Bulletin of the Technical Committee on

Data Engineering

March 2002 Vol. 25 No. 1

IEEE Computer Society

Letters

Special Issue on Organizing and Discovering the Semantic Web

Conference and Journal Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

The Data Engineering Bulletin web page is http://www.research.microsoft.com/research/db/debull.

Letter from the Editor-in-Chief

TCDE Meeting at ICDE'2002

The Technical Committee on Data Engineering held its annual meeting at its flagship conference, the International Conference on Data Engineering (ICDE'02) in San Jose, California on February 28. Minutes of this meeting, prepared by TCDE Chair Erich Neuhold are at
`http://www.research.microsoft.com/research/db/debull/minutes.doc`.

Editorial Changes

Every two years, I have the pleasure of appointing new editors for the Data Engineering Bulletin. This year, I am appointing all the editors in one fell swoop, rather than naming them incrementally over a number of issues. I am delighted that each of the following people have agreed to serve as Bulletin editors for the next two years:

Umesh Dayal of HP Labs, Palo Alto, CA. Umesh's career spans periods at CCA, and Digital as well as HP Labs. His interests include active databases, object oriented databases, workflow, data mining, and more.

Johannes Gehrke of Cornell University, Ithaca, NY. Johannes is graduate of that great database university, Wisconsin. Johannes' interests include data mining, sensor-based databases, and data streams.

Christian Jensen of the University of Aalborg, Denmark. Christian's work includes research in temporal databases, database design, query languages and most recently spatiotemporal databases.

Renee Miller of the University of Toronto, Canada. Rene is another Wisconsin graduate. Renee has focused her efforts in heterogeneous data management, metadata management, and data mining.

I look forward to working with the new editors in our continuing effort to provide the very latest information about on-going research and the very best industrial practice in the database area.

This is also, when I bid farewell to the current editors. **Luis Gravano, Alon Halevy, Sunita Sarawagi, and Gerhard Weikum** have all done the very capable jobs that we have come to expect of our editors. Putting together an issue of the Bulletin is a major undertaking. Success depends first on the editor knowing the subject area exceptionally well, and knowing who are the strong workers in the area. Then there is the period of convincing prospective authors to commit to submitting an article. The nagging stage is next, in which authors are "gently" reminded that their article is needed. And finally, there is the flurry of actual editting, rewriting, shortening, latex stumbling around, etc. leading up to the result that you, the reader actually sees. Luis, Alon, Sunita, and Gerhard have gone through this process twice in exemplary fashion. We all owe them a debt of gratitude, and I surely want to express my thanks to them as they retire as Bulletin editors.

The Current Issue

Database researchers struggle with the problem of how to capture, manipulate, and query information in a more meaningful, more "semantically rich", way. Last month's issue on text and databases was one example of this. The current issue reflects this struggle in the context of the now well-recognized importance of the web as an information source. Dealing with the "semantic web" will only get more important over time. Gerhard Weikum has successfully enticed some of the premier researchers in this area to contribute articles for this issue. The current issue very nicely captures some of the fascinating work going on in making sense of the web and the information stored on it. I want to thank Gerhard for once again doing a fine job on a very important topic.

David Lomet
Microsoft Corporation

Letter from the new TC Chair

Dear Colleagues,

Since January 2002 I am the new Chair of the IEEE Technical Committee on Data Engineering. First of all I would like to thank Betty Salzberg for her excellent work as the former Chair of the TCDE during her terms from 1998-2001. Happily she now agreed to serve as my Vice Chair. I would also like to thank the other members of the TC for their work. Since they have done and are doing great work, I have asked them and they have all agreed to work together with me in the new executive committee. Hence the executive committee is:

- Prof. Erich J. Neuhold (Chair)
- Betty Salzberg (Vice Chair)
- Per-Ake (Paul) Larson (Treasurer/Secretary)
- Svein-Olaf Hvasshovd (European Coordinator)
- Masaru Kitsuregawa (Asian Coordinator)
- Ron Sacks-Davis (Australian Coordinator)
- David Lomet (Bulletin Editor)
- Marianne Winslett (ACM SIGMOD Liaison and also SIGMOD Vice Chair)

The annual meeting of the TC this year was held during the ICDE 2002 in San Jose. The minutes of the meeting, for those who were not able to attend, can be found below. Beside the reports on finances and the Data Engineering Bulletin, the planning for the upcoming ICDE conferences for the years 2003-2006 has been presented. ICDE 2003 will be held in Bangalore, India, and ICDE 2004 in Boston, Massachusetts. In conjunction with the next ICDEs we would like to establish one or two workshops in addition to the traditional RIDE workshop. I invite everybody to send me proposals with your ideas. In addition, new ideas regarding conferences we should cooperate with, seminars to be organized, and general comments and expectations you have for the TC, are always welcome.

The WEB page (in multiple languages) of the TC on Data Engineering can be found at
`http://www.ccs.neu.edu/groups/IEEE/tcde/index.html`,
where information on the Executive Committee, sponsored and in-cooperation activities and a few other things can be obtained.

Erich J. Neuhold
Fraunhofer IPSI

Letter from the Special Issue Editor

It is almost a decade ago when the World Wide Web revolutionized our thinking about information organization and scale. We are now witnessing another major transition of our capabilities to cope with Internet-based information. Some people refer to this as the third-generation Internet, with the original infrastructure for e-mail and ftp-style data transfer being the first generation and the http- and html-based Web being the second generation. The envisoned Semantic Web is aiming to facilitate automated Web services, integrate the data world behind portals of the Deep Web, and provide the means for semantic interoperability, Internet-wide knowledge discovery, and more precise searching.

If and when this vision will come true or whether it will involve a gradual long-term process rather than another revolution, are completely open. The challenge involves two complementary research avenues: on one hand, more explicit structuring and richer metadata is required for information organization, on the other hand, more intelligent search capabilities need to be developed to cope with the extreme diversity of Internet information. Data warehousing, with its two facets of data cleaning and data mining, has taught us that this duality of information organization and information discovery calls for research on both ends. Information model integration and logic-based knowledge representation are as important as XML querying, Web crawling, and ranked retrieval. Consequently, the Semantic Web challenge requires cooperation between different research communities and synergies from different paradigms.

This pluralism is reflected in the articles compiled in this issue of the Data Engineering Bulletin: they span contributions from artificial intelligence, database technology, and information retrieval. The first two articles, by Ian Horrocks and by Alexander Maedche et al., discuss the role of description logics and ontologies for richer data representation and and for organizing Web portals. The third article, by Yannis Papakonstantinou and Vasilis Vassalos, examines the role of XML querying in a mediator for integrating heterogeneous data sources. The fourth paper, by Serge Abiteboul et al., addresses the scale and dynamics of the Web in efficiently computing Google-style authority scores. The fifth and sixth papers, by Soumen Chakrabarti et al. and by Luis Gravano et al., investigate the integration of crawling and automatic classification in order to construct thematic directories. The issue is concluded with the seventh article, by Norbert Fuhr and myself, on applying information retrieval techniques to XML data.

I hope you find the challenge of organizing and discovering the envisioned Semantic Web as exciting as I do, and I hope that the seven papers in this issue and the pointers provided by them are insightful and helpful for your work.

Gerhard Weikum
University of the Saarland
Saarbruecken, Germany

DAML+OIL: a Description Logic for the Semantic Web

Ian Horrocks
Department of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, UK
horrocks@cs.man.ac.uk

Abstract

Ontologies are set to play a key role in the "Semantic Web", extending syntactic interoperability to semantic interoperability by providing a source of shared, precisely defined terms. DAML+OIL is an ontology language specifically designed for use on the Web; it exploits existing Web standards (XML and RDF), adding the familiar ontological primitives of object oriented and frame based systems, and the formal rigor of a very expressive description logic. The logical basis of the language means that reasoning services can be provided, both to support ontology design and to make DAML+OIL described Web resources more accessible to automated processes.

1 Introduction

The World Wide Web has been made possible through a set of widely established standards which guarantee interoperability at various levels. For example, the TCP/IP protocol has ensured interoperability at the transport level, while HTTP and HTML have provided a standard way of retrieving and presenting hyperlinked text documents. Applications have been able to use this common infrastructure and this has made possible the World Wide Web as we know it now.

The "first generation" Web consisted largely of handwritten HTML pages. The current Web, which can be described as the second generation, has made the transition to machine generated and often active HTML pages. Both first and second generation Web were meant for direct human processing (reading, browsing, form-filling, etc.). The third generation aims to make Web resources more readily accessible to automated processes by adding meta-data annotations that describe their content. This idea was first delineated, and named the *Semantic Web*, in Tim Berners-Lee's recent book "Weaving the Web" [5].

If meta-data annotations are to make resources more accessible to automated agents, it is essential that their meaning can be understood by such agents. This is where *ontologies* will play a crucial role, providing a source of shared and precisely defined terms that can be used in meta-data. An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity obviously facilitates machine

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

understanding. Examples of the use of ontologies could include e-commerce sites [16], search engines [17] and Web services [19].

2 Web Ontology Languages

The recognition of the key role that ontologies are likely to play in the future of the Web has led to the extension of Web markup languages in order to facilitate content description and the development of Web based ontologies, e.g., XML Schema,[1] RDF[2] (Resource Description Framework), and RDF Schema [7]. RDF Schema (RDFS) in particular is recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations.

RDFS is, however, a very primitive language (the above is an almost complete description of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes, e.g., to determine the semantic relationship between syntactically different terms.

The recognition of these requirements has led to the development of DAML+OIL, an expressive Web ontology language. DAML+OIL is the result of a merger between DAML-ONT, a language developed as part of the US DARPA Agent Markup Language (DAML) programme[3]) and OIL (the Ontology Inference Layer) [9], developed by a group of (mostly) European researchers.[4]

3 DAML+OIL and Description Logics

DAML+OIL is designed to describe the *structure* of a domain; it takes an object oriented approach, describing the structure in terms of *classes* and *properties*. An ontology consists of a set of *axioms* that assert, e.g., subsumption relationships between classes or properties. Asserting that resources[5] (pairs of resources) are instances of DAML+OIL classes (properties) is left to RDF, a task for which it is well suited. When a resource r is an instance of a class C we say that r has type C.

From a formal point of view, DAML+OIL can be seen to be equivalent to a very expressive description logic (DL), with a DAML+OIL ontology corresponding to a DL terminology (Tbox). As in a DL, DAML+OIL classes can be names (URIs) or *expressions*, and a variety of *constructors* are provided for building class expressions. The expressive power of the language is determined by the class (and property) constructors supported, and by the kinds of axiom supported.

Figure 1 summarises the constructors supported by DAML+OIL. The standard DL syntax is used for compactness as the RDF syntax is rather verbose. In the RDF syntax, for example, $\geq$2hasChild.Lawyer would be written as

```
<daml:Restriction daml:minCardinalityQ="2">
  <daml:onProperty rdf:resource="#hasChild"/>
  <daml:hasClassQ rdf:resource="#Lawyer"/>
</daml:Restriction>
```

[1] http://www.w3.org/XML/Schema/

[2] http://www.w3c.org/RDF/

[3] http://www.daml.org/

[4] http://www.ontoknowledge.org/oil

[5] Everything describable by RDF is called a resource. A resource could be Web accessible, e.g., a Web page or part of a Web page, but it could also be an object that is not directly accessible via the Web, e.g., a person. Resources are named by URIs plus optional anchor ids. See http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ for more details.

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
`complementOf`	$\neg C$	$\neg$Male
`oneOf`	$\{x_1 \ldots x_n\}$	$\{$john, mary$\}$
`toClass`	$\forall P.C$	$\forall$hasChild.Doctor
`hasClass`	$\exists P.C$	$\exists$hasChild.Lawyer
`hasValue`	$\exists P.\{x\}$	$\exists$citizenOf.$\{$USA$\}$
`minCardinalityQ`	$\geq n P.C$	$\geq$2hasChild.Lawyer
`maxCardinalityQ`	$\leq n P.C$	$\leq$1hasChild.Male
`cardinalityQ`	$=n\, P.C$	$=$1 hasParent.Female

Figure 1: DAML+OIL class constructors

The meaning of the first three constructors (`intersectionOf`, `unionOf` and `complementOf`) is relatively self-explanatory: they are just the standard boolean operators that allow classes to be formed from the intersection, union and negation of other classes. The `oneOf` constructor allows classes to be defined existentially, i.e., by enumerating their members.

The `toClass` and `hasClass` constructors correspond to slot constraints in a frame-based language. The class $\forall P.C$ is the class all of whose instances are related via the property P only to resources of type C, while the class $\exists P.C$ is the class all of whose instances are related via the property P to at least one resource of type C. The `hasValue` constructor is just shorthand for a combination of `hasClass` and `oneOf`.

