[i]| ≤ 2^{i-3}}
26: new=merge({D},zombie)
27: {step3: insert the new barrel}
28: while all positions for new are occupied in n, merge the barrels with new
29: new is saved into the array n
30: {step4: execute all merges we planned}
31: COMMIT: array n becomes b, obsolete barrels are discarded
```

fact that we had to measure only some parts of engine's routines, and in JAVA, we were not able to use accurate system timers.

5 Searching

The typical searching phase consists of two steps. First, we look up all terms in a dictionary to find offset positions of the respective inverted lists. Next, the inverted lists are read and evaluated. Obviously, if the dictionary is cached in a hash table in a memory (RAM), then the majority of time is consumed by the second step.

Due to the fact that a tanker $T(n)$ must have more than $\frac{T.size()}{8}$ live documents, we can claim that a query over the tanker needs time $Q_B(8n)$ to be

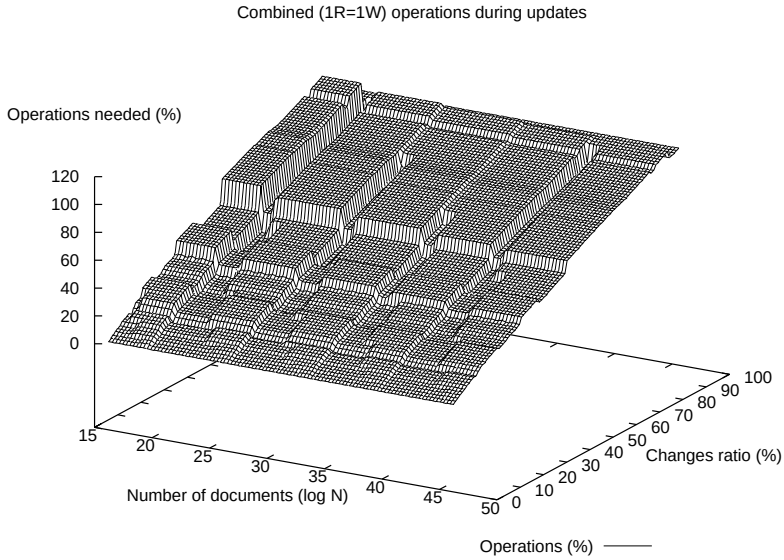


Fig. 1. Simulation.

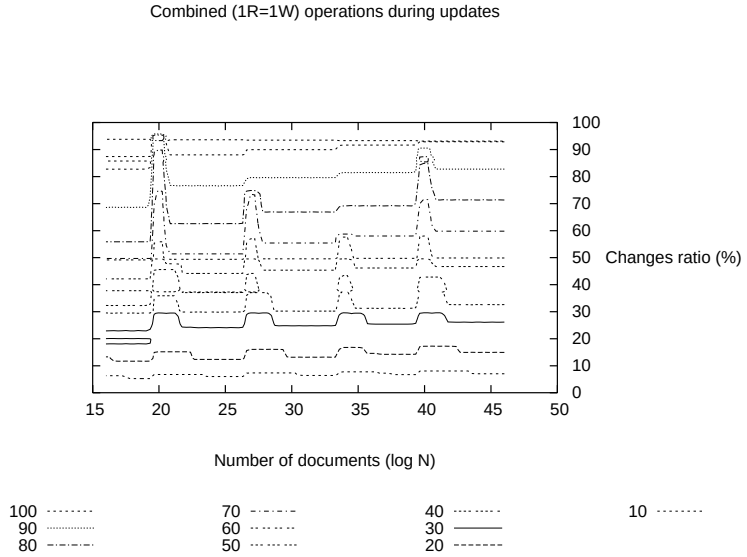


Fig. 2. Comparison: dynamization versus RFS.

solved. This holds for a single thread environment (STE), but the query could also run over each of the tanker's barrels concurrently. Then the time is rather limited by the construction size of the largest barrel in the tanker.

We are also interested what happens in a long term. It could happen that the tanker almost always contains just a few deleted documents, or rather tuples of documents denoted as deleted, and the limit case (the tanker has up to 7/8 of deleted documents) occurs only time to time and infrequently. Obviously, the opposite case could also happen.

The answer to this is given by Figure 3 describing the experiment. The tanker T is an index over 2^N documents and it consists of one barrel B without deleted documents $B.deleted() = 0$. The barrel is placed at position N in the tanker ($T[N] = B$). The updater always removes and inserts chg percent of the index size (X-axis). We measure the ratio $\frac{T.size()-T.deleted()}{T.size()}$ and the Y-axis then presents the average value of the ratio after 10000 updates. It should also be noted that we simulated the algorithm for collections which are almost unrealistic for evaluation ($15 < N < 47$), but the collection size has not any impact on the results, so we decided not to include this parameter in the experiments presented herein.

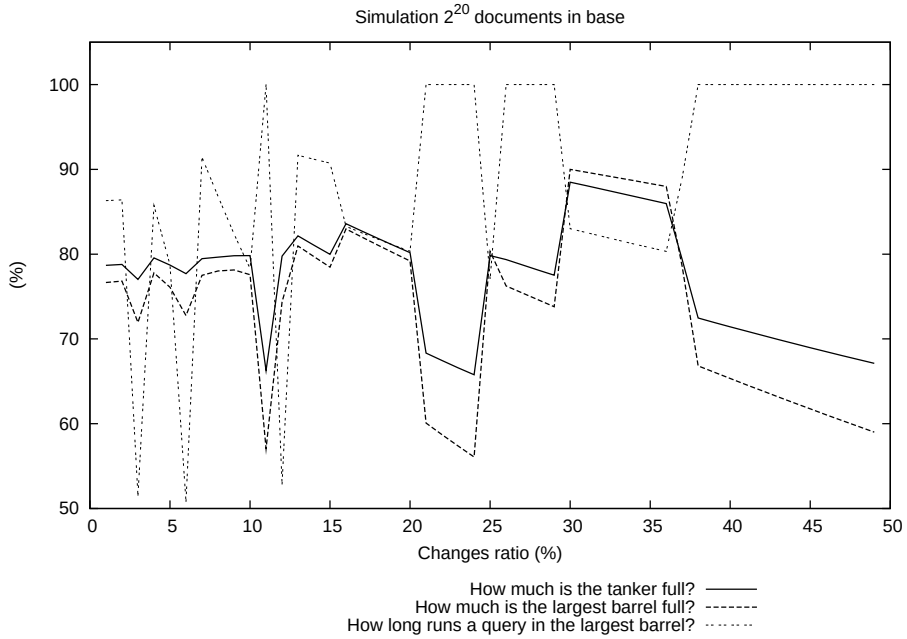


Fig. 3. Simulation of the dynamization algorithm.

The figure shows that we lost about 25% (in a long term) when using the dynamization for an index that is updated concurrently and the number of changes

is less than 10% – a typical situation for a live web search engine. We may also lost about 35% in several concrete cases. Fortunately, our algorithm could be enhanced with a routine that checks whether we are in such a bad case. If so, the routine could shorten or lengthen the period of updates and move us to a better case.

The presented figure is related to the situation when the barrels of a tanker are evaluated sequentially. How is this changed when the barrels are evaluated in a multi-threaded environment (MTE) and they operate over a shared result list?

The answer is given by the ratio of the construction size of the largest barrel and the number of live documents in the tanker (as explained above). The simulation is presented in Figure 3. It shows that we could save up to 50% of time (comparing to a fully optimized index) when the number of changes is within a reasonable boundary (less than 30%).

Now, another point is interesting as well – how many live documents are stored in the largest barrel? The answer to this shows us how effectively we can evaluate a query in MTE. The respective simulation is presented in Figure 3 and we could again claim that only about 25% of performance is lost due to the deleted documents (in a long term).

It was still told about the average case, or rather, about the long-term effectiveness. However, a real threat was not discussed yet – what will happen in the worst case when the algorithm only guarantees that the response is given (up to) eight-times slower than with the fully optimized index? Could it be improved this parameter? Can the speed be improved when the barrels are ordered by a page rank?

6 The Web, practical notes

Our algorithm could run eight-times slower than the fully optimized index in the querying phase. However, according to [10], we could still save more than 60% of time with skipping of $L = 100$ (L – skip factor [10]). All we need to do is implement our bit array (storing the 1-bits for deleted documents) as an inverted list and include it into the queries. Since the bit array is (can be) fully kept in RAM, we can save more than was described in the cited paper – the authors assumed that all inverted lists are read from a hard disk, while we have one of them in RAM.

Moreover, if the inverted lists are sorted by a document rank (the highest first), then just a few first blocks are read, because all other hits would be computed for less-important pages, so they would not be included amongst top-N.

We must only find a correct starting point of an inverted list (disk seek operation) and start to read sequentially. In fact the most expensive routine is the first seek operation. According to our tests with the EGOTHOR system [7] we can claim, that a disk block (4kB) is able to store about 500-1000 tuples of an inverted list. Obviously, such a block is often quite enough to generate top-10. Using the dynamization, we may have to generate top-80 (up to 70 documents

could be still denoted as deleted), but this could be still covered by the tuples of the first block implying no extra I/O operations.

The following experiment shows what happens with the index, when the index is ordered by page rank. Our configuration follows these rules:

- only *chg* documents are changing;
- the probability that a document is modified since the last update is *chg* (changes ratio);
- top-10 is always constructed;
- the index is updated 10000 times;
- if a document was not modified in last 1000 updates, it is supposed to be static.

Since the engine operates on the Web, we assume that the top-10 hits are (primarily) looked for in the last barrels constructed with Alg. 2. It is based on the fact, that the most wanted documents are probably saved in the most up-to-date documents of high page ranks.

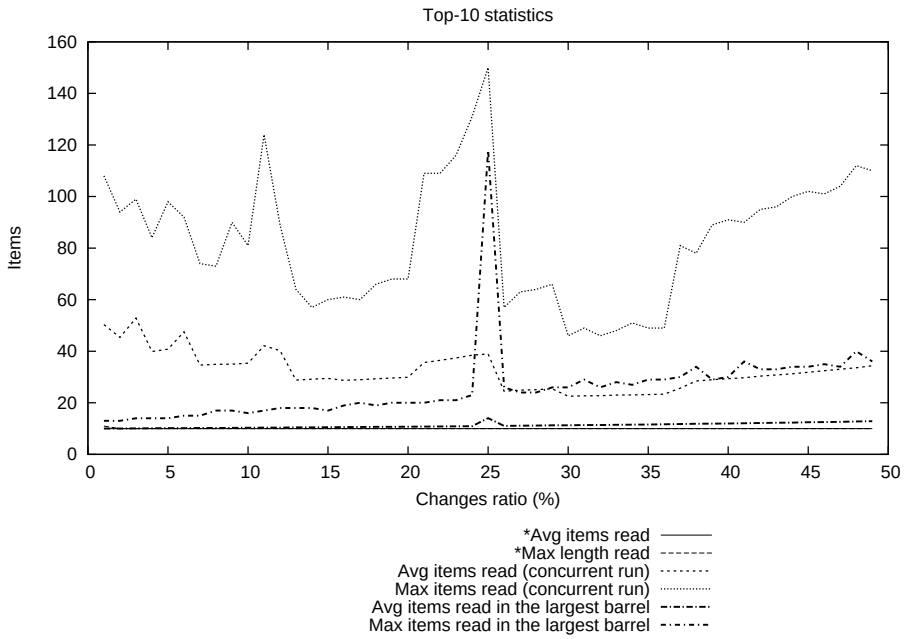


Fig. 4. Length of the inverted lists for top-10 evaluation.

However, if our query algorithm cannot adopt the evaluation (where the last barrels have higher priority), all barrels must be asked for top-N lists and the final top-N is their joint. Then one might be interested in the maximum length of an inverted list we must scan in any of the barrels (they are asked for the

hit lists concurrently). Next, the total number of tuples read in all barrels is an interesting value as well.

Both values (including their means after 10000 updates) are presented for top-10 queries in Figure 4. The figure also presents the number of tuples read in a situation when we can adopt the special evaluation as mentioned above. These values are denoted by an asterisk.

For instance, when 5% documents are changing since last update, then we must read 10.15 tuples from the largest barrel in an average case (the maximum value is 16). It implies, if the querying runs concurrently, the queries run 1.6x slower than over a compact index.

The "1.6x" factor could imply more I/O operations for high N. Fortunately, if we prepare the hit lists for "top-N queries" where N is in reasonable boundaries (for instance <100), the higher demand for read operations is not significant. This is based on the fact that the inverted lists are always read by disk blocks and it does not matter whether we read 1 byte or the full block. If one tuple was stored using 5 bytes then one disk block of 4kB could save about 819 tuples and the factor of 1.6 would not represent any serious threat.

Although we believe that our simulation is good and it accurately reflects reality some of the experiments were verified on a collection of size 2^{21}. When the test was repeated with 10% of the documents randomly changed, it was lost about 25% of time on querying in STE mode (the test computed full hit lists, not only top-10). Our theoretical assumption of 21% was not achieved, but the difference is small, such that it can again be rooted in the fact that we had to measure only some parts of engine's routines, and in JAVA, we were not able to use accurate system timers.

The same test was also repeated in MTE. For this, we borrowed a high performance RAID-0 array that could represent a real hardware for a search engine. JDK 1.5 and Java NIO were used on 2xAMD Opteron and the result surpassed our theoretical simulation by five percent (62% comparing to 57%). The next experiment was pointed to top-10 calculation. Unfortunately, it was not possible to measure any significant difference, because all the differences were negligible. This is also confirmed by the emulation presented in this paper.

7 Conclusion

It was shown that the method can work much more effectively comparing to "rebuild from scratch". On the other hand, the trade-off is based on the fact that we are satisfied with slower querying – up to eight times comparing to a fully optimized index. Fortunately, it was shown that the factor can be further lowered. Our preliminary experiments and simulations show that the extra operations are almost fully compensated by the hardware architecture of current computers, where it does not matter whether a program reads one byte or hundred bytes from a disk.

The method can be used in configurations where one cannot easily update values stored in inverted lists, and it makes the modern techniques (e.g. landmarks) unusable.

References

1. Apache, Jakarta project: Lucene. `http://jakarta.apache.org`
2. Baeza-Yates, R. and Ribeiro-Neto, B.: Modern Information Retrieval. Chapter 8. ACM Press 1999
3. Brin, S. and Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30 (1998) 1–7, 107–117
4. Clarke, C. and Cormack, G.: Dynamic Inverted Indexes for a Distributed Full-Text Retrieval System. TechRep MT-95-01, University of Waterloo, February 1995
5. Cutting, D. and Pedersen, J.: Optimizations for dynamic inverted index maintenance. Proceedings of SIGIR (1990) 405-411
6. Harman, D. K., et al: Inverted files. In Information Retrieval: Data Structures and Algorithms, Prentice-Hall (1992) 28-43
7. EGOTHOR, JAVA IR system. `http://www.egothor.org/`
8. Lim, L., et al: Characterizing Web Document Change, LNCS 2118, 133–146, 2001
9. Lim, L., et al: Dynamic Maintenance of Web Indexes Using Landmarks. Proc. of the 12th W3 Conference, 2003
10. Moffat, A. and Zobel, J.: Self-Indexing Inverted Files for Fast Text Retrival. ACM TIS, 349–379, October 1996, Volume 14, Number 4
11. Mehlhorn, K.: Data Structures and Efficient Algorithms, Springer Verlag, EATCS Monographs, 1984
12. Mehlhorn, K. and Overmars, M. H.: Optimal Dynamization of Decomposable Searching Problems. IPL 12, 93–98, 1981
13. Mehlhorn, K.: Lower Bounds on the Efficiency of Transforming Static Data Structures into Dynamic Data Structures. Math. Systems Theory 15, 1–16, 1981
14. Salton, G. and Lesk, M. E.: Computer evaluation of indexing and text processing. Journal of the ACM, **15** 1 (1968) 8-36
15. Salton, G.: The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall Inc., Englewood Cliffs, 1971

Semantic Web: Web Patterns in Web Page Semantics

Miloš Kudělka[1], Václav Snášel[1], Eyas El-Qawasmeh[2] and Ondřej Lehečka[1]

[1]Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava–Poruba, Czech Republic
{milos.kudelka,vaclav.snasel,ondrej.lehecka}@vsb.cz
[2]Department of Computer Science,
Jordan Universit
eyas@just.edu.jo

Abstract. This paper introduces a novel method for semantic annotation of web pages. Semantic annotation is performed with regard to unwritten and empirically proven agreement between users and web designers using Web patterns. This method is based on extraction of patterns which are characteristic for concrete domain. Patterns provide formalization of the agreement and allows assignment of semantics to parts of web pages. Experimental results verify the effectives of the proposed method.

1 Introduction

Semantic annotation is the process of adding formal semantics (metadata, knowledge) to the web content for the purpose of more efficient access and management. Currently, the researchers are working on the development of fully automatic methods for semantic annotation (see [4]).

This paper has two objectives. The first one is to simplify the querying, and the second is to improve relevance of answers.

For our research we consider important semantic annotation and tracing user behavior when querying in search engines. Published research results showed the necessity of perceiving annotation and query formulation as two sides of one coin. We consider the semantic annotation in two meanings according to [15].

- Page related semantic metadata.
- A method to generate the metadata.

Currently, there are two trends in the field of semantic analysis. One of them provides mechanism to semiautomatic (or manual) page annotation using ontology description languages and creation of semantic web documents. The second approach prefers an automatic annotation of real internet pages. This approach expects that acquiring annotation manually during document creation is hardly achieved.

Across all directions it is possible to see the use of ontology based mechanisms, in the case of second approach along with knowledge bases. Our view at the problem of semantic analysis is in many cases similar to shown techniques and tends to the automatic approach of semantic annotation. However our motivation of what and how to annotate is different since it is a higher abstraction level. The reason for this is that we do not work with the semantics in meaning of the precise content of document, but more likely with the form of the document, which is related to the content and chosen domain.

The approach which is similar to ours is mentioned in [16]. It uses TXL language to analyze chosen parts of pages to obtain structured domain specific information (tourism domain). Other similar approaches include: automatic transformation of arbitrary table-like structures into knowledge models [21], formalized methods of processing the format and content of tables to relevant reusable conceptual ontology [24], or domain-oriented approach to web data extraction based on a tree structure analysis [22].

On the side of query string analysis and construction appears approaches which are helpful for the user when formulating his requirement. One of the possible approaches is to use a cognitive search model [27] which describes a web search system prototype based on ontology that uses a cognitive model of the process of human information acquisition. Another way is to help user with specification of query based on interaction with user using genetic algorithms and fuzzy logic, see [6, 17, 12].

The organization of this paper will be as follows. Section 2 presents short description of our research. Section 3 presents the patterns basics. Sections 4 and 5 present goals of our research. In sections 6 and 7 there are described preparation of our experiment and analysis of results, and finally section 8 is the conclusion.

2 Our research

This paper introduces a new perspective which connects both of our goals from introduction in native way. Key aspect of our perspective is smart focusing on user and his expectances.

We want to move towards users expectations when searching information on web. To be able to do this we need the user to share his expectance with us. A simpler way is to turn our questions to professional web sites designers. Their mission is to fulfill users expectance. Proof of this is that high-quality web pages and web solutions are widely accepted by users.

Professional web designers apply practices which come up from users experiences. These practices relate with human sensation and allow simple orientation in supplied information. Solutions of the same problem are solved by different developers differently but at certain level solutions are all the same. Similar web pages contain similar components. We can define this conformity such that there are similar web page components on the pages within the same specialization. These components are designated as web patterns. Web patterns give us seman-

tic information which is based on definite and empiric proved settlement between developers and users. The following section expresses the idea of patterns.

3 Patterns basics

Patterns occur in variable areas from architecture, where they were identified first, through user interface design, to software design (see [1, 9]). In [23] there is a very clear description what patterns are.

In essence, patterns are structural and behavioral features that improve the applicability of software architecture, a user interface, a Web site or something another in some domain. They make things more usable and easier to understand.

GUI patterns supply solution of typical problems within design of user interface. Patterns provides the users with a way that explains how to deal with common situations. Typical examples are organization of user controls into lists or tabs and so on. GUI patterns describe on general level how to make structure of information within user interface. They tell us which components to use, how they should work together and how to work with them. Many examples can be seen in [23, 2, 25, 11, 20]. GUI patterns are rather technical; they describe how to solve certain problem. Different domains then provide environment for use of patterns in concrete context. In this article we choose selling products domain (there are patterns identified for other domains as well, [26]). We can find common features in user interface within the selling products domain. These features express typical tasks with information (showing the price information, purchasing possibility, showing the product detail information). When implementing web site the web designers proceed the same way. They also use patterns even if they dont call them so (see [7]). During the analysis of selling products domain and even our experiments we worked with a ten of patterns which come out from [7].

Fig. 1 demonstrates an example where there is a cut of web page with selling of product on eBay.com. Founded patterns are graphically marked on the page. There are five patterns Sign on possibility, Price information, Purchase possibility, Rating, Special offer.

GUI and domain patterns are designated for web designers and experts on domains. They are written with free text whereas structure of description is formalized. They have reference character which implies different way of their implementation. For our research purposes we do need to find description of patterns which will be independent of the web designers and implementation and which will be useful for semantic analysis of web pages. Patterns on the page represent, in certain degree, what user can expect (as a consequent of agreement between web designers and users). For us this is the key prerequisite for technical usage of patterns. We are able to find out algorithms which are able to determine whether a pattern is on the page or is not and as a consequence of this we can annotate this page (and then use this annotation next time).

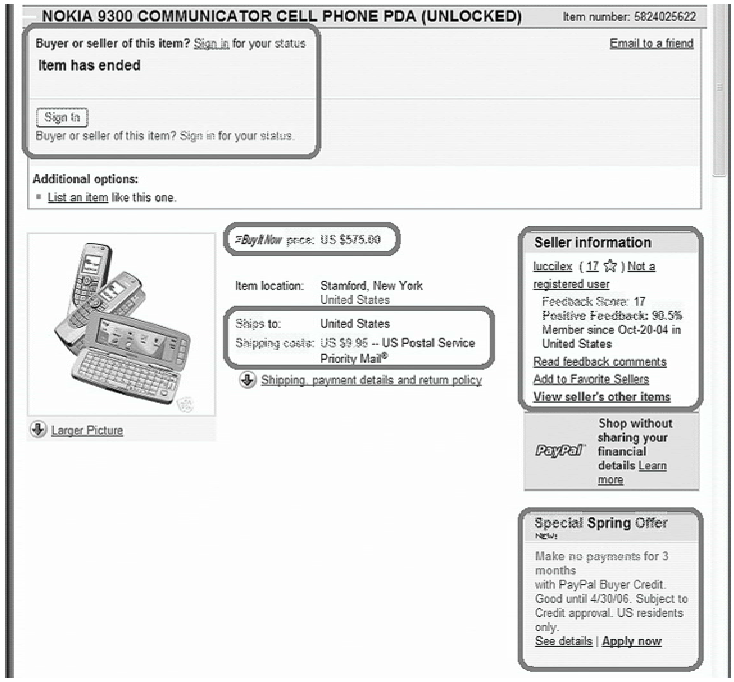


Fig. 1. Web page with marked patterns

4 Query simplifying

In the proposed method, the user selects a set of words to specify his requirement. Our approach to querying is based on the same principle but with more accessible expression tool for users. Pattern Price information contains much stronger information that the price is on the page.

With the patterns we can set up catalogues and profiles which simplify the selection process for the user (see Fig. 2). It will remain to user that he will have to enter subject of search but patterns will help him to specify expectations.

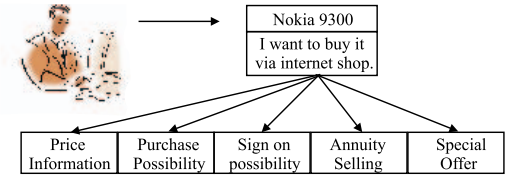


Fig. 2. Query from selling product domain

5 Improvement of answer relevance

Note that when annotating pages we are working with the same information which the user uses to make query. Moreover this information has stronger semantic content than for example enumeration of keywords. Assume that we have annotated pages in a database with regard to patterns so there is information about which patterns are contained on each page. We can then use this information in two manners.

1. When showing web search engine results. For every displayed page link in returned set of links, we can add information about which patterns has been founded on the page. So user can on the first look recognize whether the page will fulfill his expectance.

2. We can count with the patterns already when performing search and sort page links with regard to weight of required patterns we have found on the page.

As it is written previously, patterns describe widely an accepted solutions to users. As a consequence of this, user will be given high-quality designed pages earlier in selection, it means on the first positions of page links result set.

6 Experiments preparation

In the beginning of this document it was explained that we will model user expectations with patterns as a common tool for communication between user and web designer. There is problem in formalization of this approach. Patterns on the page do not appear in exact form. Patterns more likely follow the users point of view, less likely the technological aspects. The pattern is therefore very little dependent on implementation details. We assume that a crucial feature of presence of the pattern on the page is that individual elements of the pattern appear more or less together. More formal description of this deduction is described again in [23, 18]. According to the Gestalt principles, the visual systems usually implement the four basic principles: Proximity, Similarity, Continuity, and Closure.

Following the four principles (Gestalt principles) we can suppose a page pattern as a group of characteristic technical elements (whose are based on GUI patterns such as lists, tables, continuous texts) and group of domain specific elements for the domain we are involved in (typical keywords related to given pattern and other entities such as the price, date, percent etc.). The key aspect of the pattern manifestation is that the introduced elements are close to each other. We can focus on the structure of the page during page analysis. Although the form is not the most important thing (because it may differ in implementation), but the content is.

It is not necessary to provide the deep analysis of page structure, because the technical elements provide just the environment for keeping the information together.

6.1 Choosing domain and domain patterns

We chose the domain of e-commerce for testing purposes. Our goal was to integrate common users to testing so we chose one of the most used domains. During the domain analysis phase we sorted out nine domain web patterns Sign on possibility, Price information, Purchase possibility, Special Offer, Annuity selling, Product details, Comments and reviews, Discussion and FAQ, Advertising. They are the semantic elements, which are commonly expected by the user along the identification of the product.

6.2 Pattern dictionary preparation

For every one of the introduced web patterns we have manually chosen a group of words, which occurrence is characteristic for pattern on the web page. The words have been inserted into the database and serve as input for algorithms performing analysis of web pages. The set of chosen words needs to be understood as a starting set, which is automatically expanded according to deeper analysis of the pattern (the similar approach is presented in [5, 14]). Every pattern has its own dictionary of words. We chose the words that are usually used by users when querying in the search engines price, EUR, USD, offer, discount, stock, basket, buy, shop, review, forum, for sell, specifications, We have assign more than five words to each pattern preferring most frequently used words in various associations. The significant property of the dictionary words is that they are domain associated. We expect that there will be satisfied certain characteristics [13]:

- The dictionary is not too large.
- The words occur in certain schemas.
- The meaning of words is more or less unambiguous.
- The words appear frequently in the text.

During operation of the system, data is collected from algorithms processing texts of pages. Data is then continuously analyzed by algorithms which automatically precise dictionaries for each single pattern.

6.3 Patterns extraction

The main task for solution of the problem we are dealing with is to find the mechanism of pattern detection on the page. We had to develop algorithms working with content of web page. The product from the application of these algorithms has to be answer to the question about weight of pattern on the page.