The `minCardinalityQ`, `maxCardinalityQ` and `cardinalityQ` constructors (known in DLs as qualified number restrictions) are generalisations of the `toClass` and `hasClass` constructors. The class $\geq nP.C$ ($\leq nP.C$, $=n\, P.C$) is the class all of whose instances are related via the property P to at least (at most, exactly) n *different* resources of type C. The emphasis on different is because there is no unique name assumption with respect to resource names (URIs): it is possible that many URIs could name the same resource.

Note that arbitrarily complex nesting of constructors is possible. Moreover, XML Schema *datatypes* (e.g., so called primitive datatypes such as strings, decimal or float, as well as more complex derived datatypes such as integer sub-ranges) can be used anywhere that a class name might appear.

The formal semantics of the class constructors is given by DAML+OIL's model-theoretic semantics[6].

The other aspect of a language that determines its expressive power is the kinds of axiom supported. Figure 2 summarises the axioms supported by DAML+OIL. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals (resources), and various properties of properties.

A crucial feature of DAML+OIL is that `subClassOf` and `sameClassAs` axioms can be applied to arbitrary class expressions. This provides greatly increased expressive power with respect to standard frame-based languages where such axioms are invariably restricted to the form where the left hand side is an atomic name, there is only one such axiom per name, and there are no cycles (the class on the right hand side of an axiom cannot refer, either directly or indirectly, to the class name on the left hand side).

A consequence of this expressive power is that all of the class and individual axioms, as well as the `uniqueProperty` and `unambiguousProperty` axioms, can be reduced to `subClassOf` and `sameClassAs` axioms (as can be seen from the DL syntax). In fact `sameClassAs` could also be reduced to `subClassOf` as a `sameClassAs` axiom $C \equiv D$ is equivalent to a pair of `subClassOf` axioms, $C \sqsubseteq D$ and $D \sqsubseteq C$.

[6] `http://www.w3.org/TR/daml+oil-model`

Axiom	DL Syntax	Example
`subClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`sameClassAs`	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
`subPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`samePropertyAs`	$P_1 \equiv P_2$	cost $\equiv$ price
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
`sameIndividualAs`	$\{x_1\} \equiv \{x_2\}$	$\{$President_Bush$\}$ $\equiv$ $\{$G_W_Bush$\}$
`differentIndividualFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{$john$\}$ $\sqsubseteq$ $\neg\{$peter$\}$
`inverseOf`	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
`transitiveProperty`	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
`uniqueProperty`	$\top \sqsubseteq {\leq}1P$	$\top \sqsubseteq {\leq}1$hasMother
`unambiguousProperty`	$\top \sqsubseteq {\leq}1P^-$	$\top \sqsubseteq {\leq}1$isMotherOf$^-$

Figure 2: DAML+OIL axioms

As we have seen, DAML+OIL allows properties of properties to be asserted. It is possible to assert that a property is unique (i.e., functional), unambiguous (i.e., its inverse is functional) or transitive, as well as to use inverse properties.

4 Reasoning Services

As we have shown, DAML+OIL is equivalent to a very expressive description logic. More precisely, DAML+OIL is equivalent to the $\mathcal{SHIQ}$ DL [15] with the addition of existentially defined classes (i.e., the oneOf constructor) and *datatypes* (often called concrete domains in DLs [1]). This equivalence allows DAML+OIL to exploit the considerable existing body of description logic research, e.g.:

- to define the semantics of the language and to understand its formal properties, in particular the decidability and complexity of key inference problems [8];
- as a source of sound and complete algorithms and optimised implementation techniques for deciding key inference problems [15, 14];
- to use implemented DL systems in order to provide (partial) reasoning support [12, 20, 11].

A important consideration in the design of DAML+OIL was that key inference problems in the language, in particular class consistency/subsumption,[7] should be decidable, as this facilitates the provision of reasoning services. Moreover, the correspondence with DLs facilitates the use of DL algorithms that are known to be amenable to optimised implementation and to behave well in realistic applications in spite of their high worst case complexity [13, 10]. In particular, DAML+OIL is able to exploit highly optimised reasoning services provided by DL systems such as FaCT [12], DLP [20], and Racer [11], although these systems do not, as yet, support the whole DAML+OIL language (none is able to reason with existentially defined classes, i.e., the `oneOf` construct, or to provide support for all XML Schema datatypes).

Maintaining the decidability of the language requires certain constraints on its expressive power that may not be acceptable to all applications. However, the designers of the language decided that reasoning would be

[7] In propositionally closed languages like DAML+OIL, class consistency and subsumption are mutually reducible. Moreover, in DAML+OIL the consistency of an entire "knowledge base" (an ontology plus a set of class and property membership assertions) can be reduced to class consistency.

important if the full power of ontologies was to be realised, and that a powerful but still decidable ontology language would be a good starting point.

Reasoning can be useful at many stages during the design, maintenance and deployment of ontologies.

- Reasoning can be used to support ontology design and to improve the quality of the resulting ontology. For example, class consistency and subsumption reasoning can be used to check for logically inconsistent classes and (possibly unexpected) implicit subsumption relationships (as demonstrated in the OilEd[8] ontology editor [4]). This kind of support has been shown to be particularly important with large ontologies, which are often built and maintained over a long period by multiple authors. Other reasoning tasks, such as "matching" [3] and/or computing least common subsumers [2] could also be used to support "bottom up" ontology design, i.e., the identification and description of relevant classes from sets of example instances.

- Like information integration [6], ontology integration can also be supported by reasoning. For example, integration can be performed using inter-ontology assertions specifying relationships between classes and properties, with reasoning being used to compute the integrated hierarchy and to highlight any problems/inconsistencies. Unlike some other integration techniques (e.g., name reconciliation [18]), this method has the advantage of being non-intrusive with respect to the original ontologies.
- Reasoning with respect to deployed ontologies will enhance the power of "intelligent agents", allowing them to determine if a set of facts is consistent w.r.t. an ontology, to identify individuals that are implicitly members of a given class etc. A suitable service ontology could, for example, allow an agent seeking secure services to identify a service requiring a userid and password as a possible candidate.

5 Summary

DAML+OIL is an ontology language specifically designed for use on the Web; it exploits existing Web standards (XML and RDF), adding the formal rigor of a description logic. As well as providing the formal underpinnings of the language, the connection to DLs can be exploited as a source of algorithms and implementation techniques, and to provide (partial) reasoning support for DAML+OIL applications by using implemented DL systems.

DAML+OIL has already been widely adopted, with some major efforts having already committed to encoding their ontologies in the language. This has been particularly evident in the bio-ontology domain, where the Bio-Ontology Consortium has specified DAML+OIL as their ontology exchange language, and the Gene Ontology [21] is being migrated to DAML+OIL in a project partially funded by GlaxoSmithKline Pharmaceuticals in cooperation with the Gene Ontology Consortium.[9]

What of the future? The development of the semantic Web, and of Web ontology languages, presents many challenges. As we have seen, no DL system yet provides reasoning support for the full DAML+OIL language. Developing a "practical" satisfiability/subsumption algorithm (i.e., one that is amenable to highly optimised implementation) for the whole language would present a major step forward in DL (and semantic web) research. Moreover, even if such an algorithm can be developed, it is not clear if even highly optimised implementations of sound and complete algorithms will be able to provide adequate performance for typical web applications.

Acknowledgements

I would like to acknowledge the contribution of all those involved in the development of DAML-ONT, OIL and DAML+OIL, amongst whom Dieter Fensel, Frank van Harmelen, Jim Hendler, Deborah McGuinness and Peter F. Patel-Schneider deserve particular mention.

[8] `http://img.cs.man.ac.uk/oil`

[9] `http://www.geneontology.org/`.

References

[1] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.

[2] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of KI'98*, volume 1504 of *LNCS*, pages 129–140. Springer-Verlag, 1998.

[3] F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in LNAI, pages 396–408. Springer-Verlag, 2001.

[5] T. Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of CoopIS'98*, pages 280–291, 1998.

[7] S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5), 2000.

[8] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[9] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[10] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of IJCAI-01*, 2001.

[11] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR-01*, 2001.

[12] I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of TABLEAUX-98*, volume 1397 of *LNAI*, pages 307–312. Springer-Verlag, 1998.

[13] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647, 1998.

[14] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of IJCAI-01*. Morgan Kaufmann, 2001.

[15] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer-Verlag, 1999.

[16] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS*, Frontiers in Artificial Intelligence and Applications. IOS-press, 1998.

[17] D. L. McGuinness. Ontologies for electronic commerce. In *Proc. of the AAAI '99 Artificial Intelligence for Electronic Commerce Workshop*, 1999.

[18] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera ontology environment. In *Proc. of AAAI 2000*, 2000.

[19] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, March/April 2001.

[20] P. F. Patel-Schneider. DLP system description. In *Proc. of DL'98*, pages 87–89. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.

[21] The Gene Ontology Consortium. Gene ontolgy: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

SEAL — Tying Up Information Integration and Web Site Management by Ontologies

[2]Alexander Maedche, [1,3]Steffen Staab, [1,2,3]Rudi Studer, [1]York Sure, [1]Raphael Volz
maedche@fzi.de
{staab,studer,sure,volz}@aifb.uni-karlsruhe.de

[1]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS/
[2]FZI Research Center for Information Technologies,
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
http://www.fzi.de/wim/
[3]Ontoprise GmbH, Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany
http://www.ontoprise.de

Abstract

Community web sites exhibit two dominating properties: They often need to integrate many different information sources and they require an adequate web site management system. SEAL (SEmantic portAL) is a conceptual model that exploits ontologies for fulfilling the requirements set forth by these two properties at once. The ontology provides a high level of sophistication for web information integration as well as for web site management. We describe the SEAL conceptual architecture as well as its current implementation in KAON.

1 Introduction

The recent decade has seen a tremendous progress in managing semantically heterogeneous data sources. Core to the semantic reconcilation between the different sources is a rich conceptual model that the various stakeholders agree on, an *ontology* [10]. The conceptual architecture developed for this purpose now generally consists of a three layer architecture comprising (cf. [24])

1. heterogeneous **data sources** (e.g., databases, XML, but also data found in HTML tables),
2. **wrappers** that lift these data sources onto a common data model (e.g. OEM [18] or RDF [16]),
3. integration modules (**mediators** in the dynamic case) that reconcile the varying semantics of the different data sources.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Thus, the complexity of the integration/mediation task could be greatly reduced.

Similarly, in recent years the information system community has successfully strived to reduce the effort for managing complex web sites [1, 5, 4, 12, 11, 17]). Previously ill-structured web site management has been structured with process models, redundancy of data has been avoided by generating it from database systems and web site generation (including management, authoring, business logic and design) has profited from recent, also commercially viable, successes [1]. Again we may recognize that core to these different web site management approaches is a rich conceptual model that allows for accurate and flexible access to data. Similarly, in the hypertext community conceptual models have been explored that im- or explicitly exploit ontologies as underlying structures for hypertext generation and use [6, 19, 13].

Semantic Portal. The topic of this paper is SEAL (SEmantic PortAL), a framework for managing community web sites and web portals on an ontology basis. The ontology supports queries to multiple sources (a task also supported by semi-structured data models [11]), but beyond that it also includes the intensive use of the schema information itself allowing for automatic generation of navigational views[1] and mixed ontology and content-based presentation. The core idea of SEAL is that Semantic Portals for a community of users that contribute *and* consume information [20] require web site management *and* web information integration. In order to reduce engineering and maintenance efforts SEAL uses an ontology for semantic integration of existing data sources as well as for web site management and presentation to the outside world. SEAL exploits the ontology to offer mechanisms for acquiring, structuring and sharing information between human and/or machine agents. Thus, SEAL combines the advantages of the two worlds briefly sketched above.

The SEAL conceptual architecture (cf. Figure 1; details to be explained in subsequent sections) depicts the general scheme. Approaches for web site management emphasize on the upper part of the figure and approaches for web information integration focus on the lower part while SEAL combines both with an ontology as the knot in the middle.

History. The origins of SEAL lie in Ontobroker [8], which was conceived for semantic search of knowledge on the Web and also used for sharing knowledge on the Web [3], also taking advantage of the mediation capabilities of ontologies [10]. It then developed into an overarching framework for search and presentation offering access at a portal site [20]. This concept was then transferred to further applications [2],[22] and constitutes the technological basis for the portal of our institution[2] (among others)[3]. It now combines the roles of information integration in order to provide data for the Semantic Web and for a Peer-to-Peer network with presentation to human Web surfers.