To solve this task it is possible to follow various approaches. We came up in our design that in semantic annotation we need to find elements which

- are dependent on the meaning of what pattern represents for user,
- are independent of pattern implementation.

In our experiments we simplified pattern formalization problem using set of words and data entities which are characteristic for the pattern. We can choose one concrete pattern (for example, Discussion) and make analysis of big amount of web pages with discussions. After the analysis we can find out that there is quite small group of word and data entities by which it is possible to recognize the pattern.

For example, instance of pattern is some set of analyzed text segments which contains pattern entities (we do not focus on meaning of group of words but we only focus on the presence). For discovery of algorithms which can be useful for finding and analyzing selected segments the Gestalt principles can do us a good turn.

1. For proximity we defined method how to measure closeness (distance) between entities in searched text segments.

2. For similarity we defined method for measuring similarity of two searched text segments (for Discussion we are able to identify repetition of replies). We work with comparison of trees representing text segments.

3. For continuity we defined method how to find out whether two or more found text segments make together instance of pattern. We assume that two or more little- similar text segments (trees of entities from one pattern) match together.

4. For closure we defined a method for computation of weight of one single searched text segment. In essence we used two criteria. We rated shape of the segment tree (particularly ratio of height and entity count) and quantity of all words and paragraphs in text segment. On the overall computation of weight also the proximity rate participates.

With complex usage of all mentioned principles we have implemented an algorithm which offer excellent result in pattern extraction. The algorithm uses only plain text and despite it the algorithm is successful in more then 80% of cases.

6.4 Algorithm

Input for the algorithm is both set of entities which represents each word and data entity from the text of a web page and set of characteristic pattern entities. The algorithm compares these entities with characteristic pattern entities and creates representation of text segments called snippets (see [8]), which can belong to (or compose) a pattern. These representations are then used for further computations and the result is value representing weight of pattern occurrence on the page.

Listing 1.1. ComputeWeight(Input Page)

```
for each page entity in all page entities
{
    if page entity is pattern entity
    {
        if does not exist snippet
        {
            create new snippet in list of snippets
        }
        add page entity to snippet
    }
}

for each snippet in list of snippets
{
    compute proximity of snippet
    compute closure of snippet
    compute value(proximity, closure) of snippet
    if value is not good enough
    {
        remove snippet from list of snippets
    }
}

compute similarity of list of snippets
compute continuity of list of snippets
compute value( similarity , continuity ) of pattern

return value
```

6.5 Experimental application

A web application was implemented which uses Google Web API (see [10]). The system queries the Google for a set of few tens of pages which correspond to the users conditions. Then the application downloads the pages and extracts plain text from the HTML code. On the plain text an analysis is performed, and patterns extraction and evaluation of pattern weight. Then the page is evaluated as a whole. Pages are then sorted according to the overall computed value.

We tested our extract algorithms on PC Intel Pentium M 1.6 GHz with installed Windows XP. We extracted 9 patterns on each page. Performance of algorithms was 100 pages in about 1 second. Average time of 1 pattern extraction was approximately seconds.

7 Result analysis

Quality of our algorithms can be measured by the comparison of common search engine results and results of our application. There is weight of found patterns and thus fulfilling of users expectance evaluated on each page.

There were more than 200 searches of products tested (cellular phones, computers, components and peripheries, electronics, sport equipment, cosmetics, books, CDs, DVDs, etc.) in four profiles. We usually worked with first thirty of found pages; for dozens of products we tested set of first one hundred found pages. For wide testing of our application we select a miscellaneous group of people. Among others there were students of different types of schools and also people dealing with product selling on the internet.

During experiments we collected 31,738 various web pages which we got from the Google search engine using queries on products. After the analysis we discovered that on the 11,038 web pages there was not any extracted pattern even though our queries were focused on pages containing these patterns. Even though we must count with queries on which it is not possible to find enough relevant answers and also with inaccuracy of our algorithms it has to be noted one interesting thing. In spite of very precise query to searching engines user has to count with approximately one quarter up to one third of irrelevant web pages which dont contain expected information.

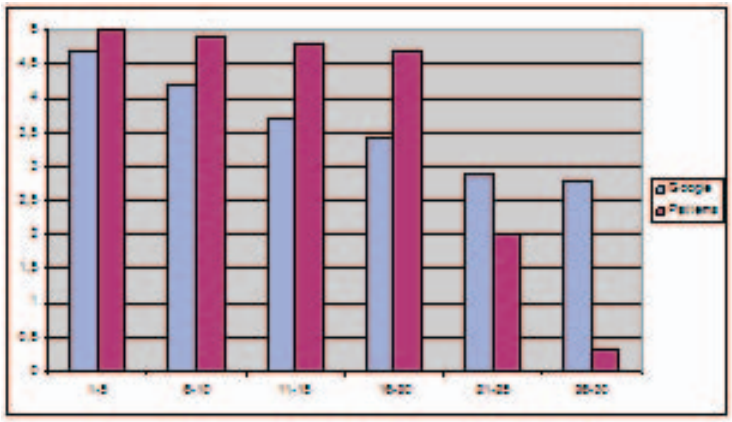


Fig. 3. First 30 pages (default and patterns sorted)

Fig. 3 shows the query results on concrete products. It includes only those queries on which exists multiply times more relevant pages than our first 30 analyzed web pages from search engine (it includes more than 200 various queries).

On the shown graph there are on the horizontal axis first 30 found pages in groups of 5. On the vertical axis there is count of relevant pages. Left columns

show counts using Google search engine. Right columns show counts using our experimental application. From the show graph flows that:

1. Irrelevant web pages are moved to the end of the result set (our approach eliminates mistakes in order of pages)

2. Using our ranking the user reaches the expected pages earlier.

In our experiment we were searching web pages in sets of thirty using very precise query. The query contained product identification (Nokia 9300) and group of six words from the pattern dictionary connected in OR relation for making query more accurate. From the searched pages our algorithm extracted nine patterns (Price Information, Purchasing Possibility, Special Offer, Annuity Selling, Product information, Discussion, Review, Sign on possibility, Advertising). For evaluation of each pattern we used seven criterions. Each criterion was rated using three-degree scale. In all it is expressed using 21 Boolean values.

We were searching optimal visual representation of our experiments results. Finally we chose very simple form (see Fig. 4). It is a two-dimensional table which contains set of downloaded pages in the same order as from the search engine in rows (one row represents one page). In each row there is result of the extracting algorithm processing.

Each column on the Fig. 4 represents rating of each criterion of patterns mentioned above. In each column there are up to 21 values denoting satisfaction of each criterion. Empty spaces represent values less then the lowest value of used scale.

Fig. 4 shows part of the table containing more then 30,000 rows. There are eight sets of pages on the figure. The first set (at the top of the picture) represents answer on query requiring discussions and reviews. Under the set there is a set representing query requiring purchase possibility with special offer and with annuity selling. In the middle of the picture there are four sets of pages representing query requiring only purchase possibility. The one set before the last set represents query requiring purchase possibility on advert. The last set of pages again represents query requiring discussions and reviews. On the figure there are from the pattern extraction point of view visible clearly readable clusters. However in each cluster there are blank gabs (rows with different rating). These gabs represent that the pages do not correspond to the required query with high probability (there was not extracted patterns we expected). Except of this we can see that patterns on the pages exist in groups. It allows us to work with page profiles during querying and evaluating of pages set. Let us take profile I want to buy via internet shop – we can work with patterns Price Information, Purchasing Possibility, Special Offer, Annuity Selling.

8 Conclusion and Future Work

The crucial aspect of our approach is that we do not need to analyze pages HTML code. Our algorithm is based on analysis of plain text of the page. For

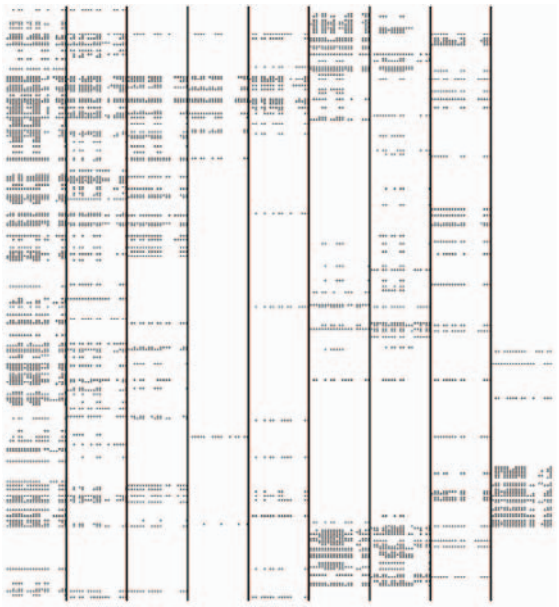


Fig. 4. Sequence of pages sets

page evaluation we dont use any meta-information about page (such as title, hyperlinks, meta-tags and so on). We also confirmed that key characteristics of web patterns are independent of language environment. We tested our method in English and Czech language environment. The only thing we had to do was to change patterns dictionaries.

We can see from our experiments that it is very useful to consider gained data about pattern existence as a metadata stored along with the page. So now we have tools which are able to discover whether the page contains certain pattern (with 80-90% accuracy).

Our approach is not universal. The reason is that the basic assumption for our approach is domain with relatively formed rules of how web pages look like (we do not expect uniformity but we expect some synchronization between users and web pages designers). This implies one interesting backside effect. There are pages which keep mentioned Gestalt principles proximity-similarity- continuity-closure earlier in result set of searched pages.

References

1. Alexander, Ch.: A Pattern Language: Towns, Buildings, Construction. New York. Oxford University Press (1977)
2. Borchers, J. O.: A Pattern Approach to Interaction Design. Proceedings of the DIS 2000 International Conference on Designing Interactive Systems, ACM Press (2000)

3. Cech, E.: Topological Spaces. J. Wiley-Interscience Publ., New York (1966)
4. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufman Publishers (2003)
5. Ciravegna, F., Chapman, S., Dingli, A. and Wilks, Y.: Learning to Harvest Information for the Semantic Web. ESWS 2004, LNCS 3053, pp. 312326, 2004. Springer-Verlag Berlin Heidelberg (2004)
6. Cordón, O., Moya, F. and Zarco, C.: Fuzzy Logic and Multiobjective Evolutionary Algorithms as Soft Computing Tools for Persistent Query Learning, in Text Retrieval Environments. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004), Budapest (Hungary) (2004) 571-576
7. Van Duyne, D. K., Landay J. A. and Hong J. I.: The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Pearson Education (2002)
8. Ferragin, P., and Gulli, A.: A personalized search engine based on Web-snippet hierarchical clustering. In: Proc. of 14th international conference on World Wide Web 2005, Chiba, Japan (2005) 801-810
9. Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
10. Google Web APIs – Home. http://www.google.com/apis/ (May 29, 2005)
11. Graham, I.: A pattern language for web usability. Addison-Wesley (2003)
12. Husek, D., Owais S., Kromer, P., Snasel V. and Neruda, R.: Implementing GP on Optimizing both Boolean and Extended Boolean Queries in IR and Fuzzy IR systems with Respect to the Users Profiles. 2006 IEEE World Congress on Computational Intelligence, CEC (2006) accepted.
13. Jianming Li, L. Z. and Yu, Y.: Learning to generate semantic annotation for domain specific sentences. In Knowledge Markup And Semantic Annotation Workshop in K-CAP 2001 (2001)
14. Karov, Y. and Edelman, S.: Similarity-based Word Sense Disambiguation. Computational Linguistics **24** 1 (1998) 41-59
15. Kiryakov, K., Popov, B., Ognyanoff, D., Manov, D., Kirilov, A. and Goranov, M.: Semantic Annotation, Indexing, and Retrieval. ISWC 2003, LNCS 2870, Springer-Verlag Berlin Heidelberg (2003) 484499
16. Kiyavitskaya, N., Zeni, N., Cordy, J. R., Mich, L. and Mylopoulos, J.: Semantic Annotation as Design Recovery. ISWC 2005, 4th International Semantic Web Conference, Galway, Ireland, submitted (April 2005) 15 pp.
17. Kraft, D. H., Petry, F. E., Buckles, B. P. and Sadasivan, T.: Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In Sanchez, E., Shibata, T., and Zadeh, L.A. (eds.), Genetic Algorithms and Fuzzy Logic Systems, Singapore: World Scientific (1997)
18. Mullet, K. and Sano, D.: Designing visual interfaces: Communication oriented techniques. Englewood Cliffs, NJ. Prentice Hall (1994)
19. Naimpally, S. A. and Warrack, B. D.: Proximity Spaces. Cambridge University Press, Cambridge (1970)
20. O'Reilly, T.: What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (2006, September 15)
21. Pivk, A.: Automatic Ontology Generation from Web Tabular Structures. PhD thesis, University of Maribor (June 2005)
22. Reis, D. C., Golgher, P. B., Silva, A. S. and Laender, A. F.: Automatic web news extraction using tree edit distance. In WWW '04: Proceedings of the 13th

international conference on World Wide Web, New York, NY, USA, ACM Press (2004) 502-511
23. Tidwell, J.: Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media, Inc. (2006)
24. Tijerino, Y. A., Embley, D. W., Lonsdale, D. W., Ding, Y. and Nagy, G.: Towards Ontology Generation from Tables. World Wide Web **8** 3 (2005) 261-285
25. Van Welie M. and van der Veer G. C.: Pattern Languages in Interaction Design: Structure and Organization. Proceedings of Interact '03, Zürich, Switzerland. IOS Press, Amsterdam (2003)
26. Wellhausen, T.: User Interface Design for Searching. A Pattern Language. http://www.tim-wellhausen.de/papers/UIForSearching/UI ForSearching.html (May 29, 2005)
27. Wechsler, K., Baier, J., Nussbaum, M. and Baeza-Yates, R.: Semantic Search in the WWW supported by a Cognitive Model. In Int. Conf. Web-Age Information Management, LNCS, Springer, Dalian, China (July 2004)

Learning Algorithms Based on Regularization*

Petra Kudová

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Praha 8, Czech Republic
petra@cs.cas.cz

Abstract. The problem of supervised learning is a subject of great interest at present. It covers wide range of tasks, such as various classification, prediction, or forecasting problems, i.e. problems that also often arise in semantic web applications. We study one approach to this problem – regularization networks. We introduce composite types of kernel functions, sum kernels and product kernels. On experiments we demonstrate the role of the regularization parameter and kernel function in the regularization network learning, and compare networks with different types of kernel functions.

1 Introduction

The amount of data produced in various areas of human activity is rapidly increasing. At the same time the interest in learning algorithms increases. In many applications we encounter the problem that given a data sample of limited size we have to find a concise description of the data.

In this paper we deal with the problem of *supervised learning*. In this case the data is a sample of input-output patterns (called training sample or training set), thus a concise description of the data is typically a function that can produce the output, given the input. Then the task of learning is to find a deterministic function that maps any input to an output such that the disagreement with future input-output observations is minimized.

An important feature of the learning algorithm is its *generalization ability*. By generalization we mean that the mapping found by the learning algorithm maps correctly not only inputs included in the training set, but also gives correct answers for input points that were not seen before.

In this work we study one particular learning algorithm, called a *regularization network*. The regularization network is derived from regularization theory that is based on the idea of simultaneous minimization of error on the training set and regularization term reflecting our knowledge about a given problem.

In the next section we briefly describe the derivation of regularization network [2, 7, 10, 3, 5]. In section 3 we introduce two types of composite kernels

* The work was partially supported by the Program "Information Society" under project 1ET100300419 and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Appplications".

(see also [6]), namely product kernels and sum kernels. In Section 4 we demonstrate the selected results of our experiments. First, we demonstrate the role of the choice of the regularization parameter and kernel function in regularization network learning. Then we compare the product and sum kernels to the simple Gaussian kernel. Finally, Section 5 summarizes the presented results and outlines the possible future applications in the semantic web.

2 Regularization Networks

Now we will formalize the problem of supervised learning. We are given a set of examples (pairs)

$$\{(\boldsymbol{x}_i, y_i) \in R^d \times R\}_{i=1}^N$$

that was obtained by random sampling of some real function f, generally in the presence of noise. The goal is to recover the function f from data, or find the best estimate of it.

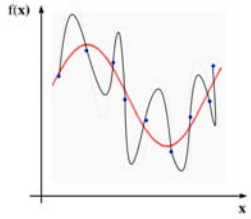


Fig. 1. The problem of learning from examples

Note that it is not necessary that the function exactly interpolates all the given data points, but we need a function with a good *generalization*, that is a function that gives relevant outputs also for the data not included in the training set (see Fig. 1).

It is easy to see that the problem is generally ill-posed. There are many functions interpolating the given data points, but not all of them also exhibit the generalization ability. Therefore it is necessary to consider some a priori knowledge or assumption about the function f. Typically, it is assumed that the function is smooth, does not oscillate too much, etc.

Then one is looking for a function minimizing the functional containing both the data term and smoothness information [9]

$$H[f] = \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2 + \gamma \Phi[f], \tag{1}$$

where Φ is called a *stabilizer* and $\gamma > 0$ is *the regularization parameter* controlling the trade-off between the closeness to data and the smoothness of the solution.

Poggio, Girrosi, and Jones [2] used the stabilizer

$$\Phi[f] = \int_{R^d} d\boldsymbol{s} \frac{|\tilde{f}(\boldsymbol{s})|^2}{\tilde{G}(\boldsymbol{s})}, \tag{2}$$

where $\tilde{f}$ indicates the Fourier transform of f , $\tilde{G}$ is some positive function that goes to zero for $||s|| \to \infty$, i.e. $1/\tilde{G}$ is a high-pass filter.

Under slight assumptions on $\tilde{G}$ it can be shown that the minimum of the functional (2) has the form of linear combination of basis functions G

$$f(x) = \sum_{i=1}^{N} w_i G(\boldsymbol{x} - \boldsymbol{x}_i) + \sum_{\alpha=1}^{k} d_\alpha \psi_\alpha(\boldsymbol{x}), \tag{3}$$

where $\{\psi_\alpha\}_{\alpha=1}^{k}$ is a basis of the k-dimensional null space $\mathbb{N}$ of the functional Φ. Coefficients d_α and w_i depend on the data and satisfy the following linear system:

$$(G + \gamma I)\boldsymbol{w} + \Psi^T \boldsymbol{d} = \boldsymbol{y} \tag{4}$$

$$\Phi \boldsymbol{w} = 0 \tag{5}$$

where I is the identity matrix.

For positive semi-definite function G the null space is empty, for conditionally positive semi-definite function G the basis of the null space is a set of polynomials, however in practical applications the polynomial term is omitted. Then the function f (3) is a linear combination of basis functions G and can be represented by a neural network with one hidden layer. We call such network a *regularization network* (RN).

Poggio and Smale [7] derived the RN using Reproducing Kernel Hilbert Spaces (RKHS).[1] Thanks to this connection to RKHS, we call the basis function of RN a *kernel function*.

Assuming that the value of γ and the type of kernel function K are given in advance, the algorithm performing the RN learning is listed in Algorithm 2.1.

Input: Data set $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N} \subseteq \mathbb{R}^d \times \mathbb{R}$
Output: Regularization Network

1. Set the centers of kernels: $\forall i \in \{1, \ldots, N\} : \boldsymbol{c}_i \leftarrow \boldsymbol{x}_i$
2. Compute the values of weights $w_1, \ldots, w_N$:

$$(\mathbf{K} + \gamma \mathbf{I})\boldsymbol{w} = \boldsymbol{y}, \tag{6}$$

where $\mathbf{I}$ is the identity matrix,
$\mathbf{K_{ij}} = \mathbf{K}(\boldsymbol{x}_\mathrm{i}, \boldsymbol{x}_\mathrm{j})$, and $\boldsymbol{y} = (y_1, \ldots, y_N)$, $\gamma > 0$.

Algorithm 2.1

[1] RKHS was defined by Aronszajn as a Hilbert space of functions with the property that each evaluation functional is bounded (see [1]).

The kernel function K and the regularization parameter γ reflect our prior knowledge of the problem at hand. In fact by their choice we formulate our prior knowledge, i.e. we formulate the problem we are going to solve. Naturally, the improper choice of these parameters, i.e. improper problem definition, may lead to useless solutions.

However, in practice we typically do not have any prior knowledge of the problem. Therefore a framework above the Algorithm 2.1 was created and optimal value for the regularization parameter and optimal kernel function are sought.[2]

3 Construction of composite kernels

The choice of kernel function corresponds to the learning problem formulation and plays therefore a crucial role in RN learning. Optimally, the kernel function should be always chosen according to the given problem, i.e. the given data set.

In practice we often meet data that are heterogenous, i.e. include attributes of different types or qualities, or differ in different regions of the input space. We believe that kernel functions constructed as combinations of simpler kernel functions may better reflect character of such data.

We proposed two types of composite kernels, product kernels and sum kernels [6].

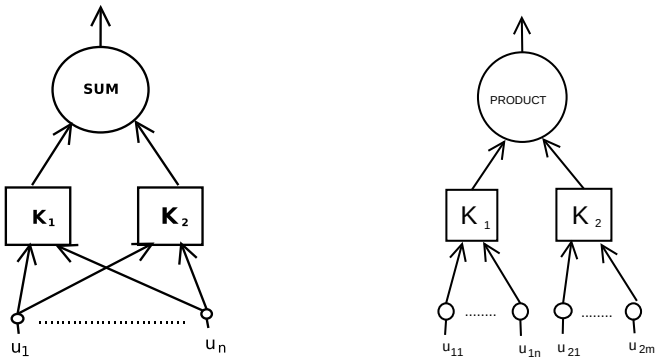


Fig. 2. a) Sum Unit, b) Product Unit.

By a *sum kernel* (see Fig 2a) we mean a unit with n real inputs and one real output. It consists of two positive definite kernel functions $K_1(c, \cdot)$, $K_2(c, \cdot)$, both evaluating the same input vector. The output of the sum unit is computed as the sum $K_1(c, x) + K_2(c, x)$.

By a *product kernel* (see Fig 2b) we mean an unit with $(n + m)$ real inputs and one real output. It consists of two positive definite kernel functions $K_1(c_1, \cdot)$,

[2] Parameters with the lowest cross-validation error are sought. See [4].

$K_2(c_2, \cdot)$, one evaluating the first n inputs and one evaluating the other m inputs. The output of the product unit is computed as the product $K_1(c_1, x_1) \cdot K_2(c_2, x_2)$.

It was shown by Aronszajn [1] that product (resp. sum) of two kernel function is again a kernel function and therefore products (resp. sums) of kernel functions can be used in the Algorithm 2.1 as a kernel function. Product kernels are supposed to be used on data with different types of attributes, since different groups of attributes can be processed by different kernel functions. Sum kernels may improve learning in cases when data differ in different regions in the input space.

4 Experimental results

The goal is to demonstrate the performance of RN with respect to different setups, i.e. choices of γ and kernel functions, and to compare RNs with simple and composite kernel functions.

For the first experiment we have chosen the well known picture of Lenna. The training set contains 2500 samples representing the image of 50×50 pixels. The obtained RN was then used to generate a 100×100 image.

Fig. 3 displays images obtained with RNs using Gaussian kernels of different widths and different regularization parameters. It is easy to see that the choice of these parameters is crucial for the performance of RN. Also the role of the regularization parameter is demonstrated, the higher the regularization parameter the smoother the result. Note, that the wrong choice of kernel parameter (such as too small width of Gaussians in the left column) cannot be cured by the change of the regularization parameter.

For comparison of different kernel functions we have chosen the data collection Proben1 [8]. The error is always normalized

$$E = 100 \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{y}_i - f(\boldsymbol{x}_i)||^2,$$

where N is the number of data samples, $\boldsymbol{y}_i$ is the desired output for the input vector $\boldsymbol{x}_i$, $f(\cdot)$ is the network output and $|| \cdot ||^2$ denotes the Euclidean norm.

For this purpose we have chosen regularization networks with the Gaussian kernels. The product and sum kernels were composed as products of two or three Gaussian functions of different widths, and sums of two Gaussian functions of different widths, respectively. In case of the product kernels, the input attributes were split into two, resp. three input vectors randomly.

The resulting errors achieved by the RNs, PKRNs, and two SKRNs on the tasks from Proben1 are compared in Table 1.

The lowest test error in each row of Table 1 is highlighted. The SKRN achieved the lowest error on 23 tasks, the RN on 13 tasks, and the PKRN on two tasks. However, the errors of all the three networks are comparable.

The SKRN showed an interesting behavior on several data sets. In Table 1 we can see that the training error on the cancer tasks and almost all variants of heart

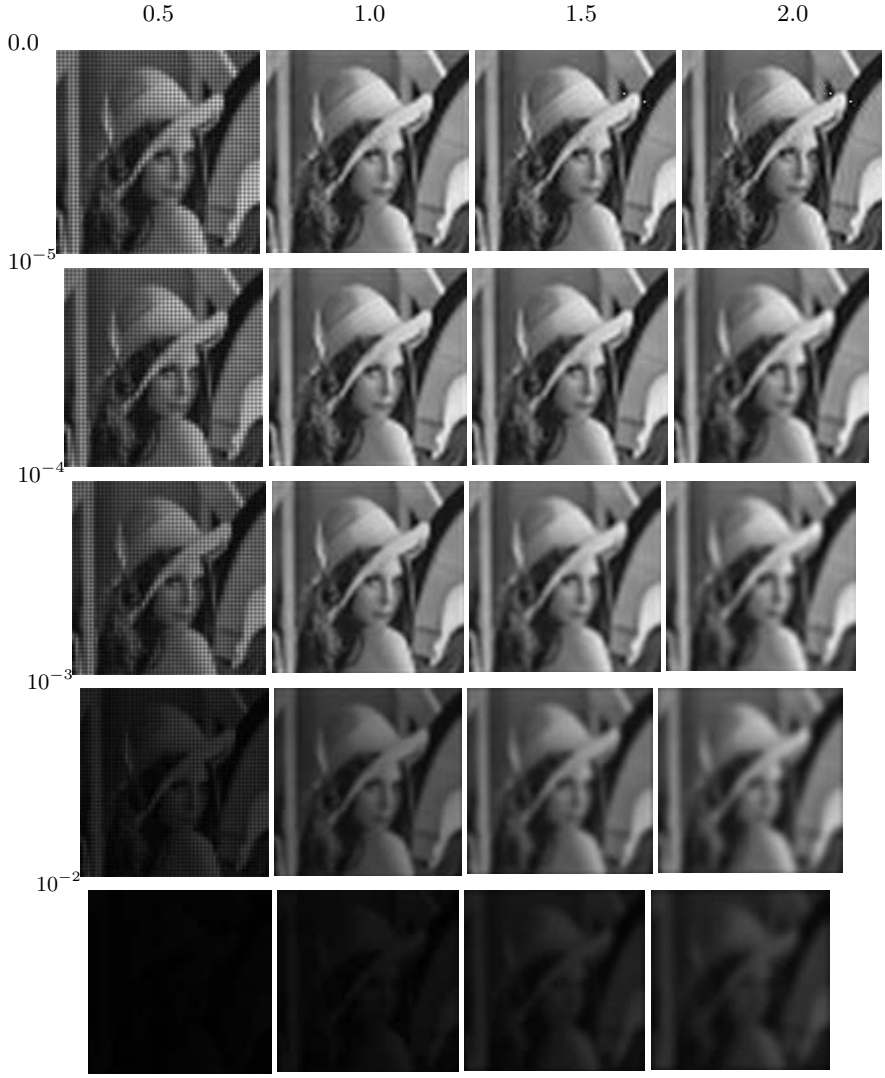


Fig. 3. Images generated by the regularization network learned on the Lenna image (50×50 pixels) using Gaussian kernels with the widths from 0.5 to 2.0 and the regularization parameters from 0.0 to 0.01.

Table 1. Comparisons of errors on the training and test set for the RN with the Gaussian kernels, the SKRN, and the PKRN.

	RN		**SKRN**		**PKRN**	
Task	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
cancer1	2.28	**1.75**	0.00	1.77	2.68	1.81
cancer2	1.86	3.01	0.00	**2.96**	2.07	3.61
cancer3	2.11	2.79	0.00	**2.73**	2.28	2.81
card1	8.75	**10.01**	8.81	10.03	8.90	10.05
card2	7.55	**12.53**	0.00	12.54	8.11	12.55
card3	6.52	12.35	6.55	**12.32**	7.01	12.45
diabetes1	13.97	16.02	14.01	**16.00**	16.44	16.75
diabetes2	14.00	**16.77**	13.78	16.80	15.87	18.14
diabetes3	13.69	16.01	13.69	**15.95**	16.31	16.62
flare1	0.36	0.55	0.35	**0.54**	0.36	**0.54**
flare2	0.42	0.28	0.44	**0.26**	0.42	0.28
flare3	0.38	0.35	0.42	**0.33**	0.40	0.35
glass1	3.37	6.99	2.35	**6.15**	2.64	7.31
glass2	4.32	7.93	1.09	**6.97**	2.55	7.46
glass3	3.96	7.25	3.04	**6.29**	3.31	7.26
heart1	9.61	**13.66**	0.00	13.91	9.56	13.67
heart2	9.33	13.83	0.00	**13.82**	9.43	13.86
heart3	9.23	15.99	0.00	**15.94**	9.15	16.06
hearta1	3.42	4.38	0.00	**4.37**	3.47	4.39
hearta2	3.54	4.07	3.51	**4.06**	3.28	4.29
hearta3	3.44	**4.43**	0.00	4.49	3.40	4.44
heartac1	4.22	**2.76**	0.00	3.26	4.22	**2.76**
heartac2	3.50	3.86	0.00	**3.85**	3.49	3.87
heartac3	3.36	**5.01**	3.36	**5.01**	3.26	5.18
heartc1	9.99	16.07	0.00	**15.69**	10.00	16.08
heartc2	12.70	**6.13**	0.00	6.33	12.37	6.29
heartc3	8.79	12.68	0.00	**12.38**	8.71	12.65
horse1	7.35	**11.90**	0.20	**11.90**	14.25	12.45
horse2	7.97	15.14	2.84	**15.11**	12.24	15.97
horse3	4.26	**13.61**	0.18	14.13	9.63	15.88
soybean1	0.12	**0.66**	0.11	**0.66**	0.13	0.86
soybean2	0.24	**0.50**	0.25	0.53	0.23	0.71
soybean3	0.23	0.58	0.22	**0.57**	0.21	0.78

is almost zero (rounded to zero). In these cases, the regularization parameter chosen by the setup is close to zero, therefore the training error is very low. Still the SKRNs possess the generalization ability.

This is caused by the kernel shape. It consists of two Gaussians, a wider one and very narrow one. The narrow Gaussian emphasizes the strict interpolation of the training sample while the generalization property is assured by the wider Gaussian function.

5 Conclusion

We studied one approach to the problem of supervised learning, the regularization networks.

We discussed the role of the regularization parameter and kernel function, and demonstrated on experiment that their choice is crucial for the regularization network performance.

We proposed two types of composite kernels and demonstrated their applicability on benchmark tasks. Good results were achieved with sum kernels. Sum of two Gaussians is especially suitable for tasks with low level of noise, where they can achieve very low training errors without the loss of generalization.

Our future work includes study of possible use of described method in semantic web applications, such as web pages classification or prediction of user's behavior. Therefore we need to study the use of kernel functions defined on types other than real numbers, i.e. categorical data, objects such as graphs, texts, etc. Composite kernels may be used to combine kernels defined on different domains and enable the application of the algorithm on data sets with heterogenous attributes.

References

1. Aronszajn, N.: Theory of reproducing kernels. Transactions of the AMS 68 (1950) 337–404
2. Girosi, F., Jones, M. and Poggio, T.: Regularization theory and Neural Networks architectures. Neural Computation 2(7) (1995) 219–269
3. Haykin, S.: Neural Networks: a comprehensive foundation. Tom Robins, 2nd edition (1999)
4. Kudová, P.: Learning with kernel based regularization networks. In Information Technologies – Applications and Theory, pages 83–92. Košice, Prrodovedeck fakulta Univerzity Pavla Jozefa Šafárika, 2005
5. Kudová, P. and Neruda, R.: Kernel based learning methods: Regularization networks and RBF networks. In Lawrence N. Winkler J., Niranjan M., editor, Deterministic and Statistical Methods in Machine Learning, Berlin, Springer-Verlag (2005) 124–136
6. Kudová, P. and Šámalová, T.: Sum and product kernel regularization networks. In L. Rutkowski, R. Tadeusiewicz, L.A. Zadeh, and J. Zurada, editors, *Artificial Intelligence and Soft Computing*, Lecture Notes in Artificial Intelligence, Berlin, Springer-Verlag (2006) 56–65
7. Poggio, T. and Smale, S.: The mathematics of learning: Dealing with data. Notices of the AMS 50(5) (2003) 536–544
8. Prechelt, L.: PROBEN1 – a set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Universitaet Karlsruhe 9 (1994)
9. Tikhonov, A. N. and Arsenin, V. Y.: Solutions of Ill-posed Problems. W. H. Winston, Washington, D.C (1977)
10. Wahba, G.: Spline models for observational data. Series in Applied Mathematics 59 (1990)

XSEM – A Conceptual Model for XML Data

Martin Necasky

Department of Software Engineering, Faculty of Mathematics and Physics,
Charles University
martin.necasky@mff.cuni.cz
http://www.necasky.net

Abstract. In this paper we briefly describe a new conceptual model for XML data called *XSEM*. The model is a combination of several approaches in the area of conceptual modeling of XML data. The model divides the process of conceptual modeling of XML data to two levels. On the first level, a designer designs an overall non-hierarchical conceptual schema of a domain. On the second level, he or she derives different hierarchical representations of parts of the overall conceptual schema using transformation operators. These hierarchical representations describe how the data is organized in an XML form.

1 Introduction

Recently, XML is used for an exchange of data between heterogeneous IS, for an internal data representation, and also as a logical database model. As XML gets near to the core of applications, modeling of XML data should become an inseparable part of modeling of application data on the conceptual level.

For example, assume a medical application integrating data about patients from several external sources. The data is stored in an internal database in an internal representation. External applications like various hospital systems, insurance systems, etc. access the data in the database through Internet in an XML form. We can imagine the following typical scenario:

1. physician in a hospital makes a diagnose of a patient
2. to decide the diagnose, the physician needs results of a patient's examination
3. hospital system requests the results from the central medical application
4. medical application exports the results from the internal representation into an XML document and sends it back to the hospital system
5. physician in a hospital diagnoses a disease of a patient
6. hospital system creates an XML document with the diagnose in a required form and sends it to the medical application
7. medical application receives the XML diagnose and transforms it into the internal representation

Fig. 1 shows how applications similar to our example medical application are organized today. Often, there is an overall conceptual schema describing

data stored in a central database and several XML schemes describing structure of XML documents used for communication between the central database and external applications. The figure shows an XML document being send by an external application to the central database. The central database must receive the document and extract the data in the document into its internal representation. There are several problems:

- conceptual schema describing the data in the database can be missing
- XML schema describing the structure of the XML document can be missing
- if the conceptual schema and the XML schema are present, there is no explicit binding between them (i.e. the author of the XML schema designed the schema manually, maybe looking at the conceptual schema, but not deriving the XML schema directly)
- scripts for extracting the data from the XML document and transforming it into the internal representation and vice versa must be programmed and maintained manually

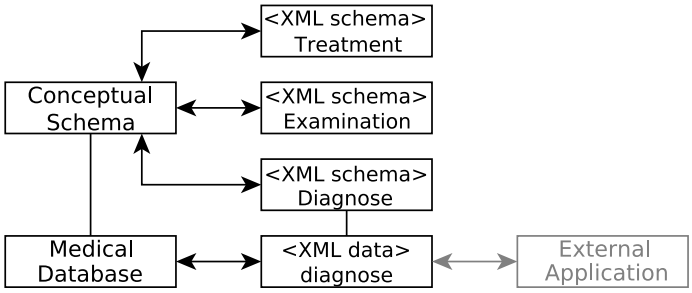


Fig. 1. Motivating Example

The challenge is to eliminate these problems by introducing a conceptual model for XML data. Such a model must allow designers to derive required XML schemes directly from the overall conceptual schema. Even though the XML schemes organize the data into hierarchices, the overall conceptual schema should not be hierarchical. Using this approach, there would be an explicit binding between the XML schemes and the overall conceptual schema which would help to generate scripts transforming data between internal and XML representations.

2 Related Work

If we want to model XML data on the conceptual level, we have to deal with some special features of XML data such as irregular structure, ordering, mixed content, and a hierarchical structure.

There are some approaches, for example ERX [4] or XER [5], extending the E-R model to be suitable for the conceptual modeling of XML data. However, E-R is not hierarchical (there are $M : N$ relationship types and n-ary relationship types) and hierarchical XML schemes must be derived in some way. The problem is that user can not specify how the data should be organized in hierarchical XML. The hierarchical structure is derived automatically without following user requirements.

Another possibility of how to model XML data is to start from a hierarchical structure. This approach may be called *hierarchical approach*. There are conceptual models based on the hierarchical approach, for example ORA-SS [1]. Using this approach we can model hierarchical structures easily. However, a problem with modeling of non-hierarchical structures arises. Moreover, hierarchical schemes are not so transparent as non-hierarchical E-R schemes. A designer must think about the data in hierarchies which is not natural in general.

The problem of the approaches is that it is not possible to design an overall non-hierarchical conceptual schema of the domain and derive several hierarchical organizations of parts of the schema describing required XML documents on the conceptual level as required by the example medical application. Moreover, it is not possible to derive different hierarchical organizations of the same parts of the overall conceptual schema.

In this paper we briefly describe a new conceptual model for XML data called *XSEM* solving the mentioned problems of the existing approaches. In [2], we offer a survey of the conceptual modeling for XML. We propose a detailed list of requirements for conceptual models for XML, describe several conceptual models for XML in a unified formalism, and compare the described models against the requirements. In [3], we describe the XSEM model formally and in detail.

3 Idea

Our idea starts from the central medical database example. There is an internal logical schema describing the structure of data stored in the medical database. Users access the data in the internal representation through XML documents described by several XML schemes. We can comprehend the XML schemes as hierarchical views on parts of the internal logical schema. Each group of users needs different structure of XML documents. These documents can contain the same data, but in different hierarchical organization.

Following the idea, we need to design an overall conceptual schema of a domain and to derive several hierarchical conceptual views from the overall conceptual schema describing required XML schemes. The derivation must be driven by a designer. For a hierarchical view, he or she specifies what part of the overall schema is to be represented in the view and how the components in the part will be organized in a hierarchy.

4 XSEM Model

XSEM is a conceptual model for XML based on the described idea. It divides a conceptual modeling process of XML data to two parts. The first part consits of designing an overall non-hierchical conceptual schema of a domain using a part of XSEM called *XSEM-ER*. The second part consists of designing hierarchical organizations of the structures from the first part using a part of XSEM called *XSEM-H*.

4.1 XSEM-ER

XSEM-ER is an extension of E-R proposed by Chen. It allows to model the special XML features mentioned in the previous text, concretely, irregular structure, ordering, and mixed content. On this level, it is not important how the modeled data are organized in hierarchies. These hierarchical organizations are derived during the second part of the modeling proces.

Fig. 1 shows an example XSEM-ER schema modeling a small part of a medical domain. As in E-R, there are strong and weak entity types and relationship types in XSEM-ER. *Strong entity types* represent stand-alone real world objects. A strong entity type is displayed as a box with a name of the entity type at the box. Fig. 1 shows strong entity types *Hospital*, *Clinic*, *Person*, *Patient*, and *Physician* representing hospitals, separate clinics, general persons divided to patients and physicians.

Except stand-alone real world objects, there are also real world objects which are not stand-alone. For example, when speaking about a department we have to specify also a hospital the department is part of. For modeling such objects we use *weak entity types*. A weak entity type is displayed as a box with inner hexagon and with a name of the entity type in the hexagon. Fig. 1 shows weak entity types *Department*, *Visit*, *History*, *Examination*, and *Diagnose*. For example, *Department* represents departments of hospitals. The strong entity type *Hospital* is called *determinant* of *Department*. It means that for each *Department* instance there must be given a *Hospital* instance. For each determinant of a weak entity type there is a solid arrow going from the weak entity type to the determinant.

Relationship types are used for modeling relationships between entity types. A relationship type is displayed as a hexagon with a name of the relationship type at the hexagon. Fig. 1 shows relationship types *Employ* and *OnBase*.

XSEM-ER adds new modeling constructs called *data node types*, and *outgoing* and *incoming cluster types* for modeling special XML features. A *data node type* is connected with an entity type called *parent* of the data node type. It represents unstructured data values assigned to an entity being an instance of the parent entity type. These values are not attribute values of the entity. They are data values which are mixed with the relationships having the entity as a participant value and the weak entities having the entity as a determinant value. In a graphical representation, a data node type is displayed as an ellipse with a name of the data node type. We display a data type of the data node type only

when necessary. In such a case, the data type is displayed at the name of the node.

For example, assume a patient visiting a physician (represented by the relationship type $Visit$ at Fig. 1). During the visit, the physician writes a description of the course of the visit. The description is not an attribute value of the visit, it is an unstructured text assigned to the visit. Moreover, parts of the description are mixed with examinations and anamneses made during the visit. To model such descriptions we use the data node type $VTxt$ shown at Fig. 1. Example 4.2 shows a possible XML representation of a visit with an unstructured text description mixed with an examination and anamnesis made during the visit.

An *outgoing cluster type* represents a union of entity types. It can be used as a participant of another outgoing cluster type, as a participant of a relationship type, or as a determinant of a weak entity type.

In a graphical representation, an outgoing cluster type is displayed as a circle with an inner label +. It is connected by a solid line with a relationship type or weak entity type it participates in. Each participant of the cluster type is connected by an arrow going from the circle to the participant.

Using outgoing cluster types we can model irregular structure of XML. For example, patients can visit physicians at departments of hospitals and at separate clinics. This is an example of irregular structure we can express in XML. We can use an outgoing cluster type $Department + Clinic$ to model this situation. The cluster type is shown at Fig. 1. Example 4.1 shows a possible XML representation of visits of a patient with a name "John Black". There are two patient's visits in the XML document. First, he visited a physician "Bill White" at a department "Department A" of a hospital "Hospital B". Then he visited a physician "Jack Brown" at a clinic "Clinic C".

```
<patient><name>John Black</name>
  <visit><date>2006-09-12</date>
    <physician><name>Bill White</name></physician>
    <department><name>Department A</name>
      <hospital><name>Hospital B</name></hospital>
    </department>
  </visit>
  <visit><date>2006-10-03</date>
    <physician><name>Jack Brown</name></physician>
    <clinic><name>Clinic C</name></clinic>
  </visit>
</patient>
```

Example 4.1: Irregular structure in XML

Incoming cluster types are used for grouping different relationship types, weak entity types, and data node types having the same participant, determinant, or parent called *parent* of the incoming cluster type. Groups of relationship

types, weak entity types, and data node types are used for modeling mixed content. Moreover, ordering can be specified on such groups.

In a graphical representation, an incoming cluster type is displayed as a circle with an inner label +. It is connected by a solid line with its parent. $P_1, \ldots, P_n$ are connected by an arrow going to the circle.

Together with data note types and ordering constraints (proposed in the next section) we use incoming cluster types for modeling mixed content in XML documents. For example, we can use the incoming cluster type $(Visit, Examination+ Anamnesis + VText)$ at Fig. 1 to model a description of a visit mixed with the examinations and anamnesis made during the visit. Example 4.2 shows an XML representation of such a description. The XML document represents the visit of the patient "John Black" visiting the physician "Bill White" at the department "Department A" of the hospital "Hospital B". Moreover, there is a description of the visit. Unstructured text parts of the description are mixed with a family anamnesis and a laboratory examination made during the visit.

Ordering between parts of the plain text description, anamnesis, and examinations of a visit is important and it must be specified explicitly in the schema. To model such situations, we propose a notion of ordering.

```
<visit><date>2006-09-12</date>
  <patient><name>John Black</name></patient>
  <physician><name>Bill White</name></physician>
  <department><name>Department A</name>
    <hospital><name>Hospital B</name></hospital>
  </department>
  Because of the family anamnesis (<anamnesis>...</anamnesis>)
  there is a suspicion of liver cancer. Consequently, I made
  a laboratory examination (<examination>...</examination>) ...
</visit>
```

Example 4.2: Mixed Content in XML

4.2 Hierarchical Projections

The notion of *hierarchical projections* represents the step between the non-hierarchical XSEM-ER level and the hierarchical XSEM-H level. It is a formalization of a binarization of relationship types and weak entity types. For example, the weak entity type $Visit$ can be represented by the following hierarchy:

- a list of patients
- for each patient the list of patient's visits
- for each patient's visit the visited physician and the department or clinic where the patient visited the physisian

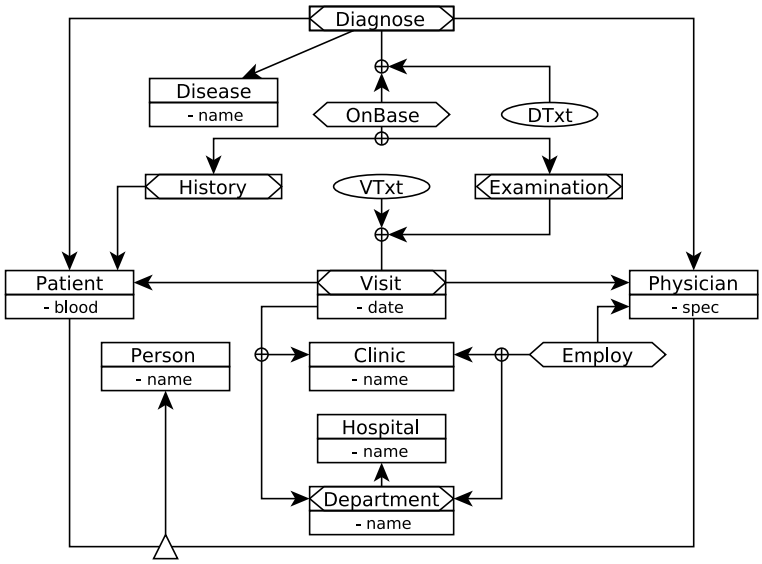


Fig. 1. XSEM-ER Schema

This hierarchy describes the structure of the XML document at Example 4.1.

Hierarchical projections formalize such descriptions of hierarchical organizations of non-hierarchical relationship types and weak entity types. For example, the previous hierarchy is described by the following three hierarchical projections:

$$Visit[Patient \rightarrow Visit] \quad (HP1)$$
$$Visit^{Patient}[Visit \rightarrow Physician] \quad (HP2)$$
$$Visit^{Patient}[Visit \rightarrow Department + Clinic] \quad (HP3)$$

$HP1$ represents a list of patient's visits. $HP2$ represents the visited physician and $HP3$ represent the department or clinic where the patient visited the physician. Another hierarchy is described by the following three hierarchical projections:

$$Visit[Department + Clinic \rightarrow Physician] \quad (HP4)$$
$$Visit^{Department+Clinic}[Physician \rightarrow Patient] \quad (HP5)$$
$$Visit^{Department+Clinic\,Physician}[Patient \rightarrow Visit] \quad (HP6)$$

It represents a hierarchy with a list of departments and clinics. For each department or clinic there is a list of physicians visited by patients at the department or clinic ($HP4$). For each physician, in the context of a department or

clinic, there is a list of patients who visited the physician at the department or clinic ($HP5$). Finally, for each patient in the context of a department or clinic and physician there is a list of patient's visits of the physician at the department or clinic ($HP6$).

For example, the resulting hierarchy describes XML documents at Example 4.3. There is a clinic "Clinic C" and physicians "Bill White" and "Jack Brown". "Bill White" was visited by patients "John Black" and "Joe Green" at the clinic (for each of the patient there is only one such a visit). "Jack Brown" was visited by the patient "John Black" at the clinic (there are two such visits).

```
<clinic><name>Clinic C</name>
  <physician><name>Bill White</name>
    <patient><name>John Black</name>
      <visit><date>2006-10-19</date></visit>
    </patient>
    <patient><name>Joe Green</name>
      <visit><date>2006-09-17</date></visit>
    </patient>
  </physician>
  <physician><name>Jack Brown</name>
    <patient><name>John Black</name>
      <visit><date>2006-10-03</date></visit>
      <visit><date>2006-10-12</date></visit>
    </patient>
  </physician>
</clinic>
```

Example 4.3: XML Described by Hierarchical Projections

4.3 Integrity Constraints

Specification of integrity constraints is an inseparable part of the conceptual modeling process. We extend the notion of *cardinality constraints* known from the E-R model and propose a notion of *ordering constraints*. The cardinality constraints are specified for hierarchical projections. We do not describe these notions here in a more detail.

4.4 XSEM-H

XSEM-H is used for specification of a hierarchical organization of a part of a given XSEM-ER schema using hierarchical projections. It does not add any semantics. An XSEM-H is an oriented graph where nodes represent entity types, relationship types, and data node types from the XSEM-ER schema and edges

represent hierarchical projections of weak entity types and relationship types from the XSEM-ER schema and references.

An XSEM-H schema is derived from an XSEM-ER schema by *transformation operators*. As parameters for the transformation, user supplies entity types, relationship types, and data node types he or she wants to represent in the hierarchical XSEM-H schema and specifies how the components should be organized in a hierarchy. We do not describe the transformation operators in more detail here.

A node is displayed as a box with a name of the represented entity type, relationship type or data node type at the box. An oriented edge is displayed as a solid arrow. Fig. 2 shows an XSEM-H schema where the edges labeled 1, 2, and 3 represent the hierarchical projections $HP1$, $HP2$, and $HP3$. The edges labeled 4, 5, and 6 represent the following hierarchical projections:

$$Department[Department \rightarrow Hospital]$$
$$VTxt[Visit \rightarrow VTxt]$$
$$Examination[Visit \rightarrow Examination]$$

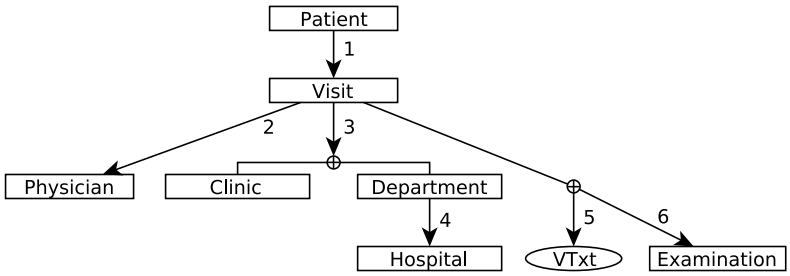


Fig. 2. XSEM-H Schema

Fig. 3 shows an XSEM-H schema where the edges labeled 7, 8, and 9 represent the hierarchical projections $HP4$, $HP5$, and $HP6$.

5 Future Work

We are going to propose algorithms for the translation of XSEM-H schemes to the logical XML level. Beside the grammar based XML schema languages such as XML Schema, we will study the usage of pattern based XML schema languages such as Schematron for the description of more complex integrity constraints. After this, we will create a prototype CASE tool for designing XSEM schemes.

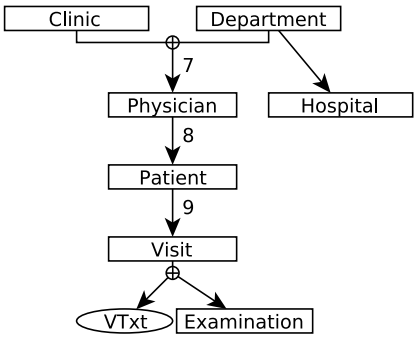


Fig. 3. XSEM-H Schema

Moreover, we will study how to store data described by an XSEM schema into a database. We will study possibilities of a combination of object-relational databases and native XML databases.

References

1. Dobbie, G., Xiaoying, W., Ling, T. W. and Lee, M. L.: ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. TR21/00, Department of Computer Science, National University of Singapore (December 2000)
2. Necasky, M.: Conceptual Modeling for XML: A Survey. Tech. Report No. 2006-3, Dep. of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague (2006) 54 p. `http://www.necasky.net/papers/tr2006.pdf`
3. Necasky, M.: XSEM – A Conceptual Model for XML Data. Accepted for APCCM 2007, Ballarat, Victoria, Australia.
4. Psaila, G.: ERX: A Conceptual Model for XML Documents, In Proceedings of the 2000 ACM Symposium on Applied Computing, Como, Italy (March 2000) 898-903
5. Sengupta, A., Mohan, S. and Doshi, R.: XER – Extensible Entity Relationship Modeling, *in* Proceedings of the XML 2003 Conference, Philadelphia, USA (December 2003) 140-154

Model of Preferences for the Relational Data Model*

Radim Nedbal

Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
radned@seznam.cz

Abstract. The aim of the paper is to present a novel, general approach to preference modelling in the framework of the relational data model. The preferences are defined between sets of relational instances, which is a nontrivial generalization of the approach aiming at incorporating ordered attribute domains into the relational data model. The main goals are as follows: an effective representation of information representable by a partial order, an intuitive preference construction and its processing throughout the query execution plan, and a suitable data structure to support it all.

1 Related Work

Lacroix and Lavency [1] originated the study of preference queries. They proposed an extension of the relational calculus in which preferences for tuples satisfying given logical conditions can be expressed.

Kießling et al. [2] and Chomicki et al. [3] proposed independently a similar framework based on a formal language for formulating preference relations. The embedding (called *Best Match Only* – BMO and *WinNow* – WN respectively) into relational query languages they use is based on the operator returning the most preferred tuples.

Börzsönyi et al. [4] introduced the *skyline operator* and described several evaluation methods for this operator. Skyline is a special case of WN and BMO.

Argawal and Wimmers [5] use quantitative preferences (scoring functions) in queries and focus on the issues arising in combining such preferences. Hristidis et al. [6] explore in this context the problems of efficient query processing using materialized views.

A more general approach to preference modelling is presented in [7]. A declarative query interface for Web repositories that supports complex expressive Web queries is defined.

* This work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization" and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

In [8], actual values of an arbitrary attribute are allowed to be partially ordered to represent user preferences. Accordingly, relational algebra operations, aggregation functions and arithmetic are redefined.

A viewpoint is presented in [9], where the relational data model is extended to incorporate partial orderings into data domains. Within the extended model, the partially ordered relational algebra (the PORA) is defined by allowing the ordering predicate to be used in formulae of the selection operation.