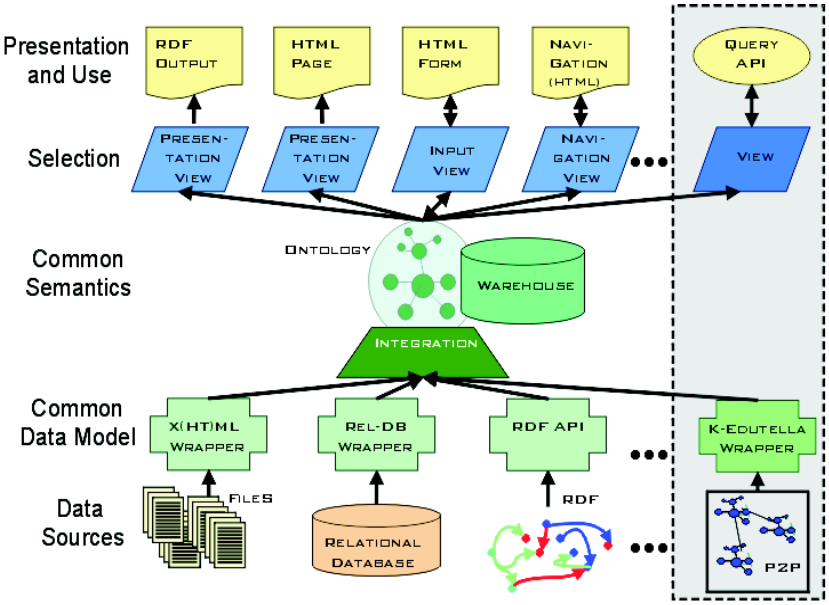


Figure 1: SEAL conceptual architecture

[1] Examples are navigation hierarchies that appear as `has-part`-trees or `has-subtopic` trees in the ontology.

[2] http://www.aifb.uni-karlsruhe.de

[3] Also the web portal of of the EU-funded thematic network ”OntoWeb” (http://www.ontoweb.org) and the KA2 community web portal (http://ka2portal.aifb.uni-karlsruhe.de)

2 Web Information Integration

One of the core challenges when building a data-intensive web site is the integration of heterogeneous information on the WWW. The recent decade has seen a tremendous progress in managing semantically heterogeneous data sources [24, 11]. The general approach we pursue is to "lift" all the different input sources onto a common data model, in our case RDF. Additionally, an ontology acts as a semantic model for the heterogeneous input sources. As mentioned earlier and visualized in our conceptual architecture in Figure 1, we consider different kinds of **data sources** of the Web as input: First of all, to a large part the Web consists of static HTML pages, often semi-structured, including tables, lists, etc. We have developed an ontology-based **HTML wrapper** that is based on a semi-supervised annotation approach. Thus, based on a set of predefined manually annotated HTML pages, the structure of new HTML pages is analyzed, compared with the annotated HTML pages and relevant information is extracted from the HTML page. The HTML wrapper is currently extended to also deal with heterogeneous XML files. Second, we use an automatic XML wrapping approach that has been introduced in [9]. The idea behind this wrapping approach is that these XML documents refer to an DTD that has been generated from the ontology. Therefore we automatically generate a mapping from XML to our data model so that integration comes for free. Third, data-intensive applications typically rely on relational databases. A relational database wrapping approach [21] maps relational database schemas onto ontologies that form the semantic basis for the RDF statements that are automatically created from the relational database. Fourth, in an ideal case content providers have been registered and agreed to describe and enrich their content with RDF-based metadata according to a shared ontology. In this case, we may easily integrate the content automatically by executing an **integration** process. If content providers have not been registered, but provide RDF-based metadata on their Web pages, we use ontology-focused metadata discovery and crawling techniques to detect relevant RDF statements.

Our generic Web information integration architecture is extensible, as shown in Figure 1. In particular, we are currently working on connecting and integrating data sources available via enhanced **Peer-2-Peer (P2P)** networks. P2P applications for searching and exchanging information over the Web have become increasingly popular. The **Edutella**[4] approach builds upon the RDF metadata standard aiming to provide an RDF-based metadata infrastructure for P2P applications, building on the recently announced JXTA framework.

It is important to mention that in our current architecture and implementation we mainly apply **static** information integration building on a warehousing approach. Means for **dynamic** information integration are currently approached for Peer-2-Peer networks and within our relational database wrapper.

3 Web Site Management

One difficulty of community portals lies in integrating heterogeneous data sources. Each source may be hosted by different community members or external parties and fulfills different requirements. Therefore typically all sources vary in structure and design. Community portals like (in our case) the web site of our own institute require coherence in hosted information on different levels. While the information integration aspect (see previous section) satisfies the need for a coherent structure that is provided by the ontology we will now introduce various facilities for construction and maintenance of websites to offer coherent style and design. Each facility is illustrated by our conceptual architecture (cf. Figure 1).

Presentation view. Based on the integrated data in the warehouse we define user-dependent presentation views. First, as a contribution to the Semantic Web, our architecture is dedicated to satisfy the needs of software agents and produces machine understandable RDF. Second, we render HTML pages for human agents. Typically *queries for content* of the warehouse define presentation views by selecting content, but also *queries for schema* might be used, e.g. to label table headers.

[4] http://edutella.jxta.org

Input view. To maintain a portal and keep it alive its content needs to be updated frequently not only by information integration of different sources but also by additional inputs from human experts. The input view is defined by *queries to the schema*, i.e. queries to the ontology itself. Similar to [14] we support the knowledge acquisition task by generating forms out of the ontology. The forms capture data according to the ontology in a consistent way which are stored afterwards in the warehouse (cf. Figure 3).

Navigation view. To navigate and browse the warehouse we automatically generate navigational structures by using *combined queries for schema and content*. First, we offer different user views on the ontology by using different types of hierarchies (e.g. *is-a*, *part-of*) for the creation of top level navigational structures. Second, for each shown part of the ontology the corresponding content in the warehouse is presented. Therefore especially users that are unfamiliar with the portal are supported to explore the schema and corresponding content.

(General) View. In the future we plan to explore techniques of handling updates on these views.

4 Technical Architecture

The technical architecture of SEAL is derived from the architecture of KAON, the Karlsruhe Semantic Web and Ontology Infrastructure[5], whose components provide the required functionalities described in the previous sections. The architecture of KAON is depicted in Figure 2. KAON components can roughly be grouped into three layers.

The data and remote services layer represents optional external services, which can be used in the upper layers, e.g. reasoning services for inferencing and querying, or connectors to the Edutella Peer-To-Peer network, and alternative storage mechanisms for the data in the previously mentioned warehouse.

The middleware layer provides a high-level API for manipulating ontologies and associated data and hides the actual manner of storage and communication from all clients. Thus clients cannot distinguish between working on the local file system (provided by the RDF API) or working on a multi-user aware server which stores data in a relational database. The middleware also provides interfaces to QEL, the query language used within the Edutella network, which is not only used to communicate queries within the peer-to-peer network but also used to query the warehouse.

The application and services layer groups applications that use services from the underlying layers. Currently these are one hand, stand-alone desktop applications built using the Ont-O-Mat application framework or portals built using the KAON portal maker, which provides the features discussed in section 3. Ont-O-Mat applications are built as plug-ins that are hosted by the Ont-O-Mat application framework. This approach guarantees maximum application interoperability within Ont-O-Mat.

Finally, core to KAON is the domain ontology itself, which is represented in RDF Schema[23] - the data model at hand for representing ontologies in the Semantic Web. It provides basic class and property hierarchies and relations between classes and objects. Historically SEAL leverages the mapping of RDF Schema model to F-Logic[15] introduced in [7] to provide views (in form of logical axioms) and a query mechanism. This allows us to rely on the reasoning services offered by OntoBroker [8] or SiLRi [7] .

5 Creating a SEAL-based Web Site

The creation of a SEAL-based web site is a multi-step process. The genesis starts with the creation of the ontology, which provides a conceptualization of the domain and is later used as the content model of the portal.

Step 1 – Ontology design: Here, several tools come in handy, within KAON Ont-O-Mat SOEP provides an editor with strong abilities regarding the evolution of the ontology. OntoEdit is a commercial tool that additionally allows to provide F-Logic axioms to refine the ontology.

[5] http://kaon.semanticweb.org

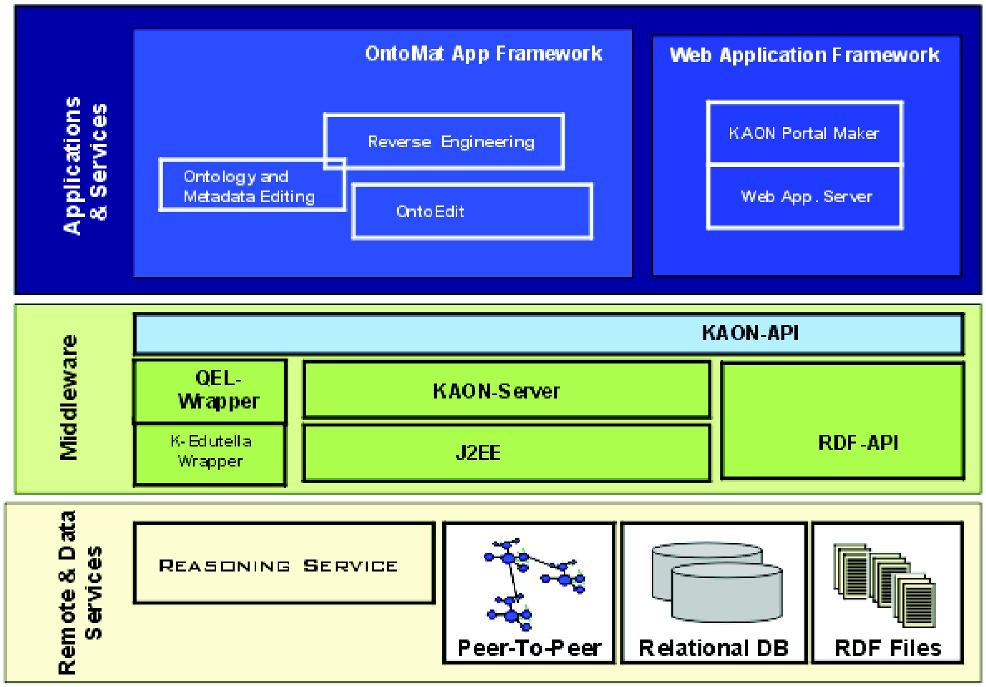


Figure 2: KAON architecture

Step 2 – Integrating Information: The next step towards the final web site is providing data. Here, we take a warehousing approach to amalgamate information coming from heterogeneous data sources.

- *RDF metadata* User-supplied HTML and PDF documents have to be annotated with metadata based on the content ontology in order to be part of the SEAL portal. These documents can be located anywhere on the web and are made part of the portal using KAON Syndicator, a component that gathers the meta data contained in resources located on the web.

- *Database Content* Today most large-scale web applications present content derived from databases. KAON REVERSE is an application that provides visual means to map the logical schema of relational databases to the integrated conceptual model provided by the ontology [21]. The user-supplied mappings are then used to transform the database content to ontology-based RDF.

- *Peer-To-Peer* Also connectors to the **Edutella** peer-to-peer network[6], that provides an RDF-based metadata infrastructure for peer-to-peer applications, are currently constructed within KAON. SEAL portals can then be used to provide a web accessible interface to Edutella based Peer-To-Peer networks

Step 3 – Site design: We derive the previously mentioned navigation model and personalization model from the ontology. Currently no extensive tool support for these tasks exist. Both models are derived from the ontology using F-Logic queries that are provided by the site administrator.