A comprehensive work on partial order in databases is [10]. It presents the partially ordered sets as the basic construct for modelling data. Collection of algebraic operations for manipulating ordered sets is investigated, and their implementation based on the use of realizers as a data structure is presented. An algorithm for generating realizers for arbitrary finite partial orders is provided.

In the context of financial and statistical applications, systems such as SEQUIN [11], SRQL [12], and more recently Aquery [13,**?**] have proposed SQL extensions to incorporate ordering.

2 Introduction into the Problem

Let us start with the following illustrating and motivating example.

Example 1. Consider a query whose output should be a group of people working for a company "R". A requirement is no master – slave relationship between any two members of the group. Furthermore, the preference for superior employees in the master – slave hierarchy should be taken into account.

The master – slave hierarchy is a piece of information we have about the knowledge domain. It is represented by means of a partial order relation. The requirement for no master – slave relationship between any two members of the group is a preference representable by a partial order. Also, the preference for superior employees in the master – slave hierarchy can be represented by means of a partial order. The overall preference will be expressed by means of a preference combinator reflecting the relationship between the above preferences. □

To sum up, the following problems arise.

- How can be a piece of information representable by a partial order incorporated into the relational data model so that it can be manipulated effectively?
- How can be a preference expressed and processed throughout the query execution plan?
- What data structure is suitable to support an effective solution to the above problems?

3 The Proposed Solution

The mathematical theory includes partial orders over infinite sets, but we restrict ourself to partial orders over finite sets, in the same way that the relational data model restricts itself to finite relations (in particular, this means that the maxima and minima are always well-defined).

3.1 Preference Modelling

Broadly speaking, a preference of a state can be understood as its proximity to the optimal state. A state can be described by its model. A derived state can be described by the model of the state it is derived from and a set of deriving logical formulas. As the model is a subset of a Herband base, it can be viewed as a theory. This theory in conjunction with a set of deriving formulas has a minimal model that describes the derived state in question. Then the proximity of a state derived from a suboptimal one to a state derived from the optimal one by the same set of formulas equates to the proximity of the suboptimal state to the optimal one. The higher proximity can be assigned to a state, the higher preference of the state can be concluded.

Note that, generally, a state might by a derivative from n other states with various preferences. Then the derived state is assigned $n-$tuple of proximities of the corresponding states.[1]

According to the above concept, a relation corresponds to a state, a relational instance to its model, and a query expression to a deriving logical formula. For instance, a selection condition is used to derive a state that we are interested in.

3.2 Preference Engineering

The goal is to provide intuitive and convenient ways to inductively construct a preference $\mathrm{P} = (R, \sqsubseteq_R)$. The inductive construction is based on the notion of *preference term*.

Definition 1 (Preference term). *Given preference terms* P_1 *and* P_2*, P is a preference term iff P is one of the following: base preference, subset preference (*$\mathrm{P} := {\mathrm{P}_1}^{\subseteq}$*), dual preference (*$\mathrm{P} := \mathrm{P}_1^{\partial}$*), complex preference gained by applying one of the following preference combinators: Pareto accumulation (*$\mathrm{P} := \mathrm{P}_1 \otimes \mathrm{P}_2$*) or prioritized accumulation (*$\mathrm{P} := \mathrm{P}_1 \& \mathrm{P}_2$*)*

Base Preferences. The set of Base preference constructors is extensible, if required by the application domain. Commonly useful constructors include the following:POS(SET) specifying a SET of values that should be preferred, NEG(SET) specifying a SET of values that should be avoided,POS\POS($\mathrm{SET}_1, \mathrm{SET}_2$) specifying two sets of values that should by preferred (values of the first set SET_1 should be preferred also to values of the second set SET_2), POS\NEG($\mathrm{SET}_1, \mathrm{SET}_2$) specifying two sets of values (values from the first set SET_1 should be preferred, and values from the second set SET_2 should be avoided), EXPLICIT represented explicitly by a realizer, and PF (preference formula), a first order formula defining a preference relation in the standard sense, namely

$$S \sqsupseteq_{\mathrm{PF}} T \iff \mathrm{PF}(S, T)$$

[1] Thus in case of a lattice ordering of proximities, the preference of a state can be determined by the least upper bound of all the proximities that can be assigned to the state.

Complex Preferences. It is possible to construct more complex preferences by means of *preference combinators*. They can combine preferences coming from one or several parties.

Definition 2 (Pareto preference: $\mathbf{P_1 \otimes P_2}$). *Both preferences* P_1 *and* P_2 *are considered to be equally important:*

$$R \sqsubseteq_{\mathrm{P}_1 \otimes \mathrm{P}_2} S \iff R \sqsubseteq_{\mathrm{P}_1} S \wedge R \sqsubseteq_{\mathrm{P}_2} S$$

Definition 3 (Prioritized preference: $\mathbf{P_1 \& P_2}$). P_1 *is considered more important than* P_2*, which is respected only when* P_1 *does not mind:*

$$R \sqsubseteq_{\mathrm{P}_1 \& \mathrm{P}_2} S \iff R \sqsubseteq_{\mathrm{P}_1} S \vee (R = S \wedge R \sqsubseteq_{\mathrm{P}_2} S)$$

Example 2. Recall the Example 1. We defined the preference P by means of prioritized preference combinator:

$$\mathrm{P} = \mathrm{P}_1 \ \& \ \mathrm{P}_2,$$

where P_1 and P_2 were base preferences defined as a NEG-preference $\mathrm{P}_1 := \mathrm{NEG}$ (NEG-set) and a PF preference $\mathrm{P}_2 := \mathrm{PF}(S, T)$ respectively. The NEG-set was defined by means of a logical formula:

$$\text{NEG-set} = \{R' \subseteq R | \exists r_1, r_2 \in R'(r_1 \sqsubseteq_R r_2)\},$$

and preference formula PF was defined by means of preference logical formula:

$$\mathrm{PF}(S, T) = \forall t \in T(t \in S \vee \exists s \in S(t \sqsubseteq_R s))$$

Note that both logical formulas contain a predicate symbol $\sqsubseteq_R$ interpreted by means of partial order representing a piece of knowledge we have about the knowledge domain. □

3.3 Partial Order Model

The base set elements of a partial order can be arbitrary objects. In the proposed model, the objects will be sets. The aim is to model order on a set of relational instances, which is a nontrivial generalization of the approach aiming at incorporating ordered attribute domains into the relational data model.

Algebra for Partial Orders. The aim is to manipulate partial orders as object in an algebra. The algebra should be closed, that is, the result of applying any operator to any partial order should be a new partial order.

The partial order algebra should include:

- **Predicates** determining satisfaction of a given property, among others: (binary) **containment** and **equality**.
- **Operators**, among others: **selection**, **duplicate elimination**, **maxima** and **minima**, extracting the maximal and minimal elements respectively, **up-tail** and **down-tail**, extracting the suborder that result from removing the minima and maxima respectively.
- **Operations**, among others (unary) **projection** and (binary) **cartesian product**, **intersection**, **union**, **difference**.

Partial Order Implementation. An implementation of the partial order model includes both a data structure to hold the partial orders, and algorithms for the algebra operations. A partial order can be represented by means of *realizer*.

Definition 4 (Realizer). *Realizer is a set of linear extensions of the partial order such that for any incomparable elements $a \parallel b$ there exists a linear extension $<_i$ in which $a <_i b$ and a linear extension $<_j$ in which $b <_j a$.*

Example 3 (Realizer). Figure 1 shows a partial order and its realizer, consisting of two linear extensions.

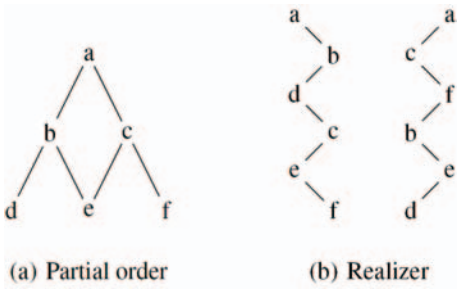


Fig. 1. Partial order and its realizer

Remark 1. A partial order realizer is a collection of sequences, each of which contains all the elements of the partial order base set. For this reason, an array is a suitable data structure to hold the realizer.

3.4 Preference Model Implementation

The aim of this subsection is to present the data structure to effectively implement the proposed partial order model. As the semantics of partial order defining preferences is vertically oriented, a data structure with a column oriented semantics in which variables are bound to arrays, not tuples, should be employed.

A promising solution is *arrable*, for array-table, which is an ordered data structure. Informally, it is a collection of named arrays whose values may be arrays themselves. Essentially, arrable is a table organized by columns.

Definition 5 (Arrable). *Let $\mathscr{T}$ be a set of types corresponding to a basic type or to a one-dimensional array of basic type elements. Let A be a finite array of elements of a type from $\mathscr{T}$. The cardinality of A is the number of elements in A's first dimension. $A[k]$ is the $k-th$ element, and k is said to be an index or position in A. An arrable is a collection of named arrays $A_1, \ldots, A_n$, each being of a type from $\mathscr{T}$, that have the same cardinality.*

Each relational algebra operator takes array-typed expressions as arguments and an according execution model should be provided.

The most common execution model is called *iterator-based*. It is however cache inefficient due to loading unnecessary columns' data to perform an operation. For instance, even if the projection involves only one attribute, it brings the entire tuple to occupy cache space.

A contrasting execution model is the *column-oriented* one found in [15]. The essence of the model consists in that instead of handling a row at a time, data is vertically partitioned in columns that are manipulated as units. For instance, selection looks at the columns involved in the selection condition and returns the IDs of rows satisfying the condition. It is not necessary to materialize result of each operation. Instead, the reference to the resulting data can be passed on to the next operation.

4 Conclusion

A generalized model of preferences for the relational data model has been proposed. It should present a basis for semantically rich, easy to handle and flexible preference model aiming at deep personalization of database queries.

The biggest difference as compared with other mentioned approaches consists in expressing preferences by means of a partial order on a set of relational instances, while the other known approaches express preferences by means of a partial order on a set of relation tuples or on attribute domains. As a relation tuple is a special case of a corresponding relation instance, the presented approach is strictly more general.

Another key difference is a use of realizers to encode preferences. Realizers present a means to process preference-related information throughout the whole query execution plan effectively. Therefore no preference-related information is lost as against the known approaches aiming at only the best preference-matching results. Thus a bigger expressivity is reached.

Realizers are employed also to effectively hold all kinds of information expressible by partial order, and corresponding algebra is proposed to manipulate them. This is the prerequisite for allowing a selection condition to contain predicates interpreted by partial orders. It has been shown [9] that this increases the expressivity of the resulting *partial order relational algebra*.

Also, a data structure, called arrable, first introduced in [15], is proposed to implement the proposed database model. It is a column oriented data structure, suitable for data representing information with "column-oriented semantics". Queries over arrables are executed by breaking them down into a sequence of array primitives, which are often small enough to fit into the main memory.

It can be shown that, in the framework of modelling preferences by means of partial order on relation instances, the *partialy ordered relational algebra* operations can be well-defined in the sense that all the known identities hold. Specifically, associativity and commutativity of the union, product, selection, and projection operators are retained.

Also, other relational operators (intersect, join, and divide), also, retain the usual properties of their classical relational counterparts.

These results are promising towards the query optimization issues, which present many open problems. In particular, evaluation and optimization of preference queries, including cost-base optimization, need to be addressed. Besides, there is an open problem of exploiting the combination of the iterator-based and the column-oriented execution model. The crux probably consists in stepwise realizer construction, modelling stepwise adding pieces of preference-related information, and the related impact against the partially ordered relational algebra operations.

References

1. Lacroix, M., Lavency, P.: Preferences; Putting More Knowledge into Queries. In Stocker, P.M., Kent, W., Hammersley, P., eds.: VLDB, Morgan Kaufmann (1987) 217–225
2. Kießling, W.: Foundations of Preferences in Database Systems. In: Proceedings of the 28th VLDB Conference, Hong Kong, China (2002) 311–322
3. Chomicki, J.: Preference Formulas in Relational Queries. ACM Trans. Database Syst. **28** 4 (2003) 427–466
4. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2001) 421–430
5. Agrawal, R., Wimmers, E.: A Framework for Expressing and Combining Preferences. In Chen, W., Naughton, J.F., Bernstein, P.A., eds.: SIGMOD Conference, ACM (2000) 297–306
6. Hristidis, V., Koudas, N., Papakonstantinou, Y.: Prefer: a system for the efficient execution of multi-parametric ranked queries. In: SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2001) 259–270
7. Raghavan, S., Garcia-Molina, H.: Complex queries over web repositories (2003)
8. Nedbal, R.: Relational Databases with Ordered Relations. Logic Journal of the IGPL **13** 5 (2005) 587–597
9. Ng, W.: An Extension of the Relational Data Model to Incorporate Ordered Domains. ACM Transactions on Database Systems **26** 3 (2001) 344–383
10. Raymond, D.R.: Partial-order databases. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada (1996) Adviser-W. M. Tompa.
11. Seshadri, P., Livny, M., Ramakrishnan, R.: The design and implementation of a sequence database system. In: VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1996) 99–110
12. Ramakrishnan, R., Donjerkovic, D., Ranganathan, A., Beyer, K.S., Krishnaprasad, M.: Srql: Sorted relational query language. In: SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management, Washington, DC, USA, IEEE Computer Society (1998) 84–95
13. Lerner, A., Shasha, D.: Aquery: Query language for ordered data, optimization techniques, and experiments. In: 29th International Conference on Very Large Data Bases (VLDB'03), Berlin, Germany, Morgan Kaufmann Publishers (2003) 345–356

14. Lerner, A.: Querying Ordered Databases with AQuery. PhD thesis, ENST-Paris, France (2003)
15. Boncz, P.A.: Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications. Ph.d. thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands (2002)

Hybrid Methods of Computational Intelligence and Software Agents*

Roman Neruda

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 18207 Prague 8, Czech Republic
roman@cs.cas.cz

Abstract. In this paper we present an approach where a hybrid computational model is represented as a set of communicating agents composing a multi-agent system. A general concept of representation of connected groups of agents is introduced and utilized for automatic building of schemes to solve a given computational task. We propose a combination of an evolutionary algorithm and a formal logic resolution system which is able to generate and verify new schemes. Furthermore, the adaptive cooperation support of individual computational agents is described, which improves their efficiency in time. These features are implemented within a software system and demonstrated on several examples.

1 Introduction

Hybrid models including combinations of artificial intelligence methods, such as neural networks, genetic algorithms (GA) and fuzzy logic controllers, have shown to be a promising and extensively studied research area [1]. They have demonstrated better performance over individual methods in many real-world tasks. The disadvantages are their bigger complexity and the need to manually set them up and tune various parameters. Also, there are not many software packages that provide a large collection of individual computational methods, as well as the possibility to connect them into hybrid schemes in various ways.

The use of distributed Multi-Agent Systems (MAS) instead of monolithic programs has become a popular topic both in research and application development. Autonomous agents are small self-contained programs that can solve simple problems in a well-defined domain [2]. In order to solve complex problems, agents have to collaborate, forming Multi-Agent Systems (MAS). A key issue in MAS research is how to generate MAS configurations that solve a given problem [3]. In most systems, an intelligent (human) user is required to set up the system configuration. Developing algorithms for automatic configuration of Multi-Agent Systems is one of major challenges for research in computational agents area.

* This work has been supported by the the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization".

We have designed a distributed multi-agent system [4] called Bang 3 [5] that provides a support for an easy creation of hybrid AI models by means of autonomous software agents [6]. *Bang* as a software system has been designed to provide general MAS functionality, optimized for the typical usage area, which is the computational intelligence modeling. Technically, *Bang* consists of the infrastructure layer and agents. The infrastructure is typically a set of programs (called airports) running in a homogeneous networked environment (such as a cluster of workstations), and connected via TCP/IP. The infrastructure provides the basic functionality for agent life cycle, communication via messages, persistence, and migration.

Agents on one machine can either possess individual processes, or share a common process. Since the communication in the computational intelligence area usually employs large volumes of data, the message delivery process has been optimized to employ compression, caching, and usage of shared memory instead of XML streams whenever possible. The whole system has been programmed in C++, mainly in order to seamlessly cooperate with several libraries for scientific computation.

2 Modelling computational methods

Among agents, there is the main group of computational agents realizing AI methods, such as several neural networks, evolutionary algorithms, and fuzzy logic controllers. There are also more 'traditional' agents such as scheduler, directory services, ontology services, etc. A special set of agents and modules provides the functionality that is described in this paper — the MAS schemes representation and autonomous behavior support. These can be seen as a higher level layer of abstraction, and are by no means necessary for simpler usage of the system. For more detailed overview of the system, cf. [5, 7].

Fig. 1 shows an example of the most simple computational MAS in *Bang* which consists only of the computational agent, data and a task manager (which can be a user interacting via GUI, or more complicated agent performing series of experiments over a cluster of workstations).

A more typical computational MAS configuration is shown on Fig. 2. There are two more complicated computational agents, the RBF neural network (RBF) and the Evolutionary algorithm (EA) agent, that cooperate with each other within a computational MAS. Each of these two agents can itself be seen as a MAS employing several simpler agents to solve a given task. In the case of the RBF network, typically, an unsupervised learning (vector quantization), and a supervised learning (gradient, matrix inverse) agent is needed. The evolutionary algorithm agent makes use of fitness (shaper) and probabilities manager (tuner). The cooperation of RBF and EA is more intricate and takes place via the fitness and chromozome agents.

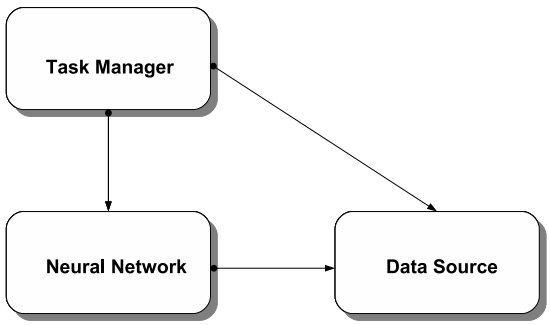


Fig. 1. Example of a small computational MAS consisting of a Task Manager agent, Data Source agent, and a computational agent (Multilayer Perceptron).

3 MAS Formal Description

Bang agents communicate via messages and triggers. Messages are XML documents send by an agent to another agent. Triggers are XML patterns with an associated function. When an agent receives a message matching the XML pattern of one of its triggers, the associated function is executed. In order to identify the receiver of a message, the sending agent needs the message itself and a link to the receiving agent. A conversation between two agents usually consists of a number of messages. In order to abstract from the actual messages, we subsume all these messages under a *message type*. The set of message types understood by an agent is called its *interface*. For outgoing messages, each link of an agent is associated with a message type. Via this link, only messages of the given type are sent. We call a link with its associated message type a *gate*.

Agent A can be connected to agent B via gate G if the message type of G is in the list of interfaces of agent B. Note that one output gate sends messages of one type only, whereas one agent can receive different types of messages. This is a very natural concept: When an agent sends a message to some other agent via a gate, it assigns a specific role to the other agent, e.g. being a supplier of training data. On the receiving side, the receiving agent usually should understand a number of different types of messages, because it may have different roles for different agents. An *agent class* is defined by an interface, a set of message types, a set of gates, and a set of types. An *agent* is an instance of an agent class. It is defined by its name and its class.

Multi-Agent Systems are assembles of agents (for now, only static aspects of agents are modeled). Therefore, a Multi-Agent System can be described by three elements: The set of agents in the MAS, the connections between these agents, and the characteristics of the MAS. The characteristics (constraints) of the MAS are the starting point of logical reasoning: In *MAS checking* the logical reasoner

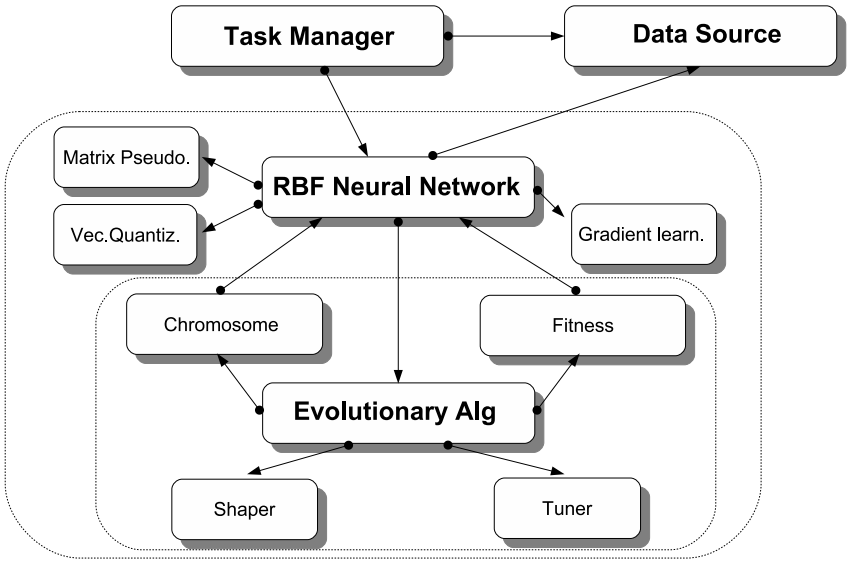


Fig. 2. Example of a more complicated computational MAS consisting of a Task Manager agent, Data Source agent, and a suite of cooperating computational agents (an RBF network agent and Evolutionary algorithm agent with necessary additional agents).

deduces if the MAS fulfills the constraints. In *MAS generation*, it creates a MAS that fulfills the constraints, starting with a partial MAS.

The above described concepts are implemented within the *Bang* software system as the BOA agent. This agent works with ontological description files of the two kinds: the Description Logics description of agent hierarchies, their gates, interfaces and message types, and the Prolog clauses describing more complicated properties and concepts, such as the form of computational MAS, or the notion of trust. Descriptions of the above shown — and similar — MASes are generated by the BOA agent and then sent to the MAS manager agent, which is able to take care of physical creation of the whole system. This includes creating suitable agents (either new ones, or reusing free existing ones, or even finding suitable ones by means of ontology services), linking their gates and interfaces, sending them appropriate initialization messages, etc. This is typically followed by an (automated) trial and evaluation of the computational MAS on a particular data set.

The following paragraphs show two examples for logical descriptions of MAS. It should be noted that these MAS types can be combined, i.e. it is possible to query for trusted, computational MAS.

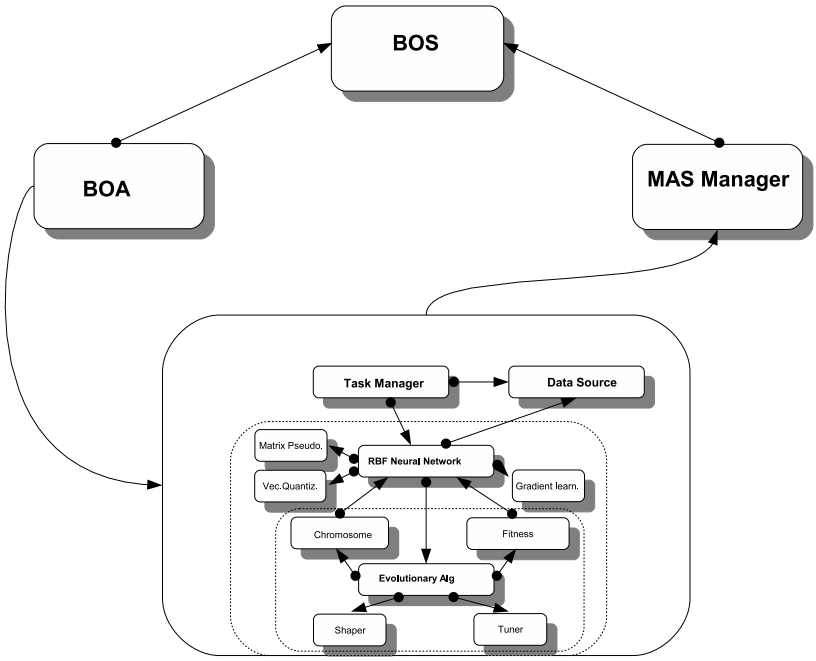


Fig. 3. The BOA agent generates a MAS configuration description and sends it to the MAS manager agent, which takes care of MAS creation and run. They both query the BOS ontology services agent.

Computational MAS A computational MAS can be defined as a MAS with a task manager, a computational agent and a data source agent which are connected:

comp_MAS(MAS) ←
type(CAC, computational_agent)∧
instance(CA, CAC)∧
has_agent(MAS, CA)∧
type(DSC, data_source)∧
instance(DS, DSC)∧
has_agent(MAS, DS)∧
connection(CA, DS, G)∧
type(TMC, task_manager)∧
instance(TMC, TM)∧
has_agent(MAS, TM)∧
connection(TM, CA, GC)∧
connection(TM, GC, GD)

Trusted MAS A MAS is trusted if all of its agents are trusted. These examples use the Prolog predicate `findall`. `findall` returns a list of all instances of a variable for which a predicate is true. In the definition of predicate `all_trusted` the usual Prolog syntax for recursive definitions is used.

```
trusted_MAS(MAS) ←
    findall(X, has_agent(MAS,X), A))∧
    all_trusted(A)
all_trusted([]) ← true
all_trusted([F|R]) ←
    instance(F,FC)∧
    type(FC, trusted)
```

4 Evolutionary Algorithm

The proposed evolutionary algorithm operates on schemes definitions in order to find a suitable scheme solving a specified problem. The genetic algorithm has three inputs: First, the number and the types of inputs and outputs of the scheme. Second, the *training set*, which is a set of prototypical inputs and the corresponding desired outputs, it is used to compute the fitness of a particular solution. And third, the list of types of agents available for being used in the scheme. Note that the evolutionary algorithm uses the agents logical description and reasoning component (described above) in order to produce only such schemes that satisfy given constrains.

We supply three operators that would operate on graphs representing schemes: *random scheme creation*, *mutation* and *crossover*. The aim of the first one is to create a random scheme. This operator is used when creating the first (random) generation. The diversity of the schemes that are generated is the most important feature the generated schemes should have. The 'quality' of the scheme (that means whether the scheme computes the desired function or not) is insignificant at that moment, it is a task of other parts of the genetic algorithm to assure this. The algorithm for random scheme creation works incrementally. In each step one building block is added to the scheme being created. In the beginning, the most emphasis is put on the randomness. Later the building blocks are selected more in fashion so it would create the scheme with the desired number and types of gates (so the process converges to the desired type of function).

The goal of the crossover operator is to create offsprings from two parents. The crossover operator proposed for scheme generation creates one offspring. The operator horizontally divides the mother and the father, takes the first part from father's scheme, and the second from mother's one. The crossover is illustrated in Fig. 4. There are two types of mutation, one randomly changes a node in the graph, and the other swaps agent connections. The latter mutation finds two links in the scheme of the same type (at random) and then switches their destinations (cf. Fig. 5).

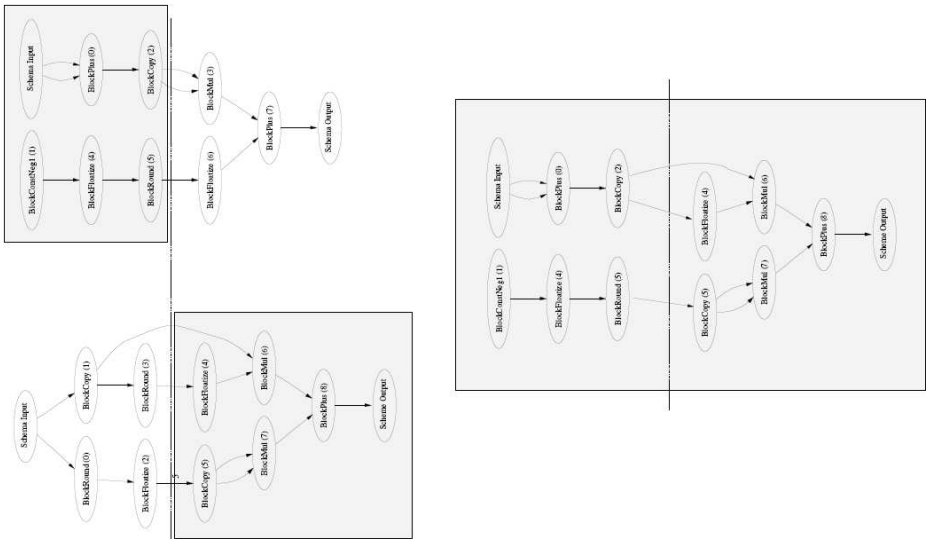


Fig. 4. Crossover of two schemes: The mother and father are horizontally divided and the offspring becomes a mixture of both

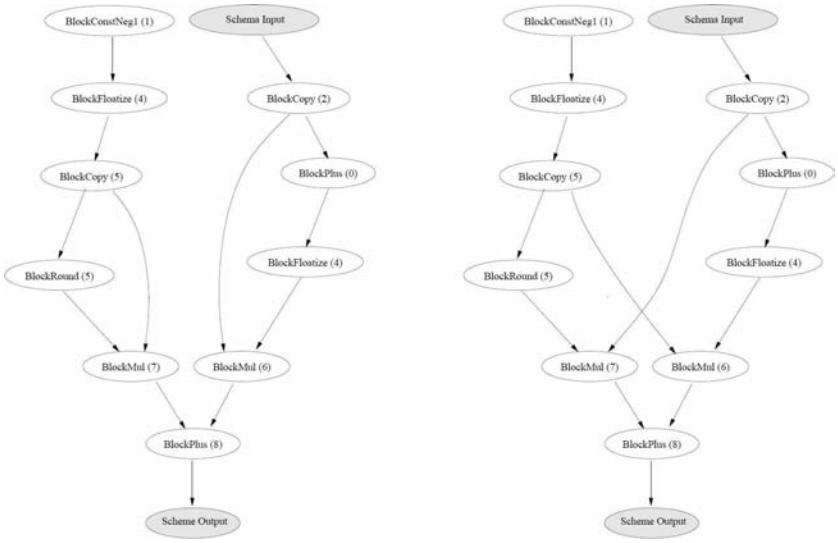


Fig. 5. Mutation of links in a scheme: The destination of two links are switched

5 Autonomous Behavior Support

In order to act autonomously, an agent should be able to cope with three different kind of problems [8]: cooperation of agents, a computation processing support, and an optimization of the partner choice.

Cooperation of agents: An intelligent agent should be able to answer the questions about its willingness to participate with particular agent or on a particular task. The following subproblems follow: (1) deciding whether two agents are able to cooperate, (2) evaluating the agents (according to reliability, speed, availability, etc.), (3) reasoning about its own state of affairs (state of an agent, load, etc.), (4) reasoning about tasks (identification of a task, distinguishing task types, etc.).

Computations processing: The agent should be able to recognize what it can solve and whether it is good at it, to decide whether it should persist in the started task, and whether it should wait for the result of task assigned to another agent. This implies the following new subproblems: (1) learning (remembering) tasks the agent has computed in the past (we use the principles of case-based learning and reasoning — see [9], [10] — to remember task cases), (2) monitoring and evaluation of task parameters (duration, progress, count, etc.), (3) evaluating tasks according to different criteria (duration, error, etc.).

Optimization of the partner choice: An intelligent agent should be able to distinguish good partners from unsuitable ones. The resulting subproblems follow: (1) recognizing a suitable (admissible) partner for a particular task, (2) increasing the quality of an evaluation with growing experience.

So, the architecture must support *reasoning*, *descriptions* of agents and tasks (we use ontologies in descriptions logics – see, e.g., [11]), *monitoring* and *evaluation* of various parameters, and *learning*. The architecture is organized into layers. Its logic is similar to the vertically-layered architecture with one-pass control (see [4, p. 36]). The lowest layer takes perceptual inputs from the environment, while the topmost layer is responsible for the execution of actions.

The architecture consists of four layers (see Fig. 6): the *monitors* layer, the *evaluators modeling* layer, the layer for *decision support*, and the *behavior generation* layer. All layers are influenced by global *preferences*. *Global preferences* allow us to model different flavors of an agent's behavior, namely, we can set an agent's pro-activity regime, its cooperation regime and its approach to reconsideration. *The monitors layer* interfaces directly with the environment. It works in a purely reactive way. It consists of rules of the form $condition \longrightarrow action$. *Evaluators modeling layer* is used to model more aggregate concepts on top of already defined concepts (either monitors or other evaluators). *Decision support layer* enables an agent to solve concrete problems. *Behavior generation layer* generates appropriate actions that the agent should perform, and thus controls the agent's behavior. The mechanisms for action generation and selection are provided by the BDI model (see [4, pages 55–61]).

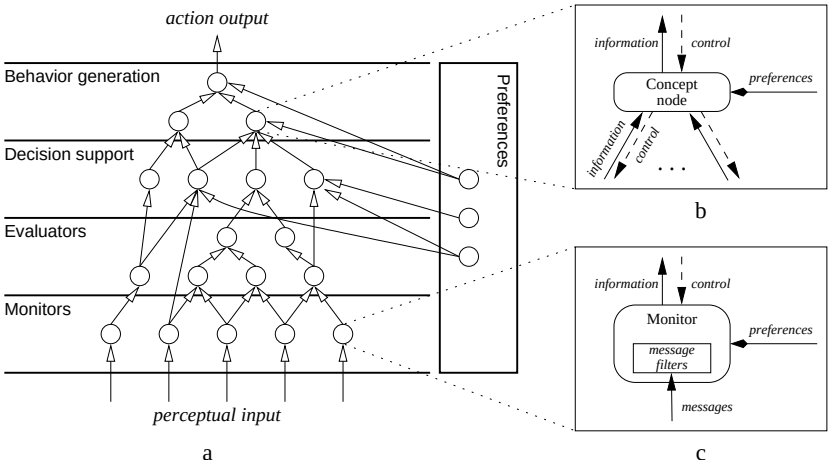


Fig. 6. Architecture — network of concepts (a); Concept node (b); Monitor (c)

6 Experiments

Here we show two experiments demonstrating the functionality of the above described approach. They illustrate the emergence of hybrid solution to a symbolic regression problem (experiment 1), and the improvement of time complexity of neural network learning with the autonomous behavior support (experiment 2).

The training sets used for the first experiment represented various polynomials. The genetic algorithm was generating the schemes containing the following agents representing arithmetical operations: *Plus* (performs the addition on floats), *Mul* (performs the multiplication on floats), *Copy* (copies the only input (float) to two float outputs), *Round* (rounds the incoming float to the integer) and finally *Floatize* (converts the int input to the float). The computation is relatively time demanding, one generation typically takes several seconds on a 2GHz Linux machine.

The results of the experiments depended on the complexity of the desired functions. The functions, that the genetic algorithm learned well and quite quickly were functions like $x^3 - x$ or x^2y^2. The learning of these functions took from tens to hundred generations, and the result scheme precisely computed the desired function. Also more complicated functions were successfully evolved. The progress of evolving function $x^3 - 2x^2 - 3$ can be seen in the Fig. 7, Fig. 8 and Fig. 9. We can see that the resulting scheme is quite big, and it took 3000 generations to achieve the maximal fitness.

In the second experiment we have adapted two existing computational agents embedding the multi-layer perceptron (MLP) and the radial basis function (RBF) neural network. These agents represent two different computational methods for

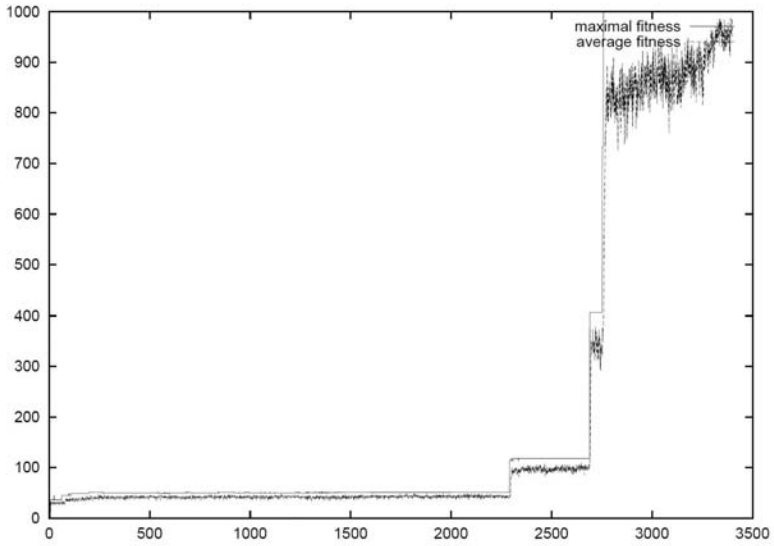


Fig. 7. Approximating function $x^3 - 2x^2 - 3$: The history of the maximal and average fitness

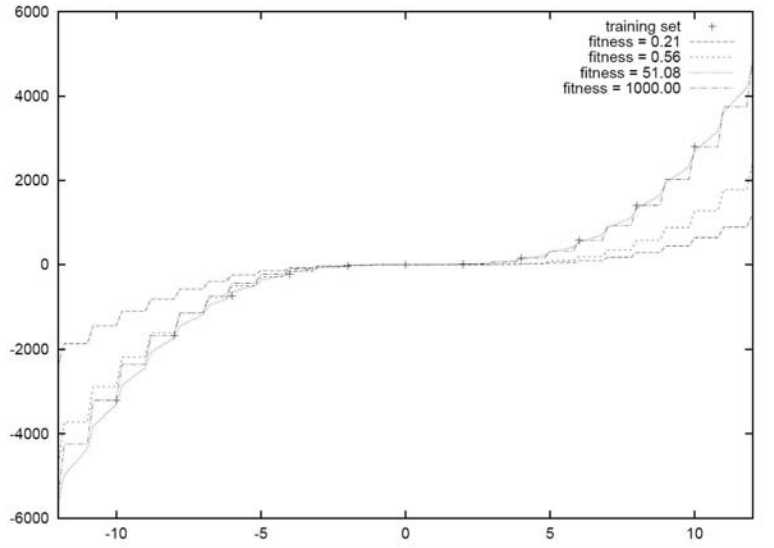


Fig. 8. Approximating function $x^3 - 2x^2 - 3$: The best schemes from generation 0, 5, 200 and 3000

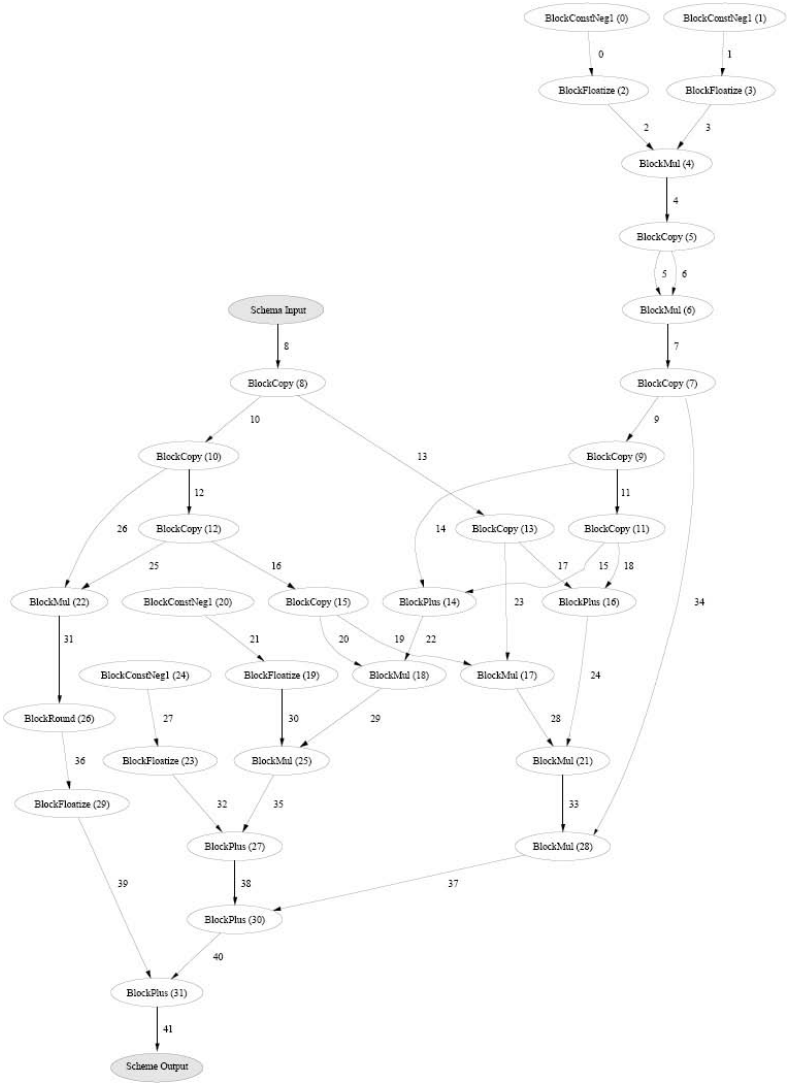


Fig. 9. Approximating function $x^3 - 2x^2 - 3$: The best scheme from generation 3000

the solution of similar categories of tasks. Both agents were tested with and without the autonomous behavior support.

Overheads of the architecture are summarized in Table 1. The creation of the agent takes 2-3 times longer since all the structures must be initialized. The communication overhead is around 30% when dealing with message delivering. However, in real-life scenario of task solving, the overhead is only about 10%.

Table 2 summarizes the measured results of *optimization of the partner choice*. We simulated a usual scenario when an agent needs to assign some tasks to one of admissible partners. This agent uses a collection of different tasks and assigns them to the computational agents successively. The total duration of the computation and the average error of computed tasks were measured. A significant improvement of the efficiency can be seen.

Experiments with *optimization by reusing results* are summarized in Table 3. We have constructed several collections of tasks with different ratios of repeated tasks (quite a usual situation when, e.g., evaluating the population in genetic algorithms). We compared the total computation-times of the whole collection with and without the optimization enabled. We can see that the optimization is advantageous when the ratio of repeated tasks is higher than 20%. When more than 40% are repeated the results are significant.

Table 1. Comparison of the agent with and without the autonomous support architecture

	Without the arch.	With the arch.
Agent creation time	3604 μs	9890 μs
Message delivery time	2056 μs	2672 μs
Total computation time	8994681 μs	9820032 μs

Table 2. Optimization of the partner choice. Comparison of choices made by different criteria

	Error	Duration
Random choice	11.70	208710ms
Best speed	1.35	123259ms
Best Accuracy	1.08	274482ms
Best services	1.17	102247ms

Table 3. Optimization by reusing the results of previously-computed tasks (duration in milliseconds).

Repeated tasks	Optimized	Standard
0 %	135777097	121712748
20%	94151838	90964553
40%	50704363	91406591
60%	47682940	90804052

7 Conclusions and Future Work

We have demonstrated that *Bang* is able to help both scientists and end-users with data analysis tasks. The niche for this software has been prototype building and testing various hybrid models so far. However, it is possible to employ it for large scale distributed computations running on a cluster of workstations. The nature of evolution of MAS schemes has brought several issues that are currently being solved. We are building an ontological descriptions of computational agents and data tasks, and we are enhancing the evolution by reasoning component. The resulting hybrid search for MAS solution to a particular problem represented by data should be not only automatic, but also feasible in terms of computational time and resources consumption.

References

1. Bonissone, P.: Soft computing: the convergence of emerging reasoning technologies. Soft Computing **1** (1997) 6–18
2. Nwana, H. S.: Software agents: An overview. Knowledge Engineering Review **11** (2) (1995) 205–244
3. Doran, J. E., Franklin, S., Jennings, N. R. and Norman, T. J.: On cooperation in multi-agent systems. The Knowledge Engineering Review **12** (3) (1997) 309–314
4. Weiss, G., ed.: Multiagents Systems. The MIT Press (1999)
5. Neruda, R., Krušina and P., Petrová, Z.: Towards soft computing agents. Neural Network World **10** (5) (2000) 859–868
6. Franklin, S. and Graesser, A.: Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Intelligent Agents III, Springer-Verlag (1997) 21–35
7. Neruda R., et al.: (Bang project home page) http://bang.sf.net
8. Vaculin, R. and Neruda, R.: Concept nodes architecture within the Bang3 system. Technical report, Institute of Computer Science, Academy of Science of the Czech Republic (2004)
9. Aha, D. W. and Wettschereck, D.: Case-based learning: Beyond classification of feature vectors. In: European Conference on Machine Learning (1997) 329–336
10. Aamodt, A. and Plaza, E.: Case-based reasoning : Foundational issues, methodological variations, and system approaches. AICom — Artificial Intelligence Communications **7** (1) (1994) 39–59
11. Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)

Ontology Acquisition Supported by Imprecise Conceptual Refinement — New Results and Reasoning Perspectives*

Vít Nováček

Faculty of Informatics, Masaryk University
Botanicka 68a, 602 00 Brno, Czech Republic
xnovacek@fi.muni.cz

Abstract. The significance of uncertainty representation has become obvious in the Semantic Web community recently. This paper presents new results of our research on uncertainty handling in ontologies created automatically by means of Human Language Technologies. The research is related to OLE (Ontology LEarning)[1] — a project aimed at bottom–up generation and merging of domain–specific ontologies. It utilises a proposal of expressive fuzzy knowledge representation framework called ANUIC. We discuss current achievements in taxonomy acquisition and outline some interesting applications of the framework regarding non–traditional reasoning perspectives.

1 Introduction

This paper builds on a novel representation of uncertain knowledge in the scope of automatic ontology acquisition [1]. The main objective of the ontology acquisition platform OLE is to implement a system that is able to automatically create and update domain specific ontologies for a given domain. We emphasise an empirical approach to the ontology construction by means of bottom-up acquisition of concepts from the domain-relevant resources (documents, web pages, corpus data, etc.). The acquisition process is incrementally boosted by the knowledge already stored in the ontology.

The concepts extracted from a single resource form so called miniontology that is instantly integrated into the current domain ontology. The integration phase is the moment when the need of uncertainty representation arises. Even if we could obtain precise conceptual constructions from individual resources (e. g. *birds fly*), we will experience infeasible consistency difficulties when trying to establish more complex precise relations between the concepts in broader scope of the whole domain (as illustrated by the popular example: the fact *birds fly* collides with the statements *penguins are birds; penguins do not fly*). Besides the inconsistency handling, there are also important cognitive motivations of the utilisation of uncertainty in our empiric ontologies that led us to the proposal of a novel framework for representing uncertain knowledge. It

* This work has been supported by the Academy of Sciences of the Czech Republic, 'Information Society' national research program, the grant AV 1ET100300419.

[1] The project's web page can be found at URL: http://nlp.fi.muni.cz/projects/ole/.

is called ANUIC (Adaptive Net of Universally Interrelated Concepts). This framework, its motivations and its technical details are depicted in detail in [1]. In this paper, we describe the progress in our current research in the meaning of new taxonomy acquisition techniques implemented and more elaborate results achieved (Section 2 and Section 3). The up-to-date results of the proposed conceptual refinement technique are described as well. Section 4 offers initial application sketches of ANUIC-based reasoning paradigm. Section 5 resumes related work briefly. We conclude the paper in Section 6.

2 Application of Conceptual Refinement in Taxonomy Acquisition

The technical description of application of our framework in the taxonomy acquisition from English natural language texts is presented here[2]. We use the well–known pattern–based technique [3] for creation of miniontologies from each input resource. These ontologies are ANUIC-integrated then in order to build a reference ontology that is exploited by the consequent steps of complex taxonomy acquisition. Section 2.1 elaborates our method based on approximate clustering that ensures significant enhancement in coverage. And eventually, the meta–algorithm of conceptual refinement is introduced in Section 2.2.

2.1 Clustering and Autonomous Annotation

The clustering is very tempting in the scope of acquisition of taxonomy of an ontology. The main idea of this approach is to gain clusters of similar terms and induce a hierarchical structure upon these terms somehow. However, these approaches usually do not offer a reliable automatic mechanism of annotation of the resulting clusters that naturally correspond to classes in an ontology.

We can obtain an initial domain ontology ontology by integration of ontologies gained from particular resources by pattern–based methods into the ANUIC format (see Section 2.2 for details). Thus we obtain a reference ontology that can be used for annotation of clusters obtained from the same resources. The contribution of such technique is obvious — we will dramatically increase the ontology coverage by incorporation of all significant terms from the resources. We have designed a method similar to *k-means* one–level clustering, tuned to suit our demands in order to identify rough clusters in the input data.

Preprocessing Specialities We require our platform to be scalable even for extreme amounts of data. Therefore we are interested also in efficiency of the clustering. The clustering speed is significantly influenced by the dimension of the feature space.

We use quite a simple and standard metric of term similarity given by a cosine distance between vectors that represent their contexts. Given a set $F = \{f_1, \ldots, f_n\}$ of features, the vector $V_T = (v_1, \ldots, v_n)$ for term T is constructed this way:

[2] We have concentrated on extraction of single general terms in the presented experimental settings, but other techniques of acquisition of more specific and multi-word terms (like those in [2]) can be easily incorporated as well.

– assign w^{d-1} to v_i, if feature f_i is contained within a vicinity of $\lceil \frac{CS}{2} \rceil$ from T; $w \in (0, 1\rangle$ is a predefined weight, CS is the relevant context size and d is distance of the respective feature from T[3];

– assign 0 to v_i otherwise.

Following the main idea of the meta–algorithm presented in Section 2.2, we extract the initial concepts from single resources of relatively small size and refine them further by empirical integration of the resulting ontologies.

Therefore we can select features only from the particular resources. We have tested several measures (like TF/IDF) for feature selection on the whole domain as well as on the isolated resources, but we have found out that a specific heuristics performs best for our method. We simply discard *hapax legomena* (terms with frequency equal to 1) from the resource's dictionary (without application of a stop-list, because we would like to have the functional words in our contexts as well). Thus we obtain a feature space of dimension in range from 500 to 1,500 for most of the resources, which is satisfactory. The terms themselves are extracted in a similar way — a general English stop-list is applied and the terms with frequency above a given threshold are considered as terms. Frequency of 5 was found to be reasonable for it does not eliminate any domain–specific words and does not bring too many unwanted garbage–words in most cases.

Simplified Rough Clustering The variants of *k-means* clustering are discussed and briefly analysed for example in [4]. General characteristic of all algorithms of this kind is that they find k points (centres) in the data space that minimise average square distance of all points in the data–set from these centres. Then the clusters are usually defined as a k-sized set of balanced groups of points that are nearest to particular centres.

Many algorithms find a local minimum for the problem in an iterative manner. We have found usual implementation of *k-means* clustering unsuitable for our reasons mainly due to their speed. We are not very interested in optimality of our clusters. Moreover, the points in our data space are quite uniformly distributed in most cases because of the restricted size of the feature space and characteristics of natural language that lay beyond the feature selection.

We have implemented a non–optimal (even locally), but very efficient technique that provides us with rough clustering of the initial resources that is further utilised within the refinement of ontologies gained from particular resources[4].

The method of simplified rough clustering is described as follows:

input: set V of feature–vectors mapped to respective terms, k (number of clusters), r (number of optimisation repeats - value 5 was found to be sufficient)
output: clusters of terms

[3] When $w = 1$, the contexts are represented as bag of words. When $w < 1$, their distance from term T is projected into the vector characteristics. The context size was set to 14 — an average length of sentence in the resource sets. Lower or higher context sizes were tried as well, but without any significant contribution. All the other parameter settings used are further specified in Section 3.

[4] The technique presented here could be however used for preparing reasonable initial means and related reduction of iterations for the classical *k-means* clustering methods.

1. prepare the initial means:
   1.1 choose a random element from input vectors, include it in a set I_m;
   1.2 repeat until the set I_m does not have k elements:
      1.2.1 compute centroid c of the set of vectors I_m;
      1.2.2 choose a vector v from V that is farthest from c;
      1.2.3 include the v into I_m;
2. for each vector d_i in I_m, iteratively factorise V into a system B of respective sets B_i such that all these sets differ in size by at most 1 and they contain the vectors from V that are nearest possible to d_i;
3. construct a set C that contains centroids for all sets in B;
4. for each vector in V, construct the map M_s of the nearest vectors from C to their order in the sequence (0 corresponds to the nearest whereas the $k-1$ to the farthest centroid from C);
5. repeat r-times and record respective clusters along with their score:
   5.1 randomly shuffle V and assign it to temporary set S;
   5.2 sequentially process S and assign each vector to the nearest available cluster represented by centroids from C, keeping the clusters as balanced in size as possible;
   5.3 compute the score for the obtained clustering by summing up the numbers pointed by respective cluster–centroid in M_s for each vector in each cluster;
5. return clusters with the best (lowest) score;

Annotation The annotation of the clusters using the reference ontology is then implemented this way:

input: clusters of terms, reference ontology, t_h (hyperonymy confidence threshold, 0.7 was found to be reasonably discriminative)
output: annotated classes of terms in the form of a simple ontology

1. for each cluster:
   1.1 create an empty set H of hyperonymic relation "stubs";
   1.2 for each term in the cluster:
      1.2.1 if the term is present in the reference ontology, include all of its hyperonyms with conviction μ-measure higher than t_h into set H, recording the frequency of the respective observation for future merging;
   1.3 annotate all terms in the cluster with the hyperonyms from the set H;
2. return the respective ontology representation for the annotated clusters;

2.2 Refinement by Integration

The conceptual refinement idea lies in integration of small ontologies into bigger ones, smoothing many of the crisp and possibly incorrect relations by uncertain empirical evidence from large number of observations. It is inspired by a simple analogy of human mind and utilises the inherent dynamics and uncertainty of the ANUIC framework[5]. The process of integration of newly coming facts is similar to the process of how people construct their conceptual representations — first they have almost crisp evidence of a relation between objects (for example that *dogs have four legs*). This opinion is strengthened by further observations of four–legged dogs, but one day they see a cripple dog having only three legs. So they have to add another instance of the "*have-number-of-legs*" relation, but with much more decreased relevancy (unless they keep seeing other and other different three–legged dogs). And this is an example of what we call continuous *meaning refinement* of conceptual models and what is happening also when new resources are integrated within the ANUIC framework.

[5] The heuristic used for efficient assignment of the fuzzy measures that symbolise relevancy of the statements represented in ANUIC is given in [1]. Note that for the optimal performance of the ANUIC–based integration, it is needed to process at least tens of relevant resources of a sufficient size (hundreds or more words). We should not await reasonable refinement results when providing only few documents with size of a couple of sentences — even human learners cannot process such small amounts of data in order to create a valuable opinion about the domain's conceptual structure.