Navigation model Beside the hierarchical, tree-based hyperlink structure which corresponds to the hierarchical decomposition of the domain, the navigation module enables complex graph-based semantic hyperlinking, based on ontological relations between concepts (nodes) in the domain. The conceptual approach to hyperlinking is based on the assumption that semantically relevant hyperlinks from a web page correspond to conceptual relations, such as `memberOf` or `hasPart`, or to attributes, like `hasName`. Thus, instances in the knowledge base

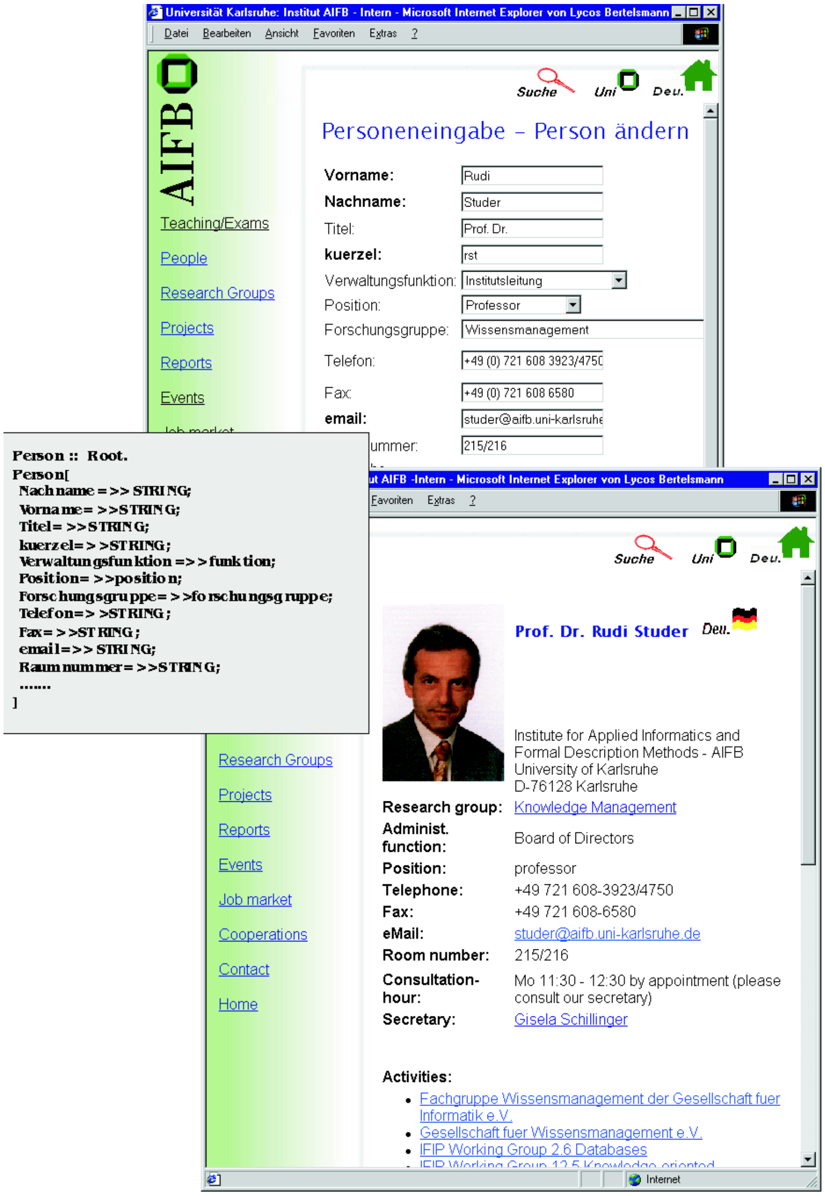
[6] http://edutella.jxta.org

may be presented by automatically generating links to all related instances. For example, on personal web pages there are, among others, hyperlinks to web pages that describe the corresponding research groups, secretary and professional activities (*cf.* Figure 3).

Step 4 – Web design: The derived models constructed in step 3 serve as input to the KAON Portal Maker, which renders the information in HTML. The implementation of KAON portal Maker adheres strictly to a model-view-controller design pattern. The ontology and the derived models are encapsulated by an abstract data model and the presentation of the information is created using template technologies like JSP, ASP or XSLT.

Default controllers are provided for standard application logic like updating data and generating links to other presentation objects. The reader may note that the default controllers can be replaced by custom-made controllers provided by the site administration.

KAON Portals also provides default templates that provide the most often used representations for information objects (like list-entries, forms for web-based data provision etc.) For instance, the AIFB portal includes an input template (*cf.* Figure 3, upper part) generated from the concept definition of person (*cf.* Figure 3, middle left) and a sheet like representation to produce the corresponding person web page (*cf.* Figure 3, lower part). These default templates can easily be customized for special purposes.



Personeneingabe – Person ändern

Vorname:	Rudi
Nachname:	Studer
Titel:	Prof. Dr.
kuerzel:	rst
Verwaltungsfunktion:	Institutsleitung
Position:	Professor
Forschungsgruppe:	Wissensmanagement
Telefon:	+49 (0) 721 608 3923/4750
Fax:	+49 (0) 721 608 6580
email:	studer@aifb.uni-karlsruhe
ummer:	215/216

```
Person :: Root.
Person[
 Nachname=>> STRING;
 Vorname=>>STRING;
 Titel=>>STRING;
 kuerzel=>>STRING;
 Verwaltungsfunktion=>>funktion;
 Position=>>position;
 Forschungsgruppe=>>forschungsgruppe;
 Telefon=>>STRING;
 Fax=>>STRING;
 email=>>STRING;
 Raumnummer=>>STRING;
 .......
]
```

Prof. Dr. Rudi Studer

Institute for Applied Informatics and
Formal Description Methods - AIFB
University of Karlsruhe
D-76128 Karlsruhe

Research group:	Knowledge Management
Administ. function:	Board of Directors
Position:	professor
Telephone:	+49 721 608-3923/4750
Fax:	+49 721 608-6580
eMail:	studer@aifb.uni-karlsruhe.de
Room number:	215/216
Consultation-hour:	Mo 11:30 - 12:30 by appointment (please consult our secretary)
Secretary:	Gisela Schillinger

Activities:

- Fachgruppe Wissensmanagement der Gesellschaft fuer Informatik e.V.
- Gesellschaft fuer Wissensmanagement e.V.
- IFIP Working Group 2.6 Databases
- IFIP Working Group 12.5 Knowledge-oriented

Figure 3: Templates generated from the web-site models

6 Discussion

The SEAL approach offers a comprehensive conceptual framework for Web information integration and Web site management. A crucial feature of SEAL is the use of an ontology as a semantic backbone for the framework. Thus, all functions for information integration as well as for information selection and presentation are glued together by a semantic conceptual model, i.e. a domain ontology. Such an ontology offers a rich structuring of concepts and relations that is supplemented by axioms for specifying additional semantic aspects. The ontological foundation of SEAL is the main distinguishing feature when comparing SEAL with approaches from the information systems community.

The STRUDEL system [11] is an approach for implementing data-intensive Web sites. STRUDEL provides a clear separation of three tasks that are important for building up a data-intensive Web site: (i) accessing and integrating the data available in the Web site, (ii) building up the structure and content of the site, and (iii) generating the HTML representation of the site pages. Basically, STRUDEL relies on a mediator architecture where the semi-structured OEM data model is used at the mediation level to provide a homogeneous view on the underlying data sources. STRUDEL then uses so-called 'site definition queries' to specify the structure and content of a Web site. When compared to our SEAL approach STRUDEL lacks the semantic level that is defined by the ontology. Furthermore, within SEAL the ontology offers a rich conceptual view on the underlying sources that is shared by the Web site users and that is made accessible at the user interface for e.g. browsing and querying.

The Web Modeling Language WebML [4] provides means for specifying complex Web sites on a conceptual level. Aspects that are covered by WebML are a.o. descriptions of the site content, the layout and navigation structure as well as personalization features.Thus, WebML addresses functionalities that are offered by the presentation and selection layer of the SEAL conceptual architecture. Whereas WebML provides more sophisticated means for e.g. specifying the navigation structure, SEAL offers more powerful means for accessing the content of the Web site, e.g. by semantic querying.

In addition to ongoing work to integrate Peer-to-Peer functions for accessing information on the Web, two topics are currently under investigation: first, the view concept that is implemented by the KAON framework does not support updates in general. Currently, only the simplistic input views provide means for updating the warehouse. Clearly, Web site users do expect to be able to update the site content. A second topic that needs further improvement is the handling of ontologies. Just offering a single, centralized ontology for all Web site users does not meet the requirements for heterogeneous user groups. Therefore, methods and tools are under development that support the handling and aligning of multiple ontologies.

The SEAL framework as well as the KAON infrastructure can be seen as steps for realizing the idea of the Semantic Web. Obviously, further steps are needed to transfer these approaches into practice.

Acknowledgements. We thank our colleagues and students at the Institute AIFB, University of Karlsruhe, at FZI Research Center for Information Technologies at the University of Karlsruhe and at Ontoprise GmbH for many fruitful interactions. Especially, we would like to thank our colleagues Siegfried Handschuh and Nenad Stojanovic for their contributions to the SEAL framework. Research reported in this paper has been partially financed by EU in the IST projects On-To-Knowledge (IST-1999-10132) and Ontologging (IST-2000-28293).

References

[1] C. R. Anderson, A. Y. Levy, and D. S. Weld. Declarative web site management with tiramisu. In *ACM SIGMOD Workshop on the Web and Databases - WebDB99*, pages 19–24, 1999.

[2] J. Angele, H.-P. Schnurr, S. Staab, and R. Studer. The times they are a-changin' — the corporate history analyzer. In D. Mahling and U. Reimer, editors, *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management. Basel, Switzerland, October 30-21, 2000*, 2000. http://www.research.swisslife.ch/pakm2000/.

[3] V. R. Benjamins, D. Fensel, S. Decker, and A. G. Perez. $(KA)^2$: Building ontologies for the internet. *International Journal of Human-Computer Studies (IJHCS)*, 51(1):687–712, 1999.

[4] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. In *WWW9 Conference, Amsterdam, May 2000*, 2000.

[5] S. Ceri, P. Fraternali, and S. Paraboschi. Data-driven one-to-one web site generation for data-intensive applications. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 615–626, 1999.

[6] M. Crampes and S. Ranwez. Ontology-supported and ontology-driven conceptual navigation on the world wide web. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia, May 30 - June 3, 2000, San Antonio, TX, USA*, pages 191–199. ACM Press, 2000.

[7] S. Decker, D. Brickley, J. Saarela, and J. Angele. A query and inference service for RDF. In *QL98 - Query Languages Workshop*, December 1998.

[8] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–269. Kluwer Academic Publisher, 1999.

[9] M. Erdmann and R. Studer. How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, 36(3):317–235, 2001.

[10] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, R. Studer, and A. Witt. Lessons learned from applying AI to the web. *International Journal of Cooperative Information Systems*, 9(4):361–282, 2000.

[11] M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. Declarative specification of web sites with Strudel. *VLDB Journal*, 9(1):38–55, 2000.

[12] P. Fraternali and P. Paolini. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In *EDBT 1998*, pages 421–435, 1998.

[13] C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual open hypermedia = the semantic web? In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001, Hongkong, China, May 1, 2001*. CEUR Workshop Proceedings, 2001. http: //CEUR-WS.org/Vol-40/.

[14] E. Grosso, H. Eriksson, R. W. Fergerson, S. W. Tu, and M. M. Musen. Knowledge modeling at the millennium: the design and evolution of PROTEGE-2000. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW-99)*, Banff, Canada, October 1999.

[15] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[16] O. Lassila and R. Swick. Resource description framework (RDF). model and syntax specification. Technical report, W3C, 1999. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax.

[17] G. Mecca, P. Merialdo, P. Atzeni, and V. Crescenzi. The (short) Araneus guide to web-site development. In *Second Intern. Workshop on the Web and Databases (WebDB'99)* , May 1999.

[18] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of the IEEE International Conference on Data Engineering, Taipei, Taiwan, March 1995*, pages 251–260, 1995.

[19] G. Rossi, A. Garrido, and D. Schwabe. Navigating between objects. lessons from an object-oriented framework perspective. *ACM Computing Surveys*, 32(30), 2000.

[20] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic community web portals. In *WWW9 / Computer Networks (Special Issue: WWW9 - Proceedings of the 9th International World Wide Web Conference, Amsterdam, The Netherlands, May, 15-19, 2000)*, volume 33, pages 473–491. Elsevier, 2000.

[21] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In *Proceedings of the ACM Symposium on Applied Computing SAC-02, Madrid, 2002*, 2002.

[22] Y. Sure, A. Maedche, and S. Staab. Leveraging corporate skill knowledge - From ProPer to OntoProper. In D. Mahling and U. Reimer, editors, *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management. Basel, Switzerland, October 30-21, 2000*, 2000. http://www.research.swisslife.ch/pakm2000/.

[23] W3C. RDF Schema Specification. http://www.w3.org/TR/PR-rdf-schema/, 1999.

[24] G. Wiederhold and M. Genesereth. The conceptual basis for mediation services. *IEEE Expert*, 12(5):38–47, Sep.-Oct. 1997.

Architecture and Implementation of an XQuery-based Information Integration Platform

Yannis Papakonstantinou
Computer Science and Engineering
University of California, San Diego
yannis@cs.ucsd.edu

Vasilis Vassalos
Information Systems Group, Stern School of Business
New York University
vassalos@stern.nyu.edu

Abstract

An increasing number of business users and software applications need to process information that is accessible via multiple diverse information systems, such as database systems, file systems, legacy applications or web services. We describe the Enosys XML Integration Platform (EXIP), a commercial XQuery-based data integration software platform that provides a queryable integrated view of such information. We describe the platform architecture and describe what the main principles and challenges are for the query engine. In particular, we discuss the query engine architecture and the underlying semistructured algebra, which is tuned for enabling query plan optimizations.