Concrete application of the above mechanism in taxonomy acquisition is quite straightforward and conforms to the following meta–algorithm description:

input: domain resources in natural language
output: uncertain complex domain taxonomy

1. process the resources by the pattern–based method and produce a set of ontologies S_p;
2. merge the ontologies in S_p into one ontology R;
3. process the resources by the clustering–based method using R as a reference ontology and produce set of ontologies S_c;
4. merge the ontologies in S_c and produce ontology C;
5. join the R and C in order to produce domain taxonomy in ontology D;
6. return D;

We can easily update the domain ontology when keeping the track of how a relation was obtained. Thus we can still identify the more precise "reference" relations in the domain ontology D. We add the new resource by processing it first by the pattern–based technique. Then we integrate the result into the domain ontology and process the resource again by clustering–based method, annotating the classes using reference subset of D. The result of this step in then integrated in D as well — the new resource is completely covered then.

3 Selected Results of Taxonomy Acquisition

In the following we describe some of the experiments with taxonomy extraction in OLE and show their results. The improvement of the integration within the ANUIC knowledge representation format is then illustrated in Section 3.2.

3.1 Extraction Phase and Its Initial Results

We tested the taxonomy acquisition on a sample of $3,272$ computer science articles, automatically downloaded from the web. The compound size of the resources was $20,405,014$ words. For the approximate manual evaluation we randomly chose five ontologies for respective resources from the whole set for each run of a method.

For each selected ontology corresponding to a resource, we computed precision as the ratio of "reasonable" relations compared to all extracted relations[6]. The reasonability of a relation was judged by a committee of computer science experts and students after analysing the respective resources.

The coverage was computed as the ratio of number of extracted significant terms (nouns for the simple experimental settings) to all significant terms present in the resource. For all the measures of informal precision (*Pr.*) and coverage (*Cov.*), an average value was computed. We present these results in Table 1, provided with respective average original resource size and number of all concepts extracted.

The *M1* row contains results of pattern–based extraction. The *M2* rows contain results of clustering–based method for respective parameter settings given in Table 2. The rows' headings present settings ID, in the columns there are values of the respective parameters. The cluster size is used for derivation of the k parameter for clustering algorithm.

[6] For the clustering–based acquisition only 50 randomly selected relations were evaluated for each ontology, because the average number of all relations was too high for manual evaluation.

Table 1. Selected results of initial taxonomy extraction

Method	Res. sz. (wrd.)	No. of conc.	No. of rel.	Pr. (%)	Cov. (%)
M1	4275	20.6	15.2	61.73	1.83
M2/S1	5777	138.4	1191.8	45.78	100
M2/S2	4669	106.2	494	33.11	100
M2/S3	4827	136.6	1336.2	46.13	100
M2/S4	5339	128.25	680	41.52	100

Table 2. Settings for clustering–based method

Settings ID	Context size	Position weight	Cluster size
S1	14	1.0	10
S2	14	1.0	5
S3	14	0.7	10
S4	14	0.7	5

3.2 Improvement Obtained by Uncertain Conceptual Refinement

In order to produce reference ontology for the autonomous cluster annotation, we generated ontologies for each resource by pattern–based OLE module and merged them into one ANUIC structure. We used the heuristics described in Section 2.2 for configuration of the parameters.

Using the ANUIC–integration we gained a taxonomy with $5,538$ classes, $9,842$ individuals[7] and $61,725$ mutual *is-a* relations.

It is very hard to formally decide what is the representation's exact improvement when compared to the knowledge stored in the former crisp ontologies. But we can again give at least a rough picture — when we considered only the relations with the highest μ–measure(s) relevant for a particular concept[8], we can compute an approximate ratio of "reasonable" relations similar to the one presented in Section 3.1. We computed the ratio on a random sample of 50 relations from the whole merged ontology and obtained the value 84 %, which definitely shows an improvement.

The ontology gained by incorporation of results of pattern–based method into ANUIC was used as the reference for clustering–based method. The results of the ANUIC merge of the source crisp ontologies for both methods and various settings of the algorithms are in Table 3 below.

We used the same merging parameters and criteria of reasonability as for the creation and evaluation of the reference ontology. Only the relations with the highest conviction(s) were taken into account for evaluation. The precision computed on random sample of 50 relations from the merged ontology is given in column Pr_{latter}. The

[7] We empirically assume that a concept is an individual as long as it has no hyponyms.

[8] Which is by the way a very strong restriction, the range of possible interpretations of the concrete conviction values is much higher.

average crisp precision for respective source ontologies is in column Pr_{former}. The improvement (in percents) is in column *Improvement*.

Table 3. Results of merging for clustering–based method

Settings ID	Pr_{former}	Pr_{latter}	*Improvement*
M1	61.73	84.0	136.08
M2/S1	45.78	65.52	143.12
M2/S2	33.11	65.38	197.46
M2/S3	46.13	63.16	136.92
M2/S4	41.52	64.07	154.31

The ontology with the best characteristics (gained with $S1$ configuration) was experimentally merged with the reference ontology. Resulting ontology has much higher range than the reference one — it contains $1,584$ classes[9] and $30,815$ individuals, interconnected by $1,293,998$ relations in the taxonomy. The approximate precision of the 50 randomly selected relations with the highest conviction was 71.05 %. It is of course slightly lower than the similar measure for the reference ontology, but this is not a big drawback when we consider the widely increased coverage of the domain.

A sample from the resulting extended uncertain domain ontology is given in Figure 1. The ovals represent classes, squares are individuals and arrows go from subconcept to its super-concept, labelled by respective conviction μ–measures.

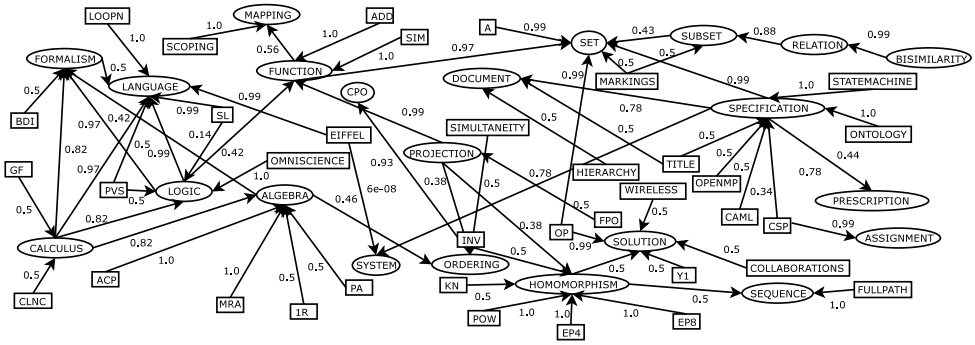


Fig. 1. Sample from the merged computer science ontology

[9] Some of the former classes were turned into individuals — this is a direct consequence of the annotation algorithm.

4 Study on the ANUIC Format Utilisation

In the following subsections we present basic reasoning perspectives of the provided universal uncertain knowledge representation paradigm[10].

4.1 Expressivity of the Proposed Format and Related Reasoning Issues

The proposed format allows to store any kind of relations between conceptual nodes. We can store anything that we can extract — the taxonomical ordering of conceptual terms, other lexico–semantic relations like synonymy and meronymy or domain–specific arbitrary relations. The edges in ANUIC can represent even negation or conceptual descriptions like conjunction or disjunction[11]. For example, the statement $A \equiv C_1 \sqcap \cdots \sqcap C_n$ in DL notation can be encoded by n edges with respective special label (symbolising conjunction) going from the node that represents A to the nodes representing $C_1, \ldots, C_n$ in ANUIC. The meaning is obvious — both representations unambiguously say that A can be viewed as a union of the denotations given by concepts $C_1, \ldots, C_n$. Similar situation arises for negation and other constructs.

The only crucial thing for both DL and ANUIC cases is a mapping from the syntactic representation level to the semantic interpretation one. Proper definition and examination of such mapping ensures meaningful reasoning for the given paradigm. Therefore we define a set of so called ρ-interpretations that must be assigned to each ANUIC model before any reasoning can be conducted on it. These ρ-interpretations define the properties, restrictions and fuzzy set–theoretic interpretations of all the relations present in the model. Moreover, we can easily enhance the pure semantic mapping by introduction of rules into the set of descriptive ρ-interpretations.

Thus we can rigorously define a context–sensitive semantics and adjust expressive power of an ANUIC model. Proper definition of the ρ-interpretation notion, rule templates, properties of the related inference calculus and outcomes of reasoning under certain different interpretation perspectives of the same model are one of the main parts of our current research.

4.2 Reduction to Classical Reasoning

We can reduce and/or transform our knowledge bases in order to support classical definition of reasoning. Inference of new facts based on the facts already stored in an ontology using extant reasoning engines can be made after application of corresponding transformation rules tailored to our uncertain knowledge representation.

We can always reduce our knowledge repository to the OWL DL format [6], which is well studied and supported by current leading–edge Semantic Web reasoning engines. We can gain a crisp Description Logics approximation by performing an α–reduction

[10] Note that the presented considerations are not meant to be exhaustive and rigorously developed. They offer rather an initial visions of future drift of our work in the ontology acquisition and reasoning fields.

[11] There exist automatic methods of extraction of these structures — see for instance [5].

using respective α–cuts[12] on fuzzy constructs contained in the ontology and by elimination of possible ρ-interpretations that are restricted in OWL DL. Then we can use widely–adopted Description Logics reasoning on such an approximation in order to learn less–expressive but crisp facts from our knowledge base.

We can make even less restrictive reduction of an ANUIC model using only restriction of some ρ-interpretations and consequent transformation into the emerging f-OWL format [7]. However, the situation in the field of f-OWL compliant reasoning is not so mature as for the crisp case yet, as some of the crucial notions (like fuzzy general concept inclusion) still remain open problems in the scope of f-OWL inference.

4.3 Sample Non-standard Inference Tasks

The basic inference task is *subsumption* or *instance checking* of two concepts, either structural (syntactic) or semantic one. Another less general logic–based procedures like determination of *least common subsumer and most specific concept* or *unification and matching* are mentioned in [8].

These tasks are no doubt important, but we would like to support also another kind of subtler, yet maybe even more valuable reasoning services within ANUIC. This is directly related to the fuzzy nature of the underlying representation and to the ρ-interpretation notion mentioned in Section 4.1. Let us name a few of the services we study in this respect:

- *determination of concept similarity:* the notion of similarity is very vague, but can be heuristically defined using fuzzy assertions on concept subsumption, equivalence and/or disjointness; thus we can discover another important relation that can help us when finding valuable answers to vague queries like "*What are methods similar to laparocscopy in the surgery field?*" etc.;
- *empirical discovery of individuals:* we have implemented only very basic heuristic for this task so far[13] and its further development is needed; fuzzy subsumption and concept specificity determination would help if there would be enough appropriate relation instances present in an ANUIC ontology;
- *discovery, merging and splitting of synonymic nests:* this is another very important task — we would like to know which terminological labels relate to a concept with some relevancy so that we can induce fuzzy sets of respective reference terms (similar to synsets in WordNet) among ANUIC nodes; merging and splitting operations are to be used for the adaptation of these sets according to newly coming data then.

5 Related Work

Our work is to some extent similar to the one presented in [5] paper on uncertainty handling in Text2Onto [9]. Text2Onto also utilises initial automated knowledge extraction methods and integrates them into so called *Learned Ontology Model*. It incorporates uncertain rating annotations of the gained relations. A DL-consistent model is selected

[12] An α–cut of a fuzzy set A is a classical crisp set of objects that have a membership value higher than $\alpha \in \langle 0, 1\rangle$ with respect to A.

[13] A concept is determined to be an individual as far as it has no empirically found subconcepts.

then as a subset of the statements in the learned model according to these annotations. On the other hand, we deal with the inconsistencies internally and allow to reason generally among all the gained knowledge under different well–defined perspectives. This provides us with very valuable option of *contextualised* inferences, among other things.

The reasoning perspectives of our paper are related to the work on fuzzy extension of OWL, which is presented in [7]. However, the possible transformations of ANUIC into the logical fuzzy formats have to be studied in more detail before we can make proper conclusions and comparisons.

6 Conclusions and Future Work

We have presented new progress in integration of automatically web–gained knowledge into the ANUIC framework that deals with uncertain knowledge in ontologies. The theoretical background of fuzzy sets methodology allows to develop an appropriate calculus and consecutively build novel inference tools to reason among the concepts stored in the expressive ANUIC format very efficiently. The main contribution rests in the presented boost of initial ontology acquisition methods. However, the format's nature together with visions offered in Section 4 shows very challenging future directions and possible outcomes of our current work.

Note that we have implemented SOLE — a basic web interface presenting the basic functionalities of the OLE ontology acquisition framework. It processes documents uploaded by users and creates (fuzzy) ontologies from them according to user–specific patterns for extraction of semantic relations. The system can be accessed at URL: `http://nlp.fi.muni.cz/projects/ole/web/`. You can download a brief manual using the SOLE link in the header of the displayed page. Type in the *test* user–name and *test* password to use a public testing account with few pre–defined relation patterns.

Our future work will focus on incorporation of results of another extraction methods into the ANUIC ontologies in order to increase the recall and number of kinds of extracted relations. A formal development and validation of a specific calculus for ANUIC-based reasoning is needed then. The mutual correspondence and transformation possibilities between ontologies in ANUIC format and formats like OWL or f-OWL must be examined as well.

References

1. Nováček, V. and Smrž, P.: Empirical merging of ontologies — a proposal of universal uncertainty representation framework. In: Lecture Notes in Computer Science. Volume 4011., Springer-Verlag Berlin Heidelberg (2006) 65–79
2. Ryu, P. M. and Choi, K. S.: An information-theoretic approach to taxonomy extraction for ontology learning. In: Buitelaar, P., Cimiano, P., Magnini, B., eds.: Ontology Learning from Text: Methods, Evaluation and Applications. IOS Press (2005) 15–28
3. Hearst, M. A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (1992) 539–545
4. Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R. and Wu, A.: An efficient k-means clustering algorithm: analysis and implementation (2002)

5. Haase, P. and Völker, J.: Ontology learning and reasoning - dealing with uncertainty and inconsistency. In: da Costa, P. C. G., Laskey, K. B., Laskey, K. J., Pool, M., eds.: Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW) (2005) 45–55
6. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I. and McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A.: OWL Web Ontology Language Reference. (2004) Available at (February 2006): `http://www.w3.org/TR/owl-ref/`.
7. Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J. and Horrocks, I.: Fuzzy owl: Uncertainty and the semantic web, International Workshop of OWL: Experiences and Directions, Galway (2005)
8. Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F.: The Decription Logic Handbook: Theory, implementation, and applications. Cambridge University Press, Cambridge, USA (2003)
9. Cimiano, P. and Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery. In: Montoyo, A., Munoz, R., Metais, E., eds.: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB). Volume 3513 of Lecture Notes in Computer Science., Alicante, Spain, Springer (2005) 227–238

Asociativní úložiště dat v prostředí sémantického webu*

Martin Řimnáč

Ústav informatiky
Pod Vodárenskou věží 2, 182 07 Praha 8
rimnacm@cs.cas.cz

Abstrakt Použití asociativního úložiště dat je jednou z možností, jak efektivním způsobem reprezentovat data. Článek se zabývá z převážné části metodou učení takového úložiště, přičemž využívá myšlenek vize sémantického webu. Dále ukazuje souvislosti této metody s teoriemi organizace paměti živých organismů včetně člověka a její učení bez snahy tyto procesy zpětně modelovat. Jelikož se nabízí možnost využít současných webových stránek jako vstupních dat, je učící algoritmus navržen inkrementálně a výhody použití takového adaptivního přístupu jsou detailně popsány. Výsledkem algoritmu je asociativní úložiště navržené na základě všech dostupných (meta)informací, na které je možné pohlížet jako na extensionální úrovni odhadnutou sémantiku dat.

1 Úvod

Schopnost živých organismů osvojovat si nové poznatky inspiruje již celá staletí lidstvo v mnohých oblastech, filozofií začínaje a psychologií či umělou inteligencí konče. Tato schopnost je podmíněna existencí paměti, prostředku, jenž umožňuje si vybavovat někdy v minulosti uložené znalosti. S rozmachem metod umělé inteligence v 70-tých letech minulého století vzniklo mnoho modelů paměti, každý však byl navržen s ohledem na konkrétní funkcionalitu a selhával v jiných oblastech. Přes veškerou snahu nebyl nalezen jeden uspokojivý model, spíše se ukazuje, že paměť je obecně celým souborem navzájem silně propojených částí s různou funkcionalitou [1]. Pomiňme nyní rozdělení na krátkodobou nebo dlouhodobou paměť a připomeňme některé vybrané základní části paměti:

- paměť vnímání - zde se krátkodobě uchovává informace pocházející z vnějšího světa, např. fonologická smyčka pro sluch. Tato část paměti krátkodobě uchovává kontext přijímaných informací.
- sémantická paměť - definuje pojmy, významy, události.
- asociativní paměť - obecně definuje vztahy mezi pojmy

* Práce byla podpořena projektem 1ET100300419 programu Informační společnost (Tématického programu II Národního programu výzkumu v ČR: Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu) a záměrem AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

Jednou ze známek inteligence je schopnost adaptace na nové skutečnosti, paměť spojená s adaptivním učením tedy nemůže být založena na intensionálním přístupu (na základě předem známých, všeobecně platných vztahů; zde by učení odpovídalo dekompozici [6, 7]), ale naopak musí adekvátně reagovat na nově příchozí skutečnosti a postupně takovýto extensionálně pojatý (tj. vztažený ke konkrétní množině dat) model upravovat. Intensionální způsob by vedl k rozpadu modelu (data neodpovídající danému modelu), naopak nevýhodou extensionálního modelu je nemožnost zaručit jeho správnost, tedy model lze generovat nejlepší možný vzhledem ke skutečnostem z minulosti, ale nezaručuje, že bude platný (tedy bez potřeby úpravy) i pro všechny možné nově příchozí skutečnosti v budoucnosti. Další nevýhodou je možná degenerace modelu při nevhodně zvoleném formalismu úložiště.

Ukazuje se jako velmi přínosné pohlížet na uchovávané skutečnosti ze dvou pohledů [1, 16]:

- Pohled *instanční*, obsahující konkrétní parametry objektu zasazené do pojmů ze sémantické paměti.
- Pohled *zobecněný*, popisující obecné platné vztahy dané skutečnosti.

Příkladem může být skutečnost "Vlak 9006 odjíždí ze stanice Praha-Hlavní nádraží ve směru na Praha - Vršovice ve 13:15 z 5. nástupiště, kolej A" se zobecněným modelem "skutečnosti odjezd vlaku". Zpětně, rozpoznáme-li správný model, do kterého vzniklou skutečnost zařadit, víme, na které parametry se máme zaměřit. Z hlediska psychologie je zajímavé sledovat prolínání těchto přístupů v sémantické paměti[1].

Z hlediska paměti můžeme definovat tyto základní operace:

- vybavení z paměti
- uložení do paměti
- uvažování v paměti, podobnost

2 Asociativní úložiště

Navrhněme nyní jeden z možných formalismů pro asociativní úložiště dat.

Použijme teorii relačních databází jako základní kámen formalismu; metody pro získávání (odhad) extensionálních funkčních závislostí z dat [8–14], přičemž použijme inkrementální mechanismus [15, 16]. Požadujme, aby použitý formalismus odpovídal formátu RDF sémantického webu [2, 3], představujícího množinu binárních predikátů $\{(objekt, vlastnost, hodnota)\}$, což umožní použití těchto dat v sofistikovaných odvozovacích mechanismech nad RDF [4, 5]. Formalismus musí pokrývat v souladu s předchozím textem jak instanční, tak globální pohled na úložiště.

[1] Rozložení figur na šachovnici uložené jako soubor instancí modelu "figura na pozici" (začátečník) versus celá konfigurace figur jako jeden sémantický symbol (profesionální hráč).[1]

2.1 Formalismus

Navrhněme nyní implementaci úložiště pomocí binárních matic a indexů definující význam pozic v jednotlivých maticích.

Nechť v souladu s předchozím textem úložiště pokrývá jak zobecněnou úroveň (pomocí incidenční matice mezi atributy), tak úroveň instanční (pomocí incidenční matice mezi elementy - dvojicemi $(atribut, hodnota)$. Pro jednoduchost se prozatím uvažujme pouze vztahy mezi jednoduchými atributy.

Zadefinujme

- matici instancí Φ:

$$\Phi = \{\phi_{ij}\} : \phi_{ij} = \begin{cases} 1 \text{ pokud element } e_i \rightsquigarrow e_j \\ 0 \text{ jinak} \end{cases} \tag{1}$$

- matici extensionálních funkčních závislostí Ω:

$$\Omega = \{\omega_{ij}\} : \omega_{ij} = \begin{cases} 1 \text{ pokud atribut } A_i \rightarrow A_j \\ 0 \text{ jinak} \end{cases} \tag{2}$$

- matici porušených funkčních závislostí $\mho$:

$$\mho = \{\omega^{\sharp}_{ij}\} : \omega^{\sharp}_{ij} = \begin{cases} 1 \text{ pokud atribut } A_i \nrightarrow A_j \\ 0 \text{ jinak} \end{cases} \tag{3}$$

- matici aktivních domén atributů Δ:

$$\Delta = \{\delta_{ij}\} : \delta_{ij} = \begin{cases} 1 \text{ pokud element } e_j = (A_i, v_j), v_j \in \mathscr{D}_{\alpha}(A_i) \\ 0 \text{ jinak} \end{cases} \tag{4}$$

- a dále index
  - atributů $\mathscr{I}_A : \{A_i\} \rightarrow N$
  - hodnot $\mathscr{I}_V : \{v_i\} \rightarrow N$
  - elementů $\mathscr{I}_E : \{e_i\} \rightarrow N$

S ohledem na úvodní kapitolu, indexy $\mathscr{I}_A, \mathscr{I}_V, \mathscr{I}_E$ a matice Δ patří do sémantické paměti, incidenční matice $\Phi, \Omega, \mho$ pak representují paměť asociativní.

2.2 Vybavování z paměti

Proces vybavování skutečností z paměti v tomto případě odpovídá dotazování nad úložištěm, dotaz i výsledek můžeme reprezentovat pomocí vektoru $\boldsymbol{x}$. Principiálně můžeme definovat dvě operace:

- `generalizaci` – vektor $\boldsymbol{x}_k$ je rozšířen o aktivace elementů, jenž z původního podle Φ vyplývají.

$$\boldsymbol{x}^G_{k+1} = \Phi \cdot \boldsymbol{x}_k \tag{5}$$

- `specializaci` – ze kterých elementů mohou ve vektoru $\boldsymbol{x}_k$ aktivované elementy vyplývat (vektor může odpovídat více objektům).

$$\boldsymbol{x}^S_{k+1} = \Phi^T \cdot \boldsymbol{x}_k \tag{6}$$

Finálního výsledku můžeme dosáhnout:

- v jednom kroku – pak matice Φ obsahuje (i poměrně značný) počet redundantních prvků. Tato forma matice je vhodná pro dotazování.
- v nejvýše n krocích, n je velikost matice Φ – speciálním případem je matice neobsahující žádné redundantní prvky. Taková matice je určena pro ukládání na paměťová média.

Hledání matice neobsahující redundantní prvky lze převést na minimalizační problém v prostoru zobecněných vztahů např. pomocí [15] a uvažovat pouze instance vztahů z této minimalizované množiny. Problém minimalizace lze formulovat jako[2]:

$$\Omega^\star = \arg\min_{E(\Omega')} \{\Omega = \Omega' \cdot \ldots \cdot \Omega'\} \tag{7}$$

$$E(\Omega') = \sum_{\forall i,j} (\Omega' \odot (\Lambda^T - \Lambda)) \tag{8}$$

$$\Lambda = \Delta_{\mathscr{T}} \cdot \mathbf{1}(\text{size}(\Delta_{\mathscr{T}}))^T \tag{9}$$

V souvislosti s generalizací (5) definujme pojem *konzistence*. Vektor $\boldsymbol{x}$ je konzistentní, pokud každý aktivovaný element odpovídá jinému atributu. Úložiště Φ je konzistentní, pokud výsledek generalizace každého konzistentního vektoru $\boldsymbol{x}_k$ je opět konzistentní vektor.

2.3 Vztah mezi instancemi a zobecněným popisem

Ukažme souvislost mezi maticí instancí Φ a maticí zobecněných vztahů – extensionálních funkčních závislostí Ω.

Nechť Φ je konzistentní a $\exists\phi_{ij} = 1$, přičemž element e_i odpovídá atributu $A_{i'}$ a e_j analogicky $A_{j'}$. Za tohoto předpokladu ale nutně $\omega_{i'j'} = 1$ (je-li ϕ_{ij} instancí nějaké zobecněného vztahu, pak musí existovat i samotný vztah). Tedy:

$$\Omega = \Delta^T \cdot \Phi \cdot \Delta \tag{10}$$

Naopak systém funkčních závislostí generuje pozice, ve kterých se mohou nacházet instance. Tento vztah lze využít pro ponechání pouze konzistentních prvků v úložišti.

$$\Phi = \Phi' \odot \Delta \cdot \Omega \cdot \Delta^T \tag{11}$$

Přesněji řečeno, konzistentní úložiště nesmí pokrývat žádnou instanci, jenž by odpovídala v minulosti již porušené funkční závislosti, neboli[3]:

$$\Phi = \Phi' \odot \Delta \cdot (\mathbf{1}(size(\mho)) - \mho) \cdot \Delta^T \tag{12}$$

[2] Operátor $\odot$ značí násobení prvek po prvku

[3] Tento vztah umožňuje uvažovat prázdné hodnoty atributů ve vstupních datech.

2.4 Proces učení

Navrhněme nyní inkrementální postup, jak získat úložiště pokrývající vstupní množinu dat.

Inkrementální algoritmus pro odhadování struktury dat může být založen na faktu, že pokud úložiště pokrývá pouze jeden záznam $\boldsymbol{x}_0$, pak hodnotu každého atributu záznamu lze odvodit z hodnoty každého jiného atributu. V notaci binárních matic to znamená inicializaci úložiště blokem 1 jak v prostoru instancí, tak díky (10) v prostoru zobecněných vztahů (žádná z funkčních závislostí nemůže být porušena):

$$\Phi_i^{\triangledown} = \boldsymbol{x}_i^T \cdot \boldsymbol{x}_i \tag{13}$$

$$\mho_i^{\triangledown} = \mathbf{0}(size(\Omega_i)) \tag{14}$$

Na základě tohoto předpisu vygenerujeme z množiny vstupní dat $\{x_{0\ldots m}\}$ množinu inicializovaných úložišť $\{\Phi_{0\ldots m}^{\triangledown}\}$.

Inkrementální algoritmus začne s úložištěm $\Phi_0 = \Phi_0^{\triangledown}$ a $\mho_0 = \mho_0^{\triangledown}$. Pro každý další vstupní záznam $\boldsymbol{x}_i; i = \{1 \ldots m\}$ reprezentovaný úložištěm $\Phi_i^{\triangledown}$ se provede:

– Sloučí se instance z úložišť:

$$\Phi'_{i+1} = \Phi_i + \Phi_i^{\triangledown} \tag{15}$$

– Takové úložiště nemusí být konzistentní. Detekujeme spojením porušené zobecněné vztahy:

$$\mho_{i+1}^{\Delta} = \Delta((\Phi_{i+1}'^T \cdot \Delta^T) \geqslant 1) \tag{16}$$

– Na základě nově detekovaných porušení je aktualizuje matice porušených zobecněných vztahů

$$\mho_{i+1} = \mho_i + \mho_{i+1}^{\Delta} \tag{17}$$

– Nekonzistentní instance z úložiště jsou odstraněny podle (12)

$$\Phi_{i+1} = \Phi'_{i+1} \odot \Delta \cdot (\mathbf{1}(size(\mho_{i+1}^{\Delta})) - \mho_{i+1}^{\Delta}) \cdot \Delta^T \tag{18}$$

Jednoduše se nechá nahlédnout, že navržený algoritmus je polynomiálně složitý. Dále je vidět, že úložiště lze reprezentovat pouze pomocí matice instancí Φ a množinu porušených funkčních závislostí $\mho$.

3 Podpora pro komplexní atributy

V dosavadním textu jsme se omezili pouze na funkční závislosti s jednoduchými atributy na obou stranách. Tyto zobecněné vztahy však pokrývají pouze situace, kdy na základě jedné skutečnosti je možné odvodit nějakou další skutečnost. Praxe však vyžaduje uvažovat i složitější vztahy (s vyšší aritou), tedy takové, kdy výsledná skutečnost závisí na kombinaci několika navzájem nezávislých předpokladů. Pokusme se nyní rozšířit úvahy o úložišti i o tyto vztahy.

Z teorie relačních databází vyplývá, že není nutné uvažovat instance funkčních závislostí se složenými atributy v případě, že existuje pro všechny dílčí atributy ze složeného atributu jeden společný klíčový atribut; informace o existenci takového vztahu je uložena v úložišti jako metainformace. Neuvažování instancí funkčních závislostí se složenými atributy vede na 66% úsporu paměťových nároků úložiště [16].

Na základě tohoto rozšíření lze upravit generalizační mechanismus, uvažujme, že vektor γ_L reprezentuje atributy na levé straně funkční závislosti, vektor γ_R pak na straně pravé. Upravený mechanismus lze vyjádřit pomocí

$$\hat{\boldsymbol{x}}^G_{k+1} = \boldsymbol{\Phi} \cdot \boldsymbol{x}_k + \boldsymbol{\Phi}((\boldsymbol{\Phi}^T \cdot ((\boldsymbol{\Delta}^T \cdot \gamma_L) \odot \boldsymbol{x}_k)) == 1) \odot (\boldsymbol{\Delta}^T \cdot \gamma_R) \tag{19}$$

Tento vztah omezuje generalizační mechanismus pouze na vybrané (atributům na stranách odpovídající) elementy. Nejprve se pomocí specializace vyhledá množina dotazu odpovídajících klíčových elementů a k samotnému odvození dojde díky generelizaci elementů z této množiny.

Vektory atributů stran lze sloučit do jedné matice, např. pro levé strany do matice:

$$\boldsymbol{\Gamma}_L = \{\gamma_{Li}\} \tag{20}$$

Přistupme nyní k problematice odhadování vztahů s vyšší aritou stejným způsobem jako v případě uvažování pouze funkčních závislostí mezi jednoduchými atributy; zde je podle předchozího odstavce postačující v úložišti uchovávat platné instance $\boldsymbol{\Phi}$ a neplatné vztahy $\mho$ - tedy vztahy, jenž nelze pro odvozování použít, abychom dostali správný výsledek. K matici levých stran $\boldsymbol{\Gamma}_L$ zadefinujme matici porušených funkčních závislostí

$$\hat{\mho} = \{\hat{\omega}^\sharp_{ij}\} : \hat{\omega}^\sharp_{ij} = \begin{cases} 1 \text{ pokud složený atribut odpovídající } \gamma_{Li} \nrightarrow A_j \\ 0 \text{ jinak} \end{cases} \tag{21}$$

Detekci nově porušených vztahů lze provést analogickým způsobem jako (16).

3.1 Hledání levých stran

Diskutujme nyní otázku hledání levých stran. Rozšiřme nyní blokově matice levých stran a porušených funkčních závislostí těchto stran o jejich ekvivalenty pro funkční závislosti mezi jednoduchými atributy (matice (24) representuje všechny v aktuálním kroku porušené funkční závislosti):

$$\tilde{\boldsymbol{\Gamma}} = [\boldsymbol{E}|\boldsymbol{\Gamma}_L] \tag{22}$$

$$\tilde{\mho} = [\mho|\hat{\mho}] \tag{23}$$

$$\tilde{\mho}^{\Delta} = [\mho^{\Delta}|\hat{\mho}^{\Delta}] \tag{24}$$

Představme si nyní situaci, kdy funkční závislost $\{A_{\mu_1} \dots A_{\mu_a}\} \rightarrow A_j$ je nově porušena. V tomto okamžiku se pokusíme nalézt atribut A_e, kterým bychom levou stranu funkční závislosti rozšířili tak, aby z ní bylo možné hodnotu atributu A_j opět odvodit. Takový atribut musí splňovat:

$$A_e \nleftrightarrow A_j \wedge A_e \nleftrightarrow A_{\mu_i} \forall i \tag{25}$$

Alternativně, úložiště může udržovat metainformaci o *kandidátech* na levou stranu Γ_C. Takovýto kandidát vznikne pomocí[4]:

$$\forall \hat{\omega}_{ij}^{\sharp\Delta} = 1 : \gamma_C = \gamma_{Li} + \beta_(j), \Gamma_C := [\Gamma_C | \gamma_c] \tag{26}$$

Pokud vektor γ_c je shodný s a navzájem různými kandidáty z Γ_C, tento kandidát může reprezentovat levou stranu nějaké funkční závislosti, za základě níž lze v úložišti odvozovat (nediskutujme nyní triviálnost takové funkční závislosti, proces detekce netriviálních funkčních závislosti odpovídající (10) musí být odpovídajícím způsobem upraven). Poznamenejme, že je nutné tuto novou stranu otestovat pomocí (16) na konzistenci.

4 Závěr

Cílem práce je vytvořit úložiště dat umožňující jednak vyhledávání nad uloženými daty, tak schopné se neustále učit - rozšiřovat se o nové poznatky. Zásadní roli v úložišti hraje odhadnutá struktura dat umožňující je efektivním (neredundantním) způsobem uchovat. Celý článek je zasazen do kontextu inspirovaného pamětí živých organismů; ne proto, aby se navržené úložiště snažilo modelovat takovou paměť, spíše pro představu, jakým problémům obecně spojených s uchováváním a zpřístupňováním dat z paměti je potřeba čelit.

Navržená implementace úložiště pomocí binárních matic pohlíží na data ve dvou rovinách.

- V první rovině jsou uloženy instance (odhadnutých) extensionálních funkčních závislostí, článek ukazuje, že za určitého předpokladu je možné uchovávat pouze instance funkčních závislostí mezi jednoduchými atributy, což vede ke značné redukci paměťových nároků.
- V druhé rovině pak úložiště pokrývá metainformaci o porušených funkčních závislostech, tedy o vztazích, jenž nad daty v úložišti neplatí a jejich použití by vedlo k mylným závěrům.

Tyto dvě roviny pohledu jsou postačující jak pro dotazování nad daty, tak pro další rozšiřování úložiště o nová data.

Díky použití binárních matic je velmi jednoduché úložiště interpretovat ve formátu pro sémantický web. Na takto vzniklé dokumenty je možné pohlížet jako na dokumenty explicitním způsobem popisující odhadnutou sémantiku dat.

Budoucí práce se bude zaměřovat na použití řídkých matic pro uchování dat a s tím související úpravy odhadu složitosti jednotlivých operací, na rozšíření portfolia vztahů (např. detekce rekurzivních vlastností mezi objekty) a na získávání dat nejen z tabulek, ale i stromových struktur (včetně struktur primárně popisující formátování dokumentů - X-HTML dokumenty).

V případě úspěšného vyřešení zmíněných problémů získáme úložiště schopné se adaptivně učit z webových stránek, přičemž získané informace mohou být prezentovány včetně jejich odhadnuté sémantiky.

[4] $\beta(j)$ representuje vektor s jediným prvkem rovným 1 na j-té pozici.

Reference

1. Baddeley A.: Vaše paměť. ISBN: 80-7242-046-1, EAN: 9788072420469
2. Antoniou, G. and v Harmelen, F.: A Semantic Web Primer. MIT Press, ISBN: 0-262-01210-3 (2004)
3. Miller, E., Swick, R. and Brickley, D.: Resource Description Framework. `<http://www.w3.org/RDF/>` [on-line] (2004)
4. Horrocks, I. and Tessaris, S.: Querying the semantic web: a formal approach. In Proc. of Semantic Web Conf. (ISWC 2002), volume LNCS 2342, Springer-Verlag (2002) 177–191
5. Broekstra, J., Kampman, A. and v Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RFD Schema. In Proc. of Semantic Web Conf. – ISWC 2002, Springer **LNCS 2342** (2002) 54–68
6. Biskup, J., Dayal, U. and Bernstein, P. A.: Synthesising Independent Database Schemas. In SigMod (1979) 143–150
7. Bernstein, P. A., Swenson, J. R. and Tsichristzis, D. C.: A Unified Approach to Functional Dependencies and Relations. In SigMod (1975) 237–245
8. Flach, P. A. and Savnik, I.: Database Dependency Discovery: A Machine Learnig Approach. In AI Communications **12**(3) (1999) 139–160
9. Mannila, H., and Räihä, K. J.: Dependency Inference. In Proc. of VLDB, ISBN: 0-934613-46-X (1987) 155–158
10. Mannila, H. and Räithä, K. J.: Algorithms for Inferring Functional Dependencies from Relations. In Data & Knowledge Engineering, Elsevier **12** (1994) 83–99
11. Kivinen, J. and Mannila, H.: Approximate Inference of Functional Depentencies from Relations. In Proc. of 4. int. conf. on Database Theory, Berlin, Germany, ISSN: 0304-3975 (1995) 129–149
12. Mannila, H. and Räithä, K. J.: Design by Example: An Applications of Armstrong Relations. Academic Press, Journal of computer and system sciences 33 (1986) 129–141
13. Giannella, Ch. and Robertson, E.: On Approximation Measures for Functional Dependencies. In ADBIS 2002: Advances in databases and information systems, Elsevier, ISSN 0306-4379 (2004) 483–507
14. Akutsu, T., Miyano, S. and Kuhara, S.: A Simple Greedy Algorithm for Finding Functional Relations: Efficient Implementation and Average Case Analysis. Theoretical Computer Science, ISSN: 0304-3975 **292**(2) (2003) 481–495
15. Řimnáč, M.: Transforming Current Web Sources for Semantic Web Usage In Proc. of SOFSEM 2006, ISBN: 80-903298-4-5 **2** (2006) 155–165
16. Řimnáč, M.: Odhadování struktury a asociativní úložiště dat
In Doktorandský den 06, MATFYZPRESS, ISBN: 80-86732-87-8 (2006) 135–142

Towards Digital Mathematical Library
Optical Character Recognition of Mathematical Texts

Petr Sojka

Faculty of Informatics, Masaryk University in Brno, Czech Republic

Abstract. This paper describes a prototype of the OCR math engine built in the DML-CZ project. Solution stands on the combination of FineReader and InftyReader programmes. The achieved error rate (counting not only character errors, but also errors in the recognition of structure of mathematics notation) decreased from an initial 12% to under 1%.

1 Motivation

The main obstacle to easily accessing the vast amount of information in papers published in the "pre-digital born" era, is the fact that papers are not available in well-designed, standard, fully indexed electronic form, together with detailed metadata and full-text search capabilities. Digitisation projects try to diminish this barrier by not only scanning the published pages, but applying optical character recognition techniques. So far, the error rate of fully automated OCR for mathematical text was not acceptable even for recognition of bibliographic entries of scanned papers, and the texts were retyped twice and differences checked so as to achieve high precision. As the wide-spreading of XML and the adoption of MathML for math document exchange and search continues, demand for quality character recognition of math text also grows, with the aim of achieving high precision of document retrieval even for retrodigitised documents.

2 DML-CZ

Encouraged by the idea of the World Digital Mathematics Library (WDML) [2] and by digitization activities worldwide, the Czech Mathematical Society initiated the digitization project "DML-CZ: Czech Digital Mathematics Library" to ensure availability of mathematical literature which has been published throughout history in the Czech lands, in digital archival form. The project, proposed for the period 2005–2009, has been supported by the Academy of Sciences of the Czech Republic (AS CR) in the frames of the national research programme "Information Society" [8, 6].

The aim of the project is to investigate, develop and apply techniques, methods and tools that will allow the creation of an infrastructure and conditions for establishing the Czech Digital Mathematics Library (DML-CZ). The DML-CZ will contain professional journals of international standing, conference proceedings, selected monographs, textbooks, theses and research reports. Besides the

retrodigitised material the existing born-digital one will be involved and proper measures will be proposed for easy processing and incorporation of the new literature (which will be created in electronic form in the future). Presumably, in view of the common history and the lingual closeness, the suitable Slovak mathematical literature will be included as well. Efforts will be made to apply the developed procedures and tools also on the Czech journals scanned in the SUB Göttingen. The estimated extent of the relevant literature ranges from 150 to 200 thousands pages. Upon its completion the DML-CZ will be integrated into the WDML. The techniques and tools developed for the DML-CZ might be later used for digitization in other fields of science.

3 OCR in DML-CZ

Automation of digitization processes with the goal of building searchable and reusable digital library is complicated task [7]. To prepare high quality data by automatic procedures, we have made extensive experiments with OCR techniques and programmes. As the testbed we have scanned forty volumes (almost 40 000 pages) of the Czech Mathematical Journal (CMJ) in the Digitisation Centre of the Library of Academy of Sciences of the Czech Republic in Jenštejn near Prague.

Images were scanned at 600 DPI with 4bit depth. We have found that having a 4bit depth during image preprocessing in Book Restorer programme (or OCR programme itself) gives better results during OCR. Scanned images are straightened, binarized and clipped in the Book Restorer.

Tests with various OCR programmes showed that no single one gives acceptable results, with character error rates often above 10% (counting wrong character positions and font types as errors too). For text recognition, FineReader by ABBYY (`http://www.abbyy.com`) gave the best results, whereas for the structural recognition of mathematics InftyReader [10] had impressive results. We have communicated to the authors of Infty Project the possibility to combine the programmes, and got a version of the programme that is able to read PDF with a text layer inserted by FineReader [3].

We found that setting the parameters of the OCR engine (language, wordlist consultation) influences the precision significantly. We trained FineReader on the type cases used at the printer where CMJ was typeset using hot type [11].

At the end of extensive experiments, we developed a method of OCR processing consisting of several phases:

1. A page or block of text is recognised for the first time using a universal setup (non-language specific). A histogram of character bigrams and trigrams from words with lengths higher than three is created.
2. The computed histogram of the text block is compared [1] to histograms created from the journal data during the training phase for all languages used (English, French, Russian, German and Czech). Perl module Lingua::Ident is used. Block with bibliography is detected by different algorithms and is treated differently.

3. Page or block of text is processed for the second time with parameters optimised for recognised 'language' in previous step and saved as PDF with text layer.
4. PDF is passed to InftyReader and results are stored in Infty Markup Language (IML).
5. IML is postprocessed by a home-grown programme in Java to fix recognition errors of some of the accented characters that Infty does not yet have in its glyph database.

4 Results

Using the process outlined above we managed to decrease the error rate from initial 11.35% (universal language setup of FineReader) to an average 0.98% error rate. The whole processing is fully automatised after initial training. Our prototype implementation is now being integrated into the standard DML-CZ work-flow. FineReader component using the FineReader SDK is being integrated by Elsyst Engineering into Sirius system as used in the production workflow of Digitisation centre in Jenštejn.

5 Conclusion and Further Work

We still think that there is place for improvement by fine-tuning the parameters of all the programs used in the work-flow. Finally, the method could be used with minimal efforts to recognise images in other digitisation projects to obtain high precision texts in XML/MathML or LaTeX formats for text indexing or retypesetting. Detailed report on this work will appear in [9].

The next step is using the full texts of scanned articles and their Mathematical subject classification (MSC) codes for supervised training of MSC classifier, using various machine learning techniques.

Acknowledgement

Special thanks go to Radovan Panák and Tomáš Mudrák, who implemented the reprted OCR workflow as part of their MSc thesis [5, 4]. Thanks go to all members of the DML-CZ project team, not only those working on OCR subproject. This work has been partially supported by the grants 1ET200190513 and 1ET100300419.

References

1. Dunning, T.: Statistical identification of language. Technical Report MCCS 94-273, New Mexico State University, Computing Research Lab, 1994.
2. Jackson, A.: The Digital Mathematics Library. Notices of the AMS **50** 4 (2003) 918–923

3. Toshihiro Kanahori and Masakazu Suzuki: Refinement of digitized documents through recognition of mathematical formulae. In Proceedings of the 2nd International Workshop on Document Image Analysis for libraries, Lyon, France (April 2006) 95–104
4. Mudrák, T.: Digitalizace matematických textů. Master's thesis, Masaryk University in Brno (April 2006)
5. Panák, R.: Digitalizácia matematických textov. Master's thesis, Masaryk University in Brno (April 2006)
6. Rákosník, J., Sojka, P. and Šárfy, M.: Optical Character Recognition of Mathematical Texts in the DML-CZ Project, September 2006. accepted for publication as book chapter CMDE 2006 (A.K. Peters).
7. Simske, S. J. and Lin, X.: Creating Digital Libraries: Content Generation and Re-Mastering. Technical Report HPL-20003-259, HP Laboratories Palo Alto (December 2003)
8. Sojka P.: From Scanned Image to Knowledge Sharing. In Klaus Tochtermann and Hermann Maurer, editors, *Proceedings of I-KNOW '05: Fifth International Conference on Knowledge Management*, Graz, Austria, Know-Center in coop. with Graz Uni, Joanneum Research and Springer Pub. Co. (June 2005) 664–672
9. Sojka, P., Panák, R., and Mudrák, T.: Optical Character Recognition of Mathematical Texts in the DML-CZ Project, September 2006. accepted for publication as book chapter CMDE 2006 (A.K. Peters).
10. Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori: INFTY — An integrated OCR system for mathematical documents. In C. Vanoirbeek, C. Roisin, and E. Munson, editors, *Proceedings of ACM Symposium on Document Engineering 2003*, Grenoble, France, ACM (2003) 95–104
11. Wick, K.: Sazba matematických vzorců čtyřřádkovým způsobem. Czechoslovak Academy of Sciences, Prague, Czech Republic (1961)

Security, Privacy and Trust in (Semantic)Web*

Roman Špánek

Institute of Computer Science,
Academy of Sciences of the Czech Republic,
Pod Vodarenskou vezi 2,
Czech Republic,
spanek@cs.cas.cz;
Technical University of Liberec,
Halkova 6,
Czech Republic,
roman.spanek@vslib.cz

Abstract. This paper gives a short overview on security issues widely found in the Semantic Web environment. It goes through each level of the proposed Semantic Web layers and discusses security, privacy and trust for each. Then, a list of possible solutions is given. In particular XML security, RDF security, secure information integration and trust on the Semantic Web are mentioned and short discussion is given. Finally, an approach for treating security and trust based on Virtual organization is described and its advantages are provided.

1 Introduction

With increasing number of users connected by the Internet, the demands for the Internet functionality is permanently increasing. At the beginning, the Internet was a simple computer network designed to enable users communicate, publish their information and many other but simple functionalities. Nevertheless, the great success of this architecture in the mid 1990s led to the demand for more efficient and sophisticated tools for searching and accessing all the data available on the Internet. At first some less or more efficient searching engines were developed allowing users to find related data. Nevertheless even with nowadays highly sophisticated search engines (e.g. Google), it is very difficult to share and even search so huge amount of data nowadays available on the Internet. Many other issues have arisen during the time and task like seamless data integration between multiple data sources or secure data sharing across geographically

* The work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization", partly by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications" and by Ministry of Education, Youth and Sports of the Czech Republic, project No. 1M4674788502 – Advanced Remedial Technology and Processes.

dispersed organizations and also individuals have emerged. Tim Berners Lee, in reflection to the inadequacies of current Web technologies, proposed architecture that makes the Web more intelligent. His goal was to isolate the humans from data integration and manipulation which naturally led him to the thought of a machine processable Web pages (data) and the usage of ontologies for information integration. This resulted in the notion of the semantic Web [1].

The semantic Web can be thought of as an "intelligent" and sophisticated Web where one needs little or no human intervention to carry out tasks such as scheduling appointments, coordinating activities, searching for documents, and integrating disparate databases and information systems.

The semantic Web has been a hot topic for quite a long time and many researches have been working on issues of various topics connected not only to the idea of the Semantic Web. Moreover, the World Wide Web consortium (W3C) acknowledged the Semantic Web and they have also been working on standards for the Semantic Web. Although these standards mainly include specifications or recommendations for XML, RDF, and also for Interoperability, the Semantic Web must be also secure. As security might be a broad term, and it surely is, in the context of the Semantic Web the security copes with secured integration, secure data exchange and security held at all layers (see next Section 2) of the Semantic Web. Moreover, the security of the Semantic Web should also cope with trust.

This paper summarizes current standards proposed for semantic Web in Section 2, based on well known layered structure proposed by Tim Berners Lee. Section 3, on the contrary, describes security standards and proposals for the Semantic Web. In the following Section 4 we present an alternative security solution based on a reputation system.

2 Standards for the Semantic Web

This section gives a brief overview on standards available for the Semantic Web. As was mentioned in the introduction, Tim Berners Lee proposal of the Semantic Web contains of layers, which can be thought of as being isolated from the others. In Figure 1 one can see all proposed layers. At the bottom communication protocols including TCP/IP (Transmission Control Protocol/Internet protocol), HTTP (Hypertext Transfer Protocol) and SSL (Secure Socket Layer) are shown. This layer is mainly responsible for supporting communication among different sites in efficient manner. At this level, neither syntax nor the semantics is dealt with. The only purpose of this layer is to enable transfer of a Web page over the Internet.

The second layer contains XML (eXtensible Markup Language) and XML schema. As was already mentioned, the main idea of Berners Lee was to give a computers ability to automatically process data (Web pages, documents, etc.) ubiquitously found on the Internet. To enable this, a need for well defined language is obvious. XML is a markup language that follows certain rules and if all documents are marked up with XML then there is uniform representation

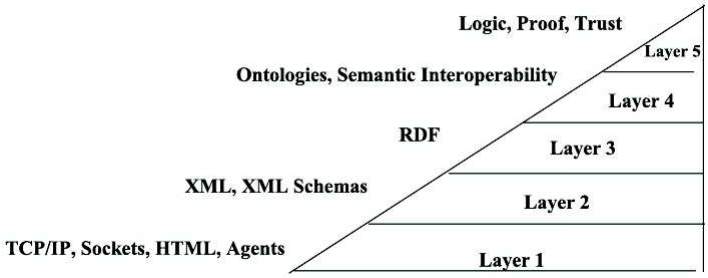


Fig. 1. Semantic Web layers

and presentation of documents. Therefore, the XML is the choice of common language for representing data on the Internet. XML schema main purpose is to describe the structure of XML documents, so that validators can check whether particular documents follows rules.

Above the level of XML and XML schema, one can find the level of RDF (Resource Description Framework). While XML focuses mainly on the syntax of a document, it is possible to have different interpretations of the same document at different sites. This is one of the major issues for seamless information integration. Led by the need for solution to this issue, W3C started discussions on a language called RDF in the late 1990s. RDF properties may be thought of as attributes of resources and in this sense corresponding to traditional attribute-value pairs. RDF properties also represent relationships between resources. RDF, however, provides no mechanisms for describing these properties, nor it does not provide any mechanisms for describing the relationships between these properties and other resources. That is the role of the RDF vocabulary description language, RDF Schema. RDF Schema defines classes and properties that may be used to describe classes, properties and other resources. RDF is vocabulary description language, RDF Schema, is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF schemas are also written as an XML documents.

Next come the Ontology layer. While RDF is only a specification language for expressing syntax and semantics, some questions concerning things like

- what entities do we need to specify?
- how can the community accept common definitions?
- etc.

may arrise. An ontology defines terms used to describe and represent an area of knowledge domain. Ontologies can be used by various communities and also applications that need to share domain information (a domain is just a specific subject area or area of knowledge). Ontologies include computer-usable definitions of basic concepts in the domains and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains, so that knowledge is made reusable. Ontologies are critical for applications that want to

search information from diverse communities. Although XML DTDs and XML Schemas are sufficient for exchanging data between parties who have agreed to definitions beforehand, their lack of semantics prevent machines from reliably performing this task given new XML vocabularies. However, in order to achieve interoperation between numerous, autonomously developed and managed schemas, richer semantics are needed. Note that even Ontologies may be specified by RDF (XML) syntax .

Finally, the highest level copes with logic, proof and trust. The question, that reader has probably cognizance about, is how one can trust information on the (Semantic)Web? Answer obviously depends from whom the information comes from. This leads us to the another question: "How to support trust?". The simplest way, how to build trust between interested parties is to allow them communicate and based on the communication somehow determine trust. While trust issues are closely related to security issues, it is discussed in the next section.

3 Security Standards for the Semantic Web

Having done overview for all[1] layers of the Semantic Web, it is time to summarize the security threats and issues related to the semantic Web environment. A clever way is to describe the security at layer by layer. As was mentioned earlier, logic, proof and trust are at the highest layer of the Semantic Web. However one cannot think about security in isolation. That is, no one layer, but at all of them, security should be solved. Therefore, in Figure 2 the same situation is shown, but now considering security aspects.

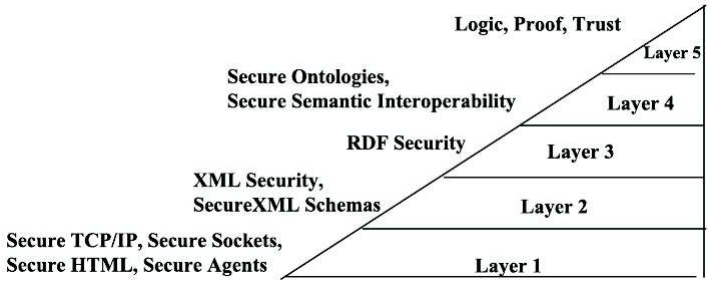


Fig. 2. Semantic Web layers with security emphasized

At the bottom level, protocols for transferring data from point to point are listed. At this level, more security options are available. This is mainly due to the fact that these protocols are used by the current Web. These include secure TCP/IP, secure sockets, and secure HTTP.

[1] attentive reader should object that the layers were not complete while at the very bottom layer is Unicode and URI.

On the other hand, securing the transmission is not enough for security in the case of managing access to the resources available. Therefore, a need for secure XML and XML schema is obvious. That is, access must be controlled to various portions of the document for reading, browsing and modifications. Various research efforts have been reported on XML security (see for example, [2]). Before we proceed to details , it should be clear that XML documents can be thought of as a tree structure. Hence, one of the main challenges is decide whether to grand access to the entire XML document or to only a part of it. An authorization models for XML was proposed by Bertino et al. They have focused on access control policies as well as on dissemination policies. The policy specification contains information what is required from users to grand access to which portions of the document. Besides the others, W3C (World Wide Web Consortium) has also specified standards for XML security. Particularly, the XML security project copes with the implementation of security standards for XML. The focus here is on XML-Signature Syntax and Processing, XML-Encryption Syntax and Processing and XML Key Management.