1 Introduction

A large variety of Web-based applications demand access and integration of up-to-date information from multiple distributed and heterogeneous information systems. The relevant data are often owned by different organizations, and the information sources represent, maintain, and export the information using a variety of formats, interfaces and semantics. The ability to appropriately assemble information represented in different data models and available on sources with varying capabilities is a necessary first step to realize the Semantic Web [3], where diverse information is given coherent and well-defined meaning. The Enosys XML Integration Platform (EXIP) addresses the significant challenges present in information integration:

- Data of different sources change at different rates, making the data warehousing approach to integration hard to develop and maintain. In addition, Web sources may not provide their full data in advance. The platform we describe resolves this challenge by being based on the on-demand mediator approach [47, 8, 32, 6, 44, 28]: information is collected dynamically from the sources, in response to application requests.

- The Mediator, which is the query processing core of the EXIP platform, has to decompose application requests into an efficient series of requests targeted to the sources. These requests have to be compatible with the query capabilities of the underlying sources. For example, if the underlying source is an XML

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

file, data may only be retrieved from the file sequentially. All data processing operations on the data of the file will have to be performed by the Mediator. On the other hand, if the underlying source is a powerful SQL DBMS, the Mediator can send to it SQL queries that delegate most of the data processing operations to the SQL query processor, hence providing multiple efficiencies: The amount of data retrieved from the database is potentially much smaller, and the source's processing power and optimization ability is harnessed to answer the integrated query.

  In order to address this challenge, the Mediator rewrites the query plan in order to push the most efficient supported query to the underlying sources. In the algebra-based EXIP query processor this is done by having the rewriter/optimizer transform the algebraic expressions to appropriate sub-expressions and "delegate" them to the sources. Wrappers are responsible for translating these subexpressions into SQL or other queries/commands acceptable to the information source [14, 26].

- Integrated views are the key abstraction offered by the EXIP platform, and they often need to be defined over a variety of sources, with different capabilities and access methods, and for use by different applications. The platform makes source metadata available for view definition in an automatic way and simplifies the view definition process via graphical tools.

- Information assets available for integration reside in diverse systems, have different structure, and are usually in heterogeneous formats. The Mediator enables and facilitates the resolution of the heterogeneities by using XQuery to perform complex structural transformations. Furthermore, extensibility of the query engine is required in order to allow easy interface with function libraries built in other programming languages, such as Java, that perform special-purpose, domain-specific transformations.

- The heterogeneity and Web-orientation of modern applications that make use of the EXIP platform for integrated access to diverse information again require a lot of flexibility from EXIP. Different applications use different XML views and queries, which need to structure the XML data as close as possible to the application needs. We have found in particular that, for presentation-oriented Web applications, XML views and queries that structure the data in a way that is "isomorphic" to the HTML structure of the produced Web pages lead to huge time savings in building the applications. Providing the flexibility to produce structures that fit the application requirements again requires a powerful language for selection, join, and transformation.

The EXIP platform uses the XPath/XQuery data model [16], augmented with Skolem functions, which were first proposed in [32] in the context of the OEM model [33]. The platform enables applications and users to access and integrate information using XQuery, the W3C draft proposal for a high-level, declarative XML query language [7]. XQuery statements can reference integrated views, also defined in XQuery with small extensions. Queries and views are translated into the semistructured algrebra used by EXIP, henceforth refered to as *XAlgebra*, are combined into a single algebra expression/plan, and are rewritten/optimized to effect query composition with the views and decomposition into plans appropriate for delegation to the underlying sources.

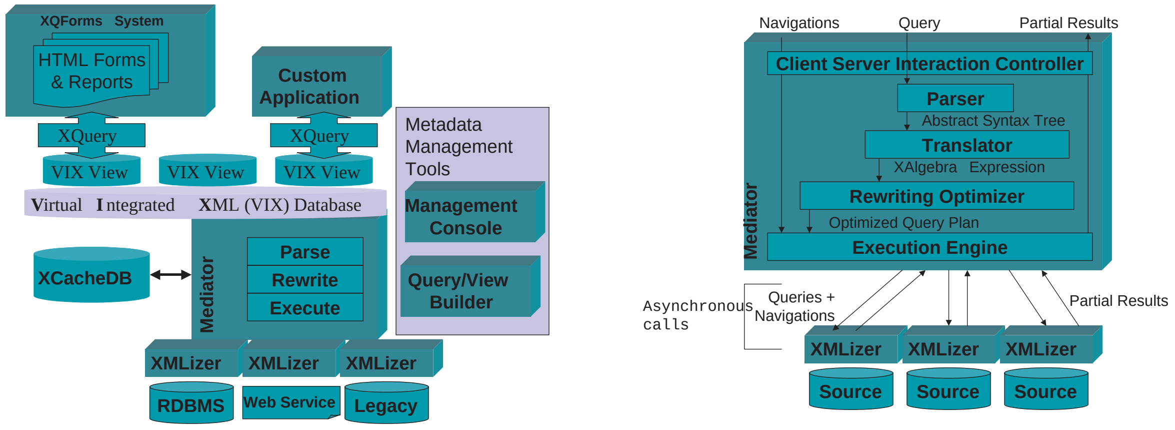
Processing XML query statements in a dynamic mediator context in a way that addresses the challenges mentioned above creates new query processing challenges, as we discuss in Section 3. In addition, the EXIP platform surrounds the Mediator with a series of components that raise the usability of the platform and increase its efficiency.

The EXIP platform is currently in use for a variety of integration projects in large corporations and is being evaluated by two leading software companies for incorporation in their data processing and application server products.

Roadmap Section 2 presents the architecture and components of the EXIP platform. Section 3 presents the internal architecture of the Mediator query processor and introduces the reader to XQuery and the semistructured algebra used in the Mediator's query processing. Section 4 discusses related work.

2 High-Level Platform Architecture and Components

The Enosys XML Integration Platform is based on the wrapper-mediator architecture, as shown in Figure 1(a). Each wrapper, called XMLizer, accesses an information source, such as a relational DBMS or a web service, and exports a Virtual XML view of it. In particular, a wrapper exports an XML Schema describing the content of the source as XML, and it allows querying of that content by the Mediator. It also obtains and exports metadata describing the capabilities of the sources, such as the existence of special access structures or adornments [20] for web services. The Mediator exports the Virtual Integrated XML (VIX) database, which provides access to all the individual views exported by the wrappers. Virtual integrated XML views and queries can be built on top of the VIX database. The views organize information from the distributed sources into XML objects that conform to an application's needs. For example, to a marketplace application the integrated XML view can provide front-end access to an integrated catalog, where the heterogeneities between the suppliers' products are resolved, and the products are integrated and classified according to the needs of the marketplace. Each product object contains catalog data along with attributes from the pricing, delivery, service and other databases. The views provide distribution transparency, *i.e.*, the originating sources and methods of access are transparent to the application. For example, it is transparent to the application that the product specifications in the catalog come from a product database, while the pricing and service information may be coming from a Customer Relationship Management (CRM) system, which often provides customized pricing and service options for each customer.



(a) The High-Level Architecture of the Enosys Platform

(b) The Query Processor Architecture

Figure 1: EXIP architecture

The virtual database and views enable the front-end applications, which may be the XQForms system [38] for declarative generation of query forms and reports or custom applications, to seamlessly access distributed heterogeneous information sources as if they were a single XML database. In particular, the application can issue an XML query against either the XML database or the views. The query typically selects information from the views and structures it in ways that are convenient for the application. For example, a HTML application will create queries that structure the XML results in a way that easily translate into the target HTML pages. In the catalog example, if the resulting HTML page presents the data grouped by product family then the XML result will greatly facilitate the construction of the HTML page if the results are organized hierarchically by product family.

An XML view can be cached into the XCacheDB, which is an XQuery database implemented on top of a relational back-end, as in Niagara [42], employing proprietary storage and query processing and translation algorithms. Typically, the data of slow and relatively static sources are collected, integrated, and cached in advance, while the information originating at fast dynamic sources is collected by accessing the sources dynamically. It is transparent to the application which pieces of the view originate from the underlying sources and which ones originate from XCacheDB.

The Mediator is accessible by applications through a query language API and a DOM-based (Document Object Model) API. Finally, the platform includes management tools that enable the user to graphically create and manage query forms, view definitions, queries, source connections and other metadata.

Example 1: Assume we want to find information about available houses, to guide our home-buying decision-making. Two important considerations are size and price, and we also want to know the quality of the schools in the home's neighborhood. Also assume that

- information on houses and their neighborhoods is accesible via a realtor's database system
- information about school performance is available from an external web service

The following XQuery retrieves the appropriate integrated information:

```
FOR $h IN database("homesDB")/*/homes/home
WHERE
    $h/area > 1,500 AND
    $h/price <500,000
RETURN <home_with_schools>
   {$h},
   {FOR  $s IN database("schoolsDB")/*/schools/school
    WHERE $s/zip = $h/zip
    RETURN
     {$s},
     {ws:schoolreview($s/name)}
   }
</home_with_schools>
```

The XQuery produces `home_with_schools` elements that group houses with the required size and within the right price range with schools in the same zip code and with their reviews. The web service `schoolreview`, offered by an external entity, *e.g.*, the Department of Education, is invoked as an external function, via a simple extension to the function naming mechanism of XQuery. The physical details of the Web service are given in its WSDL description. The school reviews, if the data were available, would also be a good candidate for storage in XCacheDB, for improved performance. A query such as this can be built graphically with EXIP's metadata-aware query and view builder.

For more detail on the components of EXIP, please refer to [37]. In the next section, we discuss briefly how such a query is processed by the EXIP Mediator.

3 Mediator Query Processing Architecture

As we described earlier, the Mediator inputs an XQuery, one or more optional view definitions (also in XQuery), a description of the sources and optional XML Schemas of the sources, and a description of available user-defined functions. It returns the XML query result. The Mediator architecture, shown in Figure 1(b), is similar to the architecture of a traditional DBMS query engine [20], with a few significant differences that are discussed in this section. A query is first parsed into an Abstract Syntax Tree. During parsing, references to data sources and external functions are resolved, using the metadata information available to the Mediator. References to views are also resolved, and view definitions are parsed. Moreover, the parser performs type checks and infers the type of the final as well as of intermediate results [13].

The produced AST is then translated by the *translator* into an XAlgebra expression. The principles of XAlgebra are described in Section 3.2. Consequently, the *rewriting optimizer* applies a series of algebraic and cost-based rewritings on the algebra expression in order to optimize it. We use a rule-based heuristic rewriter that performs optimizations such as

- constant folding
- pushing selections down the algebra expression
- optimizing the join order based on the characterisitcs of the sources (such as the existence of fast access structures)
- unfolding nested XAlgebra expressions
- optimizations targeted for navigation-based evaluation, which is described further in Section 3.1
- choosing the right kind among semantically equivalent operators, *e.g.*, sort-merge join operator or nested loops join operator.

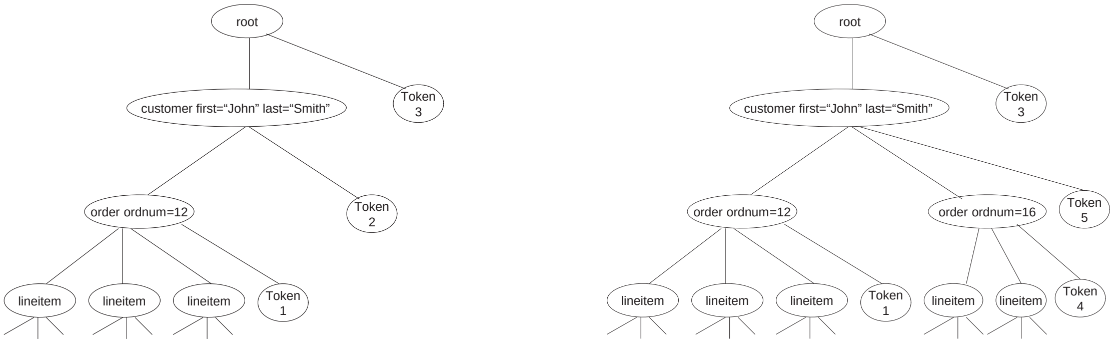
The rewriter performs *capability-sensitive decomposition*, *i.e.*, it decomposes the logical plan into the *maximal* fragments that can be executed by the data sources and the Mediator and respect the capabilities of the data sources. That is, the decomposition pushes as much processing as possible to the data sources, based on their query capabilities. In the case of relational databases, the rewriter also uses heuristics that allow it to take into account the abilities of the query optimizer of the underlying relational database, so that the decomposition produces fragments that are efficiently optimizable by the underlying system (and hence, not necessarily maximal.)

The plan is finally run by the *execution engine*. Conceptually, the execution engine receives the query plan, sends requests/queries to the wrappers, and performs necessary local processing, such as joins across data sources and XML tagging and composition of the result. In practice, the query result object is not fully materialized immediately. Instead, query evaluation is driven by the client navigations, as described below.

3.1 Navigation-Driven Evaluation

Since Web-based applications are very often accessed by large numbers of users, and often generate correspondingly large numbers of requests to the Mediator, good use of available resources, and in particular main memory and bandwidth, is of high importance. The EXIP Mediator allows "just-in-time" generation of the necessary (parts of the) query result, by integrating querying and result object navigation. In particular, the Mediator, following an initial negotiation with the client, may only send parts of a query result to the client. In place of the missing parts of the results, appropriate tokens are included that the client may send back to the server if it needs one of the missing parts. The tokens contain information needed for tne Mediator to produce the missing parts. The effects of the on-demand, navigation-driven execution model are lower memory footprint and reduced data exchanges from the data sources to the Mediator [31].

A partial result has multiple lists of elements that are *incomplete*, as in Figure 2(a), from where navigation can continue because of the information encoded in the tokens. For example, the Mediator may have exported the partial result of Figure 2(a). If the client navigates to the second child of customer John Smith, the `Token 2` is passed to the Mediator and the Mediator produces another partial result, which leads to a tree such as the one of Figure 2(b).



(a) A partial result

(b) The partial result after navigation to the right of order ”12”

Figure 2: Navigation-driven evaluation

Different clients are exposed differently to result fragmentation and navigation-based evaluation, depending on the interaction model used between the client and the Mediator server. Web-based query applications, which use the Mediator as a servlet, [1] benefit from being aware of the existence of tokens and can be made responsible for returning the appropriate tokens to the Mediator via a POST request. This approach fits well with the navigational nature of such applications, *e.g.*, with the existence of “Next” or “Drill Down” hyperlinks that are a common feature of these applications. On the other hand, a general purpose application can be shielded from the existence of tokens: a thin client library is in charge of passing the appropriate tokens to the Mediator server. The client navigates in the result fragment, unaware that parts of the result are not yet in client memory. If the client happens to navigate into a missing part, the library will send the relevant token to the server and fetch the required fragment.

A key challenge in optimizing navigation-driven query evaluation is the choice of the size and shape of fragments produced. At the one extreme, one may choose each node of the data model to be a fragment and encode the relevant token in the node itself. This approach, described in [31], is elegant but it is very inefficient in the number of round-trips that will be needed between the client and the server. Moreover, it penalizes the server with unacceptable overhead in creating tokens.

Instead, the Mediator employs a complex algorithm for *client-server interaction control (CLSIC)*, as shown in Figure 1(b). The CLSIC algorithm is reponsible for choosing the size (and shape) of the fragment that will be returned to the client. The algorithm takes as input configuration parameters provided by the client.

The Mediator execution engine supports CLSIC through a pipelined, iterator-based execution model, which additionally allows the computation state of operators to be exported and imported. Each operator can respond to a call for the “next” tuple (as we discuss in the next section, XAlgebra is tuple-oriented.) Moreover, each operator enables the production of tokens by being able to produce information about its state on command. This state information is encoded by CLSIC in tokens. Each operator is able to reproduce a prior state and continue its computation from that point on. Upon receipt of a token, CLSIC produces and imposes the appropriate state on each operator.

Note that, depending on the capabilities of the underlying sources for on-demand evaluation, the Mediator (and the corresponding XMLizer) sets up and invokes appropriate access methods of these sources, such as SAX calls or SQL cursors. For example, assume that a query produces `customer` elements, from a `customer` table of an underlying relational source. When the client issues the query, the corresponding XMLizer establishes a

[1] Such applications are often implemented using technologies such as Java Server Pages.

cursor and passes this cursor to the Mediator, together with the first fragment of the result, *e.g.*, the first 100 tuples. The Mediator encodes the cursor state information in the token. When the client asks for the 101st `customer` element, the Mediator passes the cursor information back to the XMLizer and asks for the “next” tuple. Note also that the optimization target of the rewriting optimizer can be set to either speed up response time to navigation commands or to speed up production of the full result.

3.2 Principles of XAlgebra

XAlgrebra is a fine-grained algebra used to represent and manipulate queries in the Mediator. XAlgebra is functional, and algebra expressions are fully composable. XAlgebra is also tuple-oriented, meaning that the result of most operators is a set of tuples, and draws on the relational and nested relational algebras [20]. Tuple orientation allows the construction of an iterator-based execution model, which extends the iterator model of relational databases and enables navigation-driven partial evaluation, as described in the previous section. Moreover, it enables the inclusion in the algebra of powerful operators for join and grouping operations, which are much easier specified in terms of tuples and tuple partitions [2, 11]. Finally, it fits better with the underlying relational databases that make an important category of sources for an integration platform. On the flip side, typing becomes more difficult as both tuples and lists need to be typed.

A logical query plan is simply an XAlgebra expression. The input and output of most operators is a set of tuples $\{b_i \mid i = 1, \ldots, n\}$. Each tuple $b_i \equiv [\$var_1 = val_1^i, \ldots, \$var_k = val_k^i]$ consists of variable-value pairs, also referred to as bindings. We say that the variable $\$var_j$ is bound to the value val_j^i in the tuple b_i if the pair $\$var_j = val_j^i$ appears in b_i. All input (resp. output) tuples of an operator have the same list of variables and no variable appears more than once in a tuple. Each value val_j^i can either be a constant, NULL, a single element, a list of elements or a set of tuples. Support for NULL bindings enables easier handling of semistructured data as well as easier implementation of “traditional” operators that are defined using NULLs, such as outerjoins.

XAlgebra contains different implementations of the same logical operation (*e.g.*, grouping, join) as separate operators. This allows the rewriting optimizer to consider the choice of the particular implementation together with other optimization opportunties.

4 Related Work

Data integration has been an important database topic for many years. Most of the early works focused on the integration of structured sources - primarily relational databases. A survey and summary of such works can be found in [45, 21, 30]. In the 90’s the scope of data integration systems was extended to include the integration of autonomous and non-structured sources and the “mediator” concept was created [47]. EXIP follows the architecture of earlier virtual view mediators, such as TSIMMIS [19], YAT [8], HERMES [44], Garlic [6], and the Information Manifold [28].

XAlgebra is the cornerstone of query processing in EXIP, similarly to the roles that algebras play in relational [20] and object-oriented databases [11]. Algebras were also designed for the nested relational [40] and complex value models [1]. Recently XML Query Algebras have been proposed as the underlying infrastructure of XML databases and mediators [18, 8, 31]. The common characteristic of those algebras is that the operators input and output sets of tuples. This should be contrasted with functional programming-based XML algebras, such as [4] and [13], which serves as the core of the semantics of the emerging XQuery W3C XML querying standard [7]. In those approaches the operators input and output lists of XML elements. The tuple orientation, which is also present in object-oriented algebras, allows one to carry proven aspects of relational, nested relational and object-oriented algebras into XML processing. For example, it facilitates the specification of a join operator similar to the one of relational algebra. The XAlgebra relates most to the OQL algebra [11] and the XMAS algebra [31].

The component of the rewriting optimizer that is responsible for capability-sensitive decomposition is based on the conceptual background set up in [34, 46, 39], which, in turn, are related to the background created by works on answering queries using views either in the relational model (see [25] for a comprehensive survey) or semistructured/XML models [36, 17]. The system architecture of that component is related to the ones of [23, 8] in the sense that it is built around a rewriting optimizer, such as the one of Starburst [22].

Many of the query processing challenges of the Mediator's query processor are also faced by systems that export an XML view of a single relational source [41, 42, 15, 14].

On the commercial front, other data integration companies and corresponding systems have emerged during the last few years, such as Callixa (`www.callixa.com`) and Metamatrix (`www.metamatrix.com`). More recently, the adoption of XML and its perfect fit to integration applications has led to the emergence of other XML-based information integration companies, such as Xyleme [48] and Nimble [12].

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] F. Bancilhon, S. Cluet, and C. Delobel. A query language for the O_2 object-oriented database system. In *DBPL*, pages 123–138, 1989.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001. Available at `http://www.sciam.com/2001/0501issue/0501berners-lee.html`.

[4] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. ACM SIGMOD*, 1996.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Rewriting of regular expressions and regular path queries. In *ACM PODS Conf.*, pages 194–204, 1999.

[6] M.J. Carey et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. RIDE-DOM Workshop*, pages 124–31, 1995.

[7] D. Chamberlin et al. XQuery 1.0: An XML query language. W3C working draft, available at `http://www.w3.org/TR/xquery/`.

[8] V. Christophides, S. Cluet, G. Moerkotte, and J. Simeon. On wrapping query languages and efficient xml integration. In *ACM SIGMOD Conf.*, pages 141–152, 2000.

[9] V. Christophides, S. Cluet, and G. Moerkotte. Evaluating queries with generalized path expressions. In *ACM SIGMOD Conf.*, pages 413–423, 1996.

[10] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion! In *ACM SIGMOD Conf.*, pages 177–188, 1998.

[11] S. Cluet and G. Moerkotte. Nested queries in object bases. In *Database Programming Languages (DBPL-4), Proceedings of the 4th Int'l Workshop on Database Programming Languages - Object Models and Languages*, Workshops in Computing, pages 236–242. Springer, 1993.

[12] D. Draper, A. Y. Halevy, and D. S. Weld. The Nimble integration engine. In *ACM SIGMOD Conf.*, 2001.

[13] P. Fankhauser et al. XQuery 1.0 formal semantics. W3C working draft, available at `http://www.w3.org/TR/query-semantics/`.

[14] M. Fernandez, A. Morishima, and D. Suciu. Efficient evaluation of xml middle-ware queries. In *ACM SIGMOD Conf.*, 2001.

[15] M. Fernandez, A. Morishima, D. Suciu, and W. Chiew Tan. Publishing relational data in xml: the silkroute approach. *IEEE Data Engineering Bulletin*, 24(2):12–19, 2001.

[16] Mary Fernndez, Jonathan Marsh, and Marton Nagy. XQuery 1.0 and XPath 2.0 data model. W3C working draft, available at `http://www.w3.org/TR/query-datamodel/`.

[17] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *ACM PODS Conf.*, pages 139–148, 1998.

[18] L. Galanis et al. Following the paths of XML data: An algebraic framework for XML query evaluation. Available at `http://www.cs.wisc.edu/niagara/`.

[19] H. Garcia-Molina et al. The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, 8:117–132, 1997.

[20] H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2001.

[21] A. Gupta. *Integration of Information Systems: Bridging Heterogeneous Databases*. IEEE Press, 1989.

[22] L. Haas, J. Freytag, G. Lohman, and H. Pirahesh. Extensible query processing in Starburst. In *Proc. ACM SIGMOD Conf.*, pages 377–88, 1989.

[23] L. Haas, D. Kossman, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. VLDB*, 1997.

[24] Laura Haas, Donald Kossman, Edward Wimmers, and Jun Yang. An optimizer for heterogeneous systems with non-standard data and search capabilities. *Special Issue on Query Processing for Non-Standard Data, IEEE Data Engineering Bulletin*, 19:37–43, December 1996.

[25] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 2001.

[26] J.Hammer, M. Breunig, H.Garcia-Molina, S.Nestorov, V.Vassalos, and R. Yerneni. Template-based wrappers in the tsimmis system. In *Proc. ACM SIGMOD*, pages 532–535, 1997.

[27] H. F. Korth and M. A. Roth. Query languages for nested relational databases. In *Nested Relations and Complex Objects in Databases*, pages 190–204. Springer-Verlag, 1989.

[28] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. VLDB*, pages 251–262, 1996.

[29] A. Levy, A. Rajaraman, and J. Ullman. Answering queries using limited external processors. In *Proc. PODS*, pages 237–37, 1996.

[30] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 23:267–293, 1990.

[31] B. Ludascher, Y. Papakonstantinou, and P. Velikhov. Navigation-driven evaluation of virtual mediated views. In *Proc. EDBT Conf.*, 2000.

[32] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. VLDB Conf.*, 1996.

[33] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proc. ICDE Conf.*, pages 251–60, 1995.

[34] Y. Papakonstantinou, A. Gupta, and L. Haas. Capabilities-based query rewriting in mediator systems. In *Proc. PDIS Conf.*, 1996.