As at the next layer is RDF and RDF schema, the issue is to secure them both. As RDF (RDFS) is an XML document one can think that what was proposed for XML layer should be usable either. But that is obviously not the case, while RDF copes with interpretations and semantics. The issues here include the security implications of the concepts, properties and statements. The research in this area is at the beginning at the moment.

Ontologies, the net layer in the Berners Lee concept of the Semantic Web, should be also secured. The security here means mark some parts of an ontology as classified while certain other parts as unclassified. Closely related to security is privacy. Privacy requires some techniques enabling set certain portions of the document as private while certain other portions public or semi-private. Privacy in the Semantic Web is a really big issue, while utilizing advantages of the Semantic Web and in the same way maintaining privacy and anonymity is a serious problem. Note that W3C is actively examining privacy issues and as a good starting point one can have a look at P3P (Platform for Privacy Preferences) standards.

Important point from the previous column is that security and related issues cannot be thought of as afterthought. Security has to be preserved and solved at each particular level at the time of designing proposals for this levels. It is a severe mistake to propose solutions and then try to solve security in ad-hoc manner.

4 Reputation System for Privacy and Trust in (Semantic)Web Environment

During the previous section we gave an overview on technologies and proposals for the Semantic Web ranked according to the layers of the Semantic Web. It was also mentioned that trust is very fundamental point in building secure (semantic)Web. Therefore, this section discusses approaches for managing trust.

Nowadays, approaches currently available for the *trust management* can be structured as follows.

Policy-based approach has been proposed in the context of open and distributed services architectures [3–7] as well as in the context of Grids [8] as a solution to the problem of authorization and access control in open systems. Its focus is on trust management mechanisms employing different policy languages and engines for specifying and reasoning on rules for trust establishment.

Reputation-based approach has emerged in the context of electronic commerce systems, e.g. eBay. In distributed settings, reputation-based approaches have been proposed for managing trust in public key certificates, in P2P systems, mobile ad-hoc networks, and recently, in the Semantic Web, such as [9–14]. Typically, reputation-based trust is used in distributed networks where any involved entity has only a limited knowledge about the whole network. In this approach, the reputation is based on recommendations and experiences of other users/sites.

4.1 SecGRID

Here we present our approach for managing trust in a distributed environment. Firstly, note that users in a Computational Grid are usually organized in *Virtual Organizations* (VOs). A Virtual Organization is a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that pool resources, services and information to achieve common objectives and that have precisely described mechanisms and rules when and what to share. Dynamic Virtual Organizations Membership and structure of such a VO may evolve over time to accommodate changes in requirements or to adapt to new opportunities in the business environment.

It should be clear that the structure of VO from grids is very useful in other forms of distributed environments, like Mobile databases, peer-to-peer systems and also the Semantic Web.

Motivation Our motivation can be formalized as: "The goal is to provide a set of consistent, self-organizing and self-monitoring algorithms for creation and evolution of a dynamic, scalable VO with a distributed management used for treating security issues related to the trust between users. In addition the VO should not degenerate to any of the limiting cases which are one huge VO on one side and many small VOs on the other side."

So, the whole proposal should consist of the following steps:

- *Establishment of basis VO structure.* During this step a structure of groups of users fulfilling a criterion is created.
- *Adaptation of the structure accordingly to changes.* This step is responsible for the most important aspect of our proposal-it enables self-evolution of the structure while preserving its security.
- *Complexities (space, time) should be minimal.*

Mathematical Model The weighted hypergraph is a generalization of the concept of weighted graph which, in addition, allows an edge incidence to be more than two. Formally, the hypergraph $\mathcal{H}$ is a quadruple (U, N, W_U, W_N), where U is its set of nodes, $N \in 2^V$ is its edge set (nets), and W_U and W_N present node and edge weights, respectively.

Note that, in some applications, the incidence of nodes and edges is explicitly expressed by so-called node and edge connectors [15]. Here we prefer the classical definition [16] given above which is suitable for our purposes. Most of the important concepts from graphs can be easily generalized to hypergraphs. The weights may express further properties of graphs and hypergraphs, in addition to their connections. For instance, they can express quality or priority of communication, cost issues, efficiency of evaluations in individual nodes. As we will emphasize later, in our application related to security in VOs, the node weights will express *reliability* of users, and edge weights correspond to *security of node connections*.

The basic assumption of our model is that it is not centralized but distributed. Most of data is stored locally, but we have to store some more global VO properties as well. In particular, small to medium VOs may assume that individual nodes store:

a) node related information,
b) their adjacency sets
c) weights of incident (hyper)edges

depending on the size of the their local memories. Larger VOs may need a domain-based distribution [17] for faster communication even if the local memories are relatively large. This domain-based distribution will be in our implementation represented by a concept of VO leader explained below. The following paragraphs mention some application related issues which help to reflect specific model features.

Application Our original implementation is tightly connected to clique representation of a hypergraph which uses quotient graph model but here we will not go into details. We strongly believe that an efficient implementation of the quotient graph is one of the most suitable choices. Basic properties of dynamic changes in the quotient graph model were studied in [18] and implemented in a couple of extremely optimized implementations also [19]. The basic building blocks of the approach include the following items. *Clique representation* of the hypergraph which is an efficient way to capture hyperedges by a standard sparse compressed data structures used for storing sparse graphs. Namely, groups of nodes which are mutually interconnected, and correspond thus to an hyperedge, are stored implicitly as a list of involved nodes. *Edge absorption* is a technique to optimize number of hyperedges currently treated by the model. In some cases it can happen that an edge becomes a part of another edge. In this case, the former edge is absorbed.

Edge evaluation Edge weights play very important role in the *dynamic* behavior of VOs. To enable dynamics of the VO structure, we identified possible cases that might occur during the evolution of the structure. Based on this cases we proposed a set of rules and procedures that preserve the overall security. Due to space limitations we will not discuss them in details.

Optimization and Self-monitoring There is a strong need to preserve the security management optimal both in terms of time and space. Our implementation makes use of triggers in order to evaluate and reevaluate the role of nodes in the VO. Two main types of triggers are used. *Scheduled trigger* inspects the structure and optimizes VO in the terms of space utilization and computation load. *On customer request trigger* is a way how a customer "personalization" is achieved.

Experimental Results This chapter presents experimental results obtained from our application SECGRID written in ANSI C. Our current implementation is sequential, but intended to be efficiently implemented in a message passing environment. If we start with a hypergraph instead of its creation steps by steps we need to find its connected components/VOs. For this purpose, we use the Tarjan's algorithm [20] with improvements proposed in [21].

5 Conclusions

The paper overviews the layers of the semantic Web as were proposed by the father of the (Semantic)Web, Tim Berners Lee. The overview is structured so that the first part takes all layer into account and lists all related standards and proposals. On the contrary, the second part is intended to be an overview on the very same layers but taken from the security point of view. During this part standards and proposals for each layer are listed. The last part contains our proposal for treating security issues in the Semantic Web. The proposal is based on therm Virtual Organizations broadly known from (computational) Grids. Its aim is to have a system for treating trust between parities and enable dynamics of the VO that has a strong mathematical background giving it efficient and stable implementation. The mathematical model of hypergraphs is proposed and discussed from both theoretical and practical point of view. An experimental implementation SebcGRID is mentioned at the end.

References

1. Lee, T. B., et al.: The semantic web. Scientific American (may, 2001) 34–43
2. Bertino, E., et al.: Access control for xml documents. Data and Knowledge Engineering, North Holland (2002) 237260
3. Bonatti, P. and Samarati, P.: Regulating service access and information release on the web. In CCS 00: Proceedings of the 7th ACM conference on computer and communications security, ACM Press 134143

4. Li, N. and Mitchell, J.: A role-based trust-management framework. In DARPA Information Survivability Conference and Exposition (DISCEX), Washington, D.C. (Apr. 2003)
5. Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K. E. and Winslett, M.: No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In 1st European Semantic Web Symposium (ESWS 2004), Heraklion, Crete, Greece, Springer 3053 of Lecture Notes in Computer Science (may 2004) 342356
6. Becker, M. Y. and Sewell, P.: Cassandra: distributed access control policies with tunable expressiveness. In 5th IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights (June 2004)
7. Bonatti, P. A. and Olmedilla, D.: Driving and monitoring provisional trust negotiation with metapolicies. (jun 2005) 14–23
8. Basney, J., Nejdl, W., Olmedilla, D., Welch, V. and Winslett, M.: Negotiating trust on the grid. In 2nd WWW Workshop on Semantics in P2P and Grid Computing, New York, USA (may 2004)
9. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In Proceedings of 10th International Conference on Information and Knowledge Management (2001) 310317
10. Damiani, E., di Vimercati, S. D. C., Samarati, S. P. P. S. and Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In Proceedings of ACM Conference on Computer and Communications Security (2002) 202216
11. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: Eigenrep: Reputation management in p2p networks. In Proceedings of 12th International WWW Conference (2003) 640651
12. Duma, C., Shahmehri, N., Caronni, G.: Dynamic trust metrics for peer-topeer systems. In Proceedings of 2nd IEEE Workshop on P2P Data Management, Security and Trust (in connection with DEXA05) (August 2005)
13. Kagal, L., Finin, T., Joshi, A.: A policy based approach to security for the semantic web. In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA (Oct. 2003)
14. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: Semantic web languages for policy representation and reasoning: A comparison of kaos, rei and ponder. In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA (Oct. 2003)
15. Webpage: Hyper graphs: http://perso.wanadoo.fr/universimmedia/topicmaps/-hypergraph/hg4tm.htm
16. Golumbic, M.: Algorithmic graph theory and perfect graphs. Academic Press (1980)
17. Selvakkumaran, N., Karypis, G.: Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. IEEE Transactions on Computer-aided design (2005)
18. George, A., Liu, J.: A fast implementation of the minimum degree algorithm using quotient graphs. ACM Trans. Math. Software **6** (1980) 337–358
19. George, A., Liu, J.: Computer Solution of Large Sparse Positive Definite Systems. Prentice Hall (1981)
20. Tarjan, R.: Depth first search and linear graph algorithms. SIAM Jurnal of Computing **1(2)** (June 1972) 146–160
21. Nuutila, E., Soisalon-Soininen, E.: On finding the strongly connected components in a directed graph. Information Processing Letters **49(1)** (1994)

Statistics on The Real XML Data*

Kamil Toman and Irena Mlynkova

Charles University
Faculty of Mathematics and Physics
Department of Software Engineering
Malostranske nam. 25
118 00 Prague 1, Czech Republic
{kamil.toman,irena.mlynkova}@mff.cuni.cz

Abstract. At present the eXtensible Markup Language (XML) is used almost in all spheres of human activities. We can witness a massive boom of techniques for managing, querying, updating, exchanging, or compressing XML data.
On the other hand, for majority of the XML processing techniques we can find various spots which cause worsening of their time or space efficiency. Probably the main reason is that most of them consider XML data too globally, involving all their possible features, though the real data are often much simpler. If they do restrict the input data, the restrictions are often unnatural.
We discuss the level of complexity of real XML collections and their schemes, which turns out to be surprisingly low. We involve and compare results and findings of existing papers on similar topics as well as our own analysis and we try to find the reasons for these tendencies and their consequences.

1 Introduction

Currently XML and related technologies [7] have already achieved the leading role among existing standards for data representation. They are popular for various reasons, but especially because they enable to describe the allowed structure of XML documents using powerful tools such as DTD [7] or XML Schema [9, 16, 6]. Thus we can witness a massive boom of various XML techniques for managing, processing, exchanging, querying, updating, and compressing XML data that mutually compete in speed, efficiency, and minimum space and/or memory requirements.

On the other hand, for majority of the techniques we can find various critical spots which cause worsening of their time and/or space efficiency. In the worst and unfortunately quite often case such bottlenecks negatively influence directly the most interesting features of a particular technique.

If we study the bottlenecks further, we can distinguish two typical problematic situations. Firstly, we can distinguish a group of general techniques that

* This work was supported in part by the National Programme of Research (Information Society Project 1ET100300419).

take into account all possible features of input XML data – an approach that is at first glance correct. Nevertheless the standards were proposed as generally as possible enabling future users to choose what suits them most, whereas the real XML data are usually not as "rich" as they could be – they are often surprisingly simple. Thus the effort spent on every possible feature is mostly useless and it can even be harmful.

Secondly, there are techniques that somehow do restrict features of the input XML data. Hence it is natural to expect the bottlenecks to occur only in situations when given data do not correspond to the restrictions. But the problem is that such restrictions are often "unnatural". They do not result from features of real XML data collections but from other, more down-to-earth, reasons, e.g. limitations of the basic proposal of a particular technique, complexity of such solution etc.

A solution to the given problems could be a detailed analysis of real XML data and their classification.

2 Analyses and Results

For structural analysis of XML data it is natural to view XML documents as ordered trees and DTDs or XSDs (i.e. XML Schema definitions) as sets of regular expressions over element names. Attributes are often omitted for simplicity. We use notation and definitions for XML documents and DTDs/XSDs from [8] and [5].

Up to now several papers have focused on analysis of real XML data. They analyze either the structure of DTDs, the structure of XSDs, the structure of XML data regardless their schema, or the structure of XML documents in relation to corresponding schema. The sample data usually essentially differ.

2.1 DTD vs. XML Schema

With the arrival of XML Schema, as the extension of DTD, has arisen a natural question: Which of the extra features of XML Schema not allowed in DTD are used in practise? Paper [5] is trying to answer it using analysis of 109 DTDs and 93 XSDs. Another aim of the paper is to analyze the real structural complexity for both the languages, i.e. the degree of sophistication of regular expressions used.

The former part of the paper focuses on analysis of XML Schema features. The features and their resulting percentage are:

- extension[1] (27%) and restriction (73%) of simple types,
- extension (37%) and restriction (7%) of complex types,
- `final` (7%), `abstract` (12%), and `block` (2%) attribute of complex type definitions,

[1] Extension of a simple type means adding attributes to the simple type, i.e. creating a complex type with simple content.

– substitution groups (11%),
– unordered sequences of elements (4%),
– `unique` (7%) and `key`/`keyref` (4%) features,
– namespaces (22%), and
– redefinition of types and groups (0%).

As it is evident, the most exploited features are restriction of simple types, extension of complex types, and namespaces. The first one reflects the lack of types in DTD, the second one confirms the naturalness of object-oriented approach (i.e. inheritance), whereas the last one probably results from mutual modular usage of XSDs. The other features are used minimally or are not used at all.

Probably the most interesting finding is, that 85% of XSDs define so called *local tree languages* [14], i.e. languages that can be defined by DTDs as well, and thus that the expressiveness beyond local tree grammars is needed rarely.

2.2 XML Document Analysis

Previously mentioned analyses focused on descriptions of the allowed structure of XML documents. By contrast paper [12] (and its extension [2]) analyzes directly the structure of their instances, i.e. XML documents, regardless eventually existing DTDs or XSDs.[2] It analyzes about 200 000 XML documents publicly available on the Web, whereas the statistics are divided into two groups – statistics about the Web and statistics about the XML documents.

The Web statistics involve:

– clustering of the source web sites by zones consisting of Internet domains (e.g. *.com*, *.edu*, *.net* etc.) and geographical regions (e.g. Asia, EU etc.),
– the number and volume (i.e. the sum of sizes) of documents per zone,
– the number of DTD (48%) and XSD (0.09%) references,
– the number of namespace references (40%),
– distribution of files by extension (e.g. *.rdf*, *.rss*, *.wml*, *.xml* etc.), and
– distribution of document *out-degree*, i.e. the number of `href`, `xmlhref`, and `xlink:href` attributes.

Obviously most of them describe the structure of the XML Web and categories of the source XML documents.

Statistics about the structure of XML documents involve:

– the size of XML documents (in bytes),
– the amount of markup, i.e. the amount of element and attribute nodes versus the amount of text nodes and the size of text content versus the size of the structural part,
– the amount of mixed content elements,

[2] The paper just considers whether the document does or does not reference a DTD or an XSD.

- the depth of XML documents and the distribution of node types (i.e. element, attribute, or text nodes) per level,
- element and attribute fan-out
- the number of distinct strings, and
- recursion.

The most interesting findings of the research are as follows:

- The size of documents varies from 10B to 500kB; the average size is 4,6kB.
- For documents up to 4kB the number of element nodes is about 50%, the number of attribute nodes about 30%. Surprisingly, for larger documents the number of attribute nodes rises to 50%, whereas the number of element nodes declines to 38%. The structural information still dominates the size of documents.
- Although there are only 5% of all elements with mixed content, they were found in 72% of documents.
- Documents are relatively shallow – 99% of documents have fewer than 8 levels, whereas the average depth is 4.
- The average element fan-out for the first three levels is 9, 6, and 0.2; the average attribute fan-out for the first four levels is 0.09, 1, 1.5, and 0.5. Surprisingly, 18% of all elements have no attributes at all.

A great attention is given to recursion which seems to be an important aspect of XML data. The authors mention the following findings:

- 15% of all XML documents contain recursive elements.
- Only 260 distinct recursive elements were found. In 98% of recursive documents there is only one recursive element used.
- 95% of recursive documents do not refer to any DTD or XSD.
- Most elements in ed pairs have the distance up to 5.
- The most common average fan-outs are 1 (60%) and 2 (37%), the average recursive fan-out is 2.2.

Last mentioned paper [11] that focuses on analysis of XML documents consists of two parts – a discussion of different techniques for XML processing and an analysis of real XML documents. The sample data consists of 601 XHTML web pages, 3 documents in DocBook format[3], an XML version of Shakespeare's plays[4] (i.e. 37 XML documents with the same simple DTD) and documents from *XML Data repository* project[5]. The analyzed properties are the maximum depth, the average depth, the number of simple paths, and the number of unique simple paths; the results are similar to previous cases.

[3] `http://www.docbook.org/`
[4] `http://www.ibiblio.org/xml/examples/shakespeare/`
[5] `http://www.cs.washington.edu/research/xmldatasets/`

2.3 XML Documents vs. XML schemes

Paper [13] takes up work initiated in the previously mentioned articles. It enhances the preceding analyses and defines several new constructs for describing the structure of XML data (e.g. DNA or relational patterns). It analyzes XML documents and their DTDs or XSDs eventually that were collected semi-automatically with interference of human operator. The reason is that automatic crawling of XML documents generates a set of documents that are unnatural and often contain only trivial data which cause misleading results. The collected data consist of about 16 500 XML documents of more than 20GB in size, whereas only 7.4% have neither DTD nor XSD. Such low ratio is probably caused by the semi-automatic gathering.

The data were first divided into following six categories:

- *data-centric documents*, i.e. documents designed for database processing (e.g. database exports, lists of employees etc.),
- *document-centric documents*, i.e. documents which were designed for human reading (e.g. Shakespeare's plays, XHTML [1] documents etc.)
- *documents for data exchange* (e.g. medical information on patients etc.),
- *reports*, i.e. overviews or summaries of data (usually of database type),
- *research documents*, i.e. documents which contain special (scientific or technical) structures (e.g. protein sequences, DNA/RNA structures etc.), and
- *semantic web documents*, i.e. RDF [4] documents.

The statistics described in the paper are also divided into several categories. They were computed for each category and if possible also for both XML documents and XML schemes and the results were compared. The categories are as follows:

- *global statistics*, i.e. overall properties of XML data (e.g. number of elements of various types such as empty, text, mixed, recursive etc., number of attributes, text length in document, paths and depths etc.),
- *level statistics*, i.e. distribution of elements, attributes, text nodes, and mixed contents per each level,
- *fan-out statistics*, i.e. distribution of branching per each level,
- *recursive statistics*, i.e. types and complexity of recursion (e.g. exploitation rates, depth, branching, distance of ed-pairs etc.),
- *mixed-content statistics*, i.e. types and complexity of mixed contents (e.g. depth, percentage of simple mixed contents etc.),
- *DNA statistics*, i.e. statistics focussing on DNA patterns (e.g. number of occurrences, width, or depth), and
- *relational statistics*, i.e. statistics focussing on both relational and shallow relational patterns (e.g. number of occurrences, width, or fan-out).

Most interesting findings and conclusions for all categories of statistics are as follows:

- The amount of tagging usually dominates the size of document.

- The lowest usage of mixed-content (0.2%) and empty (26.8%) elements can be found for data-centric documents.
- The highest usage of mixed-content elements (77%) can be found for document-centric documents.
- Documents of all categories are typically shallow. (For 95% of documents the maximum depth is 13, the average depth is about 5.)
- The highest amounts of elements, attributes, text nodes, and mixed contents as well as fan-outs are always at first levels and then their number of occurrences rapidly decreases.
- Recursion is quite often, especially in document-centric (43%) and exchange (64%) documents, although the number of distinct recursive elements is typically low (for each category less than 5).
- Recursion, if used, is rather simple – the average depth, branching as well as distance of ed-pairs is always less than 10.
- The most common types of recursion are linear (20% for document-centric and 33% for exchange documents) and pure (19% for document-centric and 23% for exchange documents). On the other hand almost all schemes specify mostly general type of recursion.
- The percentage of simple mixed contents is relatively high (e.g. 79% for document-centric or even 99% for exchange documents) and thus the depth of mixed contents is generally low (on the average again less than 10).
- The number of occurrences of DNA patterns is quite high, especially for research, document-centric, and exchange documents. On the other hand the average depth and width is always low (less than 7).
- The number of occurrences of relational patterns is quite high, especially for semantic-web, research, and exchange documents. The complexity (i.e. depth and width) is again quite low.
- XML schemes usually provide too general information, whereas the instance documents are much specific and simpler.

2.4 Discussion

The previous overview of existing analyses of XML data brings various interesting information. In general, we can observe that the real complexity of both XML documents and their schemes is amazingly low.

Probably the most surprising findings are that recursive and mixed-content elements are not as unimportant as they are usually considered to be. Their proportional representation is more than significant and in addition their complexity is quite low. Unfortunately, effective processing of both the aspects is often omitted with reference to their irrelevancy. Apparently, the reasoning is false whereas the truth is probably related to difficulties connected with their processing.

Another important discovery is that the usual depth of XML documents is small, the average number is always less than 10. This observation is already widely exploited in techniques which represent XML documents as a set of points in multidimensional space and store them in corresponding data structures, e.g.

R-trees, UB-trees [3], BUB-trees [10] etc. The effectiveness of these techniques is closely related to the maximum depth of XML documents or maximum number of their simple paths. Both of the values should be of course as small as possible.

Next considerable fact is that the usage of schemes for expressing the allowed structure of XML documents is not as frequent as it is expected to be. The situation is particularly wrong for XSDs which seem to appear sporadically. And even if they are used, their expressive power does not exceed the power of DTDs. The question is what is the reason for this tendency and if we can really blame purely the complexity of XML Schema. Generally, the frequent absence of schema is of course a big problem for methods which are based on its existence, e.g. schema-driven database mapping methods [15].

Concerning the XML schemes there is also another important, though not surprising finding, that XML documents often do not fully exploit the generality allowed by schema definitions. It is striking especially in case of types of recursion but the statement is valid almost generally. (Extreme cases are of course recursion that theoretically allows XML documents with infinite depth or complete subgraphs typical for document-centric XML documents.) This observation shows that although XML schemes provide lots of structural information on XML documents they can be too loose or even inaccurate.

The last mentioned analysis indicates, that there are also types of constructs (such as simple mixed contents, DNA patterns, or relational patterns etc.), that are quite common and can be easily and effectively processed using, e.g., relational databases. Hence we can expect that a method that focuses on such constructs would be much effective than the general ones.

Last but not least, we must mention the problem of both syntactic and semantic incorrectness of analyzed XML documents, DTDs, and XSDs. Authors of almost all previously mentioned papers complain of huge percentage of useless sample data – an aspect which unpleasantly complicates the analyses. A consequent question is whether we can include non-determinism and ambiguity into this set of errors or if it expresses a demand for extension of XML recommendations.

3 Conclusion

The main goal of this paper was to briefly describe, discuss, and classify papers on analyses of real XML data and particularly their results and findings. The whole overview shows that the real data show lots of regularities and pattern usages and are not as complex as they are often expected to be. Thus there exists plenty of space for improvements in XML processing based on this enhanced categorization.

References

1. The Extensible HyperText Markup Language (Second Edition). W3C Recommendation, August 2002. `http://www.w3.org/TR/xhtml1/`.

2. Barbosa, D., Mignet, L. and Veltri, P.: Studying the XML Web: Gathering Statistics from an XML Sample. In World Wide Web, Hingham, MA, USA, Kluwer Academic Publishers (2005) 413–438
3. Bayer, R.: The Universal B-Tree for Multidimensional Indexing: General Concepts. In WWCA '97, Worldwide Computing and Its Applications, International Conference, Tsukuba, Japan, Springer (1997) 198–209
4. Beckett, D.: RDF/XML Syntax Specification (Revised). W3C Recommendation (February 2004) `http://www.w3.org/TR/rdf-syntax-grammar/`.
5. Bex, G. J., Neven F. and Van den Bussche J.: DTDs versus XML Schema: a Practical Study. In WebDB '04, Proceedings of the 7th International Workshop on the Web and Databases, New York, NY, USA, ACM Press (2004) 79–84
6. Biron, P. V. and Malhotra, A.: XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (October 2004) `www.w3.org/TR/xmlschema-2/`.
7. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler E. and Yergeau, F.: Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation (February 2004) `http://www.w3.org/TR/REC-xml/`.
8. Choi, B.: What are real DTDs like? In WebDB '02, Proceedings of the 5th International Workshop on the Web and Databases, pages 43–48, Madison, Wisconsin, USA, 2002. ACM Press.
9. Fallside, D. C. and Walmsley, P.: XML Schema Part 0: Primer Second Edition. W3C Recommendation, October 2004. `www.w3.org/TR/xmlschema-0/`.
10. Fenk, R.: The BUB-Tree. In VLDB '02, Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China, Morgan Kaufman Publishers (2002)
11. Kosek, J., Kratky, M. and Snasel, V.: Struktura realnych XML dokumentu a metody indexovani. In ITAT 2003 Workshop on Information Technologies Applications and Theory, High Tatras, Slovakia (2003) (in Czech).
12. Mignet, L., Barbosa, D. and Veltri, P.: The XML Web: a First Study. In WWW '03, Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press **2** (2003) 500–510
13. Mlynkova, I., Toman, K. and Pokorny, J.: Statistical Analysis of Real XML Data Collections. Technical report 2006/5, Charles University (June 2006) `http://kocour.ms.mff.cuni.cz/~mlynkova/doc/tr2006-5.pdf`.
14. Murata, M., Lee, D. and Mani, M.: Taxonomy of XML Schema Languages using Formal Language Theory. In Extreme Markup Languages, Montreal, Canada, 2001.
15. Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D. J. and Naughton, J. F.: Relational Databases for Querying XML Documents: Limitations and Opportunities. In VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, UK, Morgan Kaufmann (1999) 302–314
16. Thompson, H. S., Beech, D., Maloney, M. and Mendelsohn, N.: XML Schema Part 1: Structures Second Edition. W3C Recommendation (October 2004) `www.w3.org/TR/xmlschema-1/`.

Seznam autorů