[35] Y. Papakonstantinou, A. Gupta, and L. Haas. Capabilities-based query rewriting in mediator systems. *Distributed and Parallel Databases*, 6:73–110, 1998.

[36] Y. Papakonstantinou and V. Vassalos. Query rewriting for semistructured data. In *ACM SIGMOD Conf.*, pages 455–466, 1999.

[37] Y. Papakonstantinou and V. Vassalos. The Enosys Markets Data Integration Platform: Lessons from the trenches. In *Proc. CIKM Conf.*, 2001.

[38] M. Petropoulos, V. Vassalos, and Y. Papakonstantinou. XML query forms (XQForms): Declarative specification of XML query interfaces. In *Proc. WWW10 Conf.*, 2001.

[39] A. Rajaraman, Y. Sagiv, and J. Ullman. Answering queries using templates with binding patterns. In *Proc. PODS Conf.*, pages 105–112, 1995.

[40] M. A. Roth, H. F. Korth, and A. Silberschatz. Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 13:389–417, 1988.

[41] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, and J. Funderburk. Querying XML views of relational data. In *VLDB Conf.*, pages 261–270, 2001.

[42] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently publishing relational data as XML documents. In *VLDB Conf.*, pages 65–76, 2000.

[43] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *Proc. VLDB Conference*, 1999.

[44] V.S. Subrahmanian et al. HERMES: A heterogeneous reasoning and mediator system. Available at `http://www.cs.umd.edu//projects/hermes/publications/abstracts/hermes.html`.

[45] G. Thomas et al. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 23:237–266, 1990.

[46] V. Vassalos and Y. Papakonstantinou. Expressive capabilities description languages and query rewriting algorithms. *Journal of Logic Programming*, 43(1):75–123, 2000.

[47] G. Wiederhold. Intelligent integration of information. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 434–437, 1993.

[48] L. Xyleme. A dynamic warehouse for XML data of the web. In *IDEAS*, pages 3–7, 2001.

Computing web page importance without storing the graph of the web (extended abstract)

S. Abiteboul
INRIA and Xyleme

M. Preda
Xyleme

G. Cobena
INRIA

Abstract

The computation of page importance in a huge dynamic graph has recently attracted a lot of attention because of the web. Page importance or page rank is defined as the fixpoint of a matrix equation. Previous algorithms compute it off-line and require the use of a lot of extra CPU as well as disk resources in particular to store and maintain the link matrix of the web. We briefly discuss a new algorithm that works on-line, and uses much less resources. In particular, it does not require storing the link matrix. It is on-line in that it continuously refines its estimate of page importance while the web/graph is visited. When the web changes, page importance changes as well. We modify the algorithm so that it adapts dynamically to changes of the web. We report on experiments on web data and on synthetic data.

1 Introduction

All the pages on the web do not have the same "importance". For example, Le Louvre homepage is more important that an unknown person's homepage. Page importance information is very valuable. It is used by search engines to display results in the order of page importance [4]. It is also useful for guiding the refreshing and discovery of pages: important pages should be refreshed more often[1] and when crawling for new pages, important pages have to be fetched first [2]. Following some previous ideas of [7], Page and Brin proposed a notion of page importance based on the link structure of the web [11]. This was then used by Google with a remarkable success. Intuitively, a page is important if there are many important pages pointing to it. This leads to a fixpoint computation by repeatedly multiplying the matrix of links between pages with the vector of the current estimate of page importance until the estimate is stable, i.e., until a fixpoint is reached.

The main issue in this context is the size of the web, billions of pages [6, 8]. Techniques have been developed to compute page importance efficiently, e.g., [5]. The web is crawled and the link matrix computed and stored. A version of the matrix is then frozen and one separate process computes page importance, which may take hours or days for a very large graph. So, the core of the technology for the off-line algorithms is fast sparse matrix multiplication (in particular by extensive use of parallelism). This is a classical area, e.g., [12]. Our algorithm computes the importance of pages on-line while crawling the web.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] Google seems to use such a policy for refreshing pages. Xyleme [13] does.

Intuitively speaking, some "cash" is initially distributed to each page and each page when it is crawled distributes its current cash equally to all pages it points to. This fact is recorded in the history of the page. The importance of a page is then obtained from the "credit history" of the page. The intuition is that the flow of cash through a page is proportional to its importance. It is essential to note that the importance we compute does not assume anything about the selection of pages to visit. If a page "waits" for a while before being visited, it accumulates cash and has more to distribute at the next visit.

The web changes all the time. With the off-line algorithm, we need to restart a computation. Although techniques can be used to take into account previous computations, several costly iterations over the entire graph have to be performed by the off-line algorithm. We modify our on-line algorithm to adapt to changes. Intuitively, this is achieved by taking into account only a recent window of the history.

We report on experiments with variants of our algorithms, synthetic data and real data obtained from real crawls of the web. One of the algorithms presented here is used in production by the Xyleme company [13].

The present paper is just an extended abstract. The full paper may be obtained from [1]. The paper is organized as follows. In Section 2, we introduce the algorithm focusing on static graphs. In Section 3, we move to dynamic graphs, i.e., graphs that are continuously updated like the web. The following section deals with implementation and discusses some experiments. The last section is a conclusion.

To conclude this introduction, we recall the standard notion of page importance that we use here.

Page importance or page rank is defined as the fixpoint of a matrix equation [11, 10]. Let G be a graph (e.g., the web) represented as a link matrix $L[1..n, 1..n]$. If there is an edge for i to j, $L[i, j] = 1$, otherwise it is 0. Let $out[1..n]$ be the vector of out-degrees. Let L' be the matrix defined by, for each i, j, $L'[i, j] = \frac{L[i,j]}{out[i]}$. The importance Imp of nodes as defined by [11] is given by:

$$Imp[j] = \sum_i (L'[i, j] * Imp[i])$$

This definition and the computation as a fixpoint are mathematically founded. One can show that the importance of a page is the probability to be on that page after an infinite random walk on the graph. So it is the steady vector of a Markov chain with a transition matrix L'. The importance can be computed as the limit of X_n for $X_n = L' * X_{n-1}$ starting, for instance, from a vector that gives equal importance to all pages. To be more precise, this holds under some hypothesis: (i) the graph has to be [9] aperiodic (a reasonable assumption on the web) and (ii) strongly connected. The mathematical model is slightly modified to take into account the fact that the web is not strongly connected. Details may be found in [1].

2 The algorithm for static graphs

We present in this section the algorithm for the case of a static graph (no update).

To start, we distribute some initial cash to each node, e.g., if there are n nodes, we distribute $1/n$ to each node. For every page, we keep two values. We call the first *cash*. In some sense, it records the recent information discovered about the page, more precisely, the sum of the cash obtained by the page since the last time it was crawled. We also record the *(credit) history* of the page, the sum of the cash obtained by the page until the last time it was crawled. The cash is stored in main memory whereas the history may be stored on disk. When a page is retrieved by the web agent, we know the pages it points to. In other words, we have at no cost the link information for the retrieved page. We record its cash in the history, i.e., we add it to the history. We also distribute this cash equally between all pages it points to. We reset the cash of the page to 0. This happens each time we read a page. We will see that this provides enough information to compute the importance of the page as used in standard methods. We will consider in a further section how this may be adapted to handle dynamic graphs.

More precisely, we use two vectors $C[1..n]$ (the cash) and $H[1..n]$ (the history). The initialization of C has no impact on the result. A more detailed history will be needed only when we move to an adaptive version of the algorithm. For now, $H[i]$ is simply a number that accumulates all the cash page i received. As history H is stored on disk, $H[i]$ is only updated when page i is crawled. The algorithm works as follows:

```
On-line Page Importance Computation (OPIC) Algorithm

for each i let C[i] := 1/n ;
for each i let H[i] := 0 ;
let t:=0 ;
let G:=0 ;
do forever
    begin
    t := t+1;
    choose some node i ;
    %% assuming each node is selected infinitely often
    H[i] += C[i];
    for each child j of i, C[j] += C[i]/out[i];
    G += C[i];
    C[i] := 0 ;
    end
```

Note that the algorithm does not impose any requirement on the order we visit the nodes of the graph as long as each node is visited infinitely often, so it can be applied "online" i.e. when pages are crawled. This dramatically reduces the system complexity and the usage of resources. At the time we visit a node (we crawl it), the list of its children is available on the document itself and does not require disk access. As long as the cash of children is stored in main memory, no disk access is necessary to update it.

Let C_t and H_t be the values of C and H at the end of the t-th step of the algorithm. The vector C_0 denotes the value of C at initialization (all entries are $1/n$). Let X_t be defined by:

$$X_t = \frac{H_t}{(\sum_i H_t[i])}, \text{i.e., } \forall j, X_t[j] = \frac{H_t[j]}{(\sum_i H_t[i])}$$

One can prove that, when t goes to infinity, $|H_t|$ goes to infinity and for each j,

$$\left|(L' * X_t)[j] - X_t[j]\right| < \frac{1}{\sum_i H_t[i]}$$

and $|X_t| = 1$. Thus the vector X_t converges to the vector of importance, i.e., to Imp.

This shows that

$$\frac{H_t[j] + C_t[j]}{(\sum_i H_t[i]) + 1}$$

provides an estimate for the importance of a page, and that it converges to it.

In other words, at each step of the computation, an estimate of the importance of page k is given by $(H[j] + C[j])/(G+1)$.

The OPIC algorithm uses only local information, i.e. the outgoing links of the page that is being crawled that can be found in the page as URLs. Thus our algorithm presents the following advantages: (i) it requires less storage resources than standard algorithms; (ii) it also requires less CPU, memory and disk access than standard algorithms; (iii) it is less complex because the web agent and the evaluator of page importance can run in a single process.

As mentioned earlier, OPIC converges independently on how pages are selected (assuming each page is selected infinitely often). However, the speed of convergence depends on this selection. For instance, consider the following two refresh strategies:

1. RANDOM: We choose the next page to crawl randomly with equal probability.

2. GREEDY: We read next the page with highest cash. This is a greedy way to increase G for each page selected. (The exact policy used in Xyleme is more complex than GREEDY but comes very close to it.)

It turns out that, as we shall see in Section 4, the convergence is much faster with GREEDY.

Note that the GREEDY selection strategy suggests a totally different way of computing importance. It turns out that if GREEDY is used, the importance of a page coincides with the frequency with which the page is read. If GREEDY is used, it thus suffices to store the times of the visits of the page to obtain that frequency and thus an estimate of the importance of the page.

3 Dynamic graph

In this section, we consider the case of a changing graph and modify the algorithm to adapt to changes.

Consider a dynamic graph (the case of the web). Pages come and disappear and edges too. To simplify, we will ignore here the changes of the set of nodes and focus on the changes of edges. So, we assume that the set of nodes is fixed. Note that the importance of a page is now a moving target and that we should not expect to obtain a perfect estimate. We may only hope to stay close to it if this importance does not change too rapidly, which is typically the case on the web. The problem with OPIC on a changing graph is that its estimate of page importance would typically change slowly and even more slowly when the graph gets older because the past history would simply have too much weight. This suggests using only some recent history, e.g., say the history that has been gathered during the last month, or two months.

To fix the ideas, we have to use some time basis. If we use a clock, results would be influenced by the speed of crawling. So, instead, a more appropriate time basis is the current value of G. So, now we think of the variable G as the current time.

Consider some time instants $t, t+T$. Let $H_{t,t+T}[i]$ be the total of cash added to the history of page i between time t and $t+T$, i.e., $H_{t+T}[i] - H_t[i]$. Let

$$\forall j, X_{t,t+T}[j] = \frac{H_{t,t+T}[j]}{(\sum_i H_{t,t+T}[i])}$$

Because the theorem did not impose any condition on the initial state of X_t, we know that $X_{t,t+T}$ converges to the vector of importance when T goes to infinity[2]. This comes to ignoring the history before time t and start with the state of the cash at time t for initial state.

To adapt to changes on the web, we therefore use time windows. We estimate the importance of a page by looking at the history between $G-T$ and G. We call the interval $[G-T, G]$ the (time) *window*. If we choose a very large window, we get a good estimate of importance (a perfect one with an infinite window: $T \to \infty$). On the other hand, a too large window means that we take into account very old information that may have changed, so our reaction to change may be too slow. Now, a very small window would be very adaptive to changes but may yield a too important error in the estimate. Thus there is a trade-off between precision and adaptability to changes and a critical parameter of the technique is the choice of the size of the window.

To compute page importance for a changing graph, we use the Adaptive OPIC algorithm based on time windows. In Adaptive OPIC, we use the cash vector in the same way as in the case of a static graph. The global variable G records the total sum of cash that has been distributed to the nodes of the graph since the initialization

[2] On the other hand, when t goes to infinity, $X_{t,t+T}$. does not converge to the vector of importance.

of the process (the current time). To avoid having a value that grows infinitely, we keep this value modulo some very large number G_0. (G_0 should be large enough so that no confusion may arise.) For the history, we keep a certain number of measures. We studied two variants of the algorithm based on the following policies:

1. FIXED WINDOW: For every page i, we store the value of cash $C_t[i]$ and the global value G_t for all times it was crawled in a fixed time window, say the last m months.

2. VARIABLE WINDOW: For every page i, we store the value of cash $C_t[i]$ and the global value G_t for the last n times it was crawled.

The Xyleme system is actually using a variant of FIXED WINDOW called BAO (for Basic Adaptive OPIC). Let T be the size of the window. The idea is that the history of a page consists of a single value, an *estimate* of what the page obtained in an interval T before the last crawl. The next time the page is crawled a new estimate is computed by interpolation based on the history and the cash it gathered between the two crawls. Details are omitted.

4 Experiments

We implemented and tested first the standard off-line algorithm for computing Page Importance, then OPIC and several variants of Adaptive OPIC, and BAO. We performed some tests on synthetic data and some on a much larger collection of URLs from the web.

Synthetic data *The graph model:* We performed most of the experiments with (reasonably small) graphs of about 100 000 nodes with a rather simple random distribution of edges. This was to be able to experiment with many variants of the algorithm, many sizes of windows and many patterns of changes (from very small changes to very intense ones). We also performed tests with much larger graphs and with much more complex edge distribution. We found that the results were not depending so much on these two criteria. For instance, to study the impact of the distribution of incoming/outgoing links and the correlations between them, we tried several graph models in the spirit of [3]. Even with significant changes in parameters describing the graphs, the patterns of the results did not change substantially from the simple graph model.

Convergence: We studied the convergence of OPIC for various page selection policies. In particular, we considered RANDOM and GREEDY. We found OPIC converges much faster with GREEDY than with RANDOM. Typically, a “reasonable error” (less than one percent) is obtained with GREEDY when each page has been read about 5 times on average. RANDOM is roughly twice slower. We also compared these, somewhat artificially, to the off-line algorithm. In the off-line, we count N steps for each iteration of the off-line algorithm. With appropriate tuning, OPIC with the GREEDY page selection policy presents the following two advantages over the off-line algorithm:

- it obtains very fast (after reading each page once or twice on average) a reasonable estimate even if the off-line algorithm eventually (after a few iterations) converges faster.

- it is tailored to provide good estimates for important pages which is what it useful in practice.

The fact that the off-line algorithm eventually converges faster (roughly twice faster) is not surprising since more information is available to it at each step.

Impact of the size of the window: As already mentioned, a small window means more reactivity to changes but at the cost of some lack of precision. A series of experiments was conducted to determine how much. Consider for instance the GREEDY policy and VARIABLE WINDOW (we keep the history for the last n crawls of the page). When n is too small (e.g., $n = 2$), the error is too large and we do not converge to a reasonable estimate. Typically, a value of n around 4 seems a reasonable compromise. The error is limited and our reactivity to change is good. Depending on some parameters such as the update rate of the graph, different window sizes perform better.

Variant of the Adaptive algorithm: We compared FIXED WINDOW and VARIABLE WINDOW. We found that VARIABLE WINDOW performs in general better. This is because FIXED WINDOW has a tendency to accumulate more data than needed for important pages and too little for unimportant ones. On the other hand, when the graph changes very rapidly, FIXED WINDOW adapts better to changes. We also found that a a yet better strategy is to use different policies depending on the importance of the page.

The BAO algorithm (interpolation on FIXED WINDOW) turned out to yield surprisingly good results, often even better than VARIABLE WINDOW, while it uses less resources. We believe that the reason for this is that it avoids some “noise” at each crawl of a page resulting from the introduction of a new measure and the deletion of an old one. The interpolation acts, in some sense, as a filter. We selected BAO for the intensive experiment with web data.

Web data We tested BAO with a real crawl of the web. We used a cluster of Linux PCs, between 4 and 8 PCs at various times of the experiment. The experiment lasted several months with the PCs active only about 50% of the time. Each PC can read about 3 million pages per day. The selection of pages was such that pages estimated as important were read first and refreshed more often. (As mentioned earlier, the refresh strategy gives results close to those of GREEDY so a page twice more important is read twice more often.)

We discovered close to one billion URLs. Some pages were never read: only 400 million of them were actually read, so we computed Page Importance over 400 million pages, to a certain extent the 400 million most important pages of the web. The top most important pages were read almost daily. We used BAO, i.e., FIXED WINDOW with interpolation. This experiment was sufficient (with limited human checking of the results) to conclude that the algorithm can be used in a production environment. Indeed, this is the algorithm currently used by Xyleme. During the test, we experimented with various sizes of the times windows. After a trial/error phase, the time window was fixed to three weeks.

More experiments are reported in the full paper.

5 Conclusion

We proposed a simple algorithm to implement with limited resources a realistic computation of page importance over a graph as large as the web. We demonstrated both the correctness and usability of the technique.

To understand more deeply the algorithms, more experiments are being conducted. Also, we are working on a precise mathematical analysis of the convergence speed of the various algorithms. One goal is to obtain variants of Adaptive OPIC that provide better estimates of page importance and also give an estimate of the error and the change rate of this importance. This includes critical aspects such as the selection of the appropriate size for the time window. Other aspects of our algorithm may also be improved such as the management of newly discovered pages that we ignored here.

Acknowledgments We want to thank Laurent Mignet and Tova Milo for discussions on the present work.

References

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. *Technical report*, 2002. http://www-rocq.inria.fr/verso.

[2] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[3] Colin Cooper and Alan M. Frieze. A general model of undirected web graphs. In *European Symposium on Algorithms*, pages 500–511, 2001.

[4] Google. www.google.com/.

[5] T. Haveliwala. Efficient computation of pagerank. *Technical report, Stanford University*, 1999.

[6] A. Broder K. Bharat. Estimating the relative size and overlap of public web search engines. *7th International World Wide Web Conference (WWW7)*, 1998.

[7] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[8] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[9] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computing Surveys*, 28(1):33–37, 1996.

[10] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web, 1998.

[11] L. Page S. Brin. The anatomy of a large-scale hypertextual web search engine. *WWW7 Conference, Computer Networks 30(1-7)*, 1998.

[12] S. Toledo. Improving the memory-system performance of sparse-matrix vector multiplication. *IBM Journal of Research and Development*, 41(6):711–??, 1997.

[13] Xyleme. www.xyleme.com/.

Analyzing fine-grained hypertext features for enhanced crawling and topic distillation

Soumen Chakrabarti
Indian Institute of Technology Bombay[*]

Ravindra Jaju
Indian Institute of Technology Bombay

Mukul Joshi
Indian Institute of Technology Bombay

Kunal Punera
Indian Institute of Technology Bombay

Abstract

Early Web search engines closely resembled Information Retrieval (IR) systems which had matured over several decades. Around 1996–1999, it became clear that the spontaneous formation of hyperlink communities in the Web graph had much to offer to Web search, leading to a flurry of research on hyperlink-based ranking of query responses. In this paper we show that, over and above inter-page hyperlinks, much semantic information can be teased out of the manner in which markup tags, such as menu-bars, tables, and lists are used to organize pages, and the context in which hyperlinks are made from a page to another. We believe that delving into fine-grained page structure will be the next wave in hypertext mining, bridging some of the gap between unstructured HTML and relatively structured XML, and preparing the stage for deeper analysis into page snippets. We talk about two applications to illustrate fine-grained page analysis: topic-guided focused crawling, and hyperlink-based ranking which takes page structure into account. In both cases, we show that exploiting fine-grained structure enhances the performance of our systems.

1 Introduction

Traditional IR has considered the *document* as the fundamental unit of analysis [18, 17]. The application of IR to Web search has continued this view, regarding whole *Web pages* as documents. Crawlers fetch content in units of whole pages, and rarely discriminate between outlinks on semantic grounds when they scan a page for new URLs to be crawled. Search engines return ranked lists of whole pages as well. Recent link analysis algorithms which rank pages on the basis of the prestige bestowed by inlinks [2, 11] (a technique called *topic distillation*) also model the Web graph at a coarse grain, with whole pages as single nodes. A score is attributed to each page and links from one page to another are viewed as a recommendation of all of the target page's content by the link creator. This *coarse-grained* approach to crawling and ranking betrays an assumption that Web pages are homogeneous in their contents, and all outlinks are equally related to the contents.

We believe that a great deal of semantic information, latent in the HTML sub-structure, is lost when one considers Web pages as indivisible. HTML tags and their arrangement as the Document Object Model or DOM tree (`http://www.w3.org/DOM/`) provide clues which prove valuable for crawling and IR tasks. Therefore, we enhance the coarse-grained graph model for the Web to a **fine-grained model** in which every

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[*] Contact author, email `soumen@cse.iitb.ac.in`

page is represented by a DOM tree, with edges internal to a DOM tree representing local tag-tree structure and ordinary hyperlinks as edges connecting two DOM trees.

At the Laboratory for Intelligent Internet Research[1] in IIT Bombay, we are using this fine-grained model for improving a number of Web IR and mining tasks. In this paper we concentrate on two applications: *focused crawling* [6] and *topic distillation* [3, 5]. We intend to make our prototypes (HTML parser and cleaner, crawler, and distillation programs) available in the public domain for research use.

The rest of the paper is arranged as follows. Section 2 describes our data model and the various issues in the parsing and sanitization of HTML pages. In the next two sections we outline two applications of the fine-grained model. Section 3 describes the how the new model can be used to improve the rate at which a focused crawler acquires relevant pages. In Section 4 we show that the application of the fine-grained model to ranking Web pages improves full-page approaches. We conclude with comments on ongoing and future work in Section 5.

2 HTML tags and parsing

In a perfectly engineered world, Web pages will be written in a markup language more sophisticated than HTML, with a universally accepted meta-data description format. XML (`http://www.w3.org/XML/`) and RDF (`http://www.w3.org/RDF/`) have made great strides in that direction, especially in specific domains like e-commerce catalogs, electronic components, genomics, and molecular chemistry. However, we believe that the Web will always remain adventurous enough that its most interesting and timely content can never be shoehorned into rigid schemata. Without downplaying the importance of XML and associated standards, we emphasize the need for HTML to DOM converters which capture the fine-grained Web page model accurately.

HTML is primarily used to specify the visual formatting of the content of a Web page, but some Web authoring idioms and the organizational functions of certain tags also help us derive semantic information. Creators of links between Web pages normally provide some information about the target page around the URL on the source page, not necessarily in the anchor text alone. E.g., a section may start with "`<h1>Japanese car makers</h1>`" and continue with a list of links, including `http://www.honda.com/` and `http://www.mazda.com/`, sites which may choose not to describe themselves in those words. Furthermore, the occurrence of two such links at close proximity (or inside, say, a `<td></td>` tag) might suggest that they point to pages similar in content. Conversely, while crawling for pages about mountain biking, we would do well to determine the DOM subtrees which are semantically dissimilar and avoid exploring links emerging from irrelevant subtrees.

The DOM tree of a page can be obtained by parsing the HTML. This, however, is not as easy as it seems. A large fraction of HTML pages on the Web violates the HTML standard. Patching the syntax is non-trivial because we wish to retain as much DOM information as we can for later analyses. For example, a `<tr>` tag implies that a `<table>` tag must have already occurred somewhere prior to it. If an enclosing `<table>` tag does not exist, it has to be inserted, but we must make sure that such an insertion does not violate other HTML syntax rules. An actual HTML page had a `<form>` tag starting outside a `<table>`, and closing within it. This caused an early version of the cleaner to close the `<table>` tag before the `</form>` tag, leading to loss in both formatting and information.

No movement of text or tags is done in the cleaning phase itself because not only can it change, albeit unintentionally, what the original author probably meant to convey, but it also makes the code increasingly complex and consequently prone to errors. Movement, especially of empty tags which might be created by the cleaner due to inappropriate use of tags, is attempted in the DOM tree construction phase using a variety of heuristics. The default patching policies can be modified by the user through a simple API.

[1] `http://www.cse.iitb.ac.in/laiir/`

3 Focused crawling guided by DOM structure

The Web has over two billion static pages today and an estimated 600 gigabytes of text changes per month [10]. General-purpose crawlers dropped to a mere 18% coverage of the Web in 1999 [13]. Matters have improved since, Google now covering over half the estimated extent of the Web. However, given the size of the Web, coverage often comes at the price of freshness, large crawlers visiting most sites every several weeks at best.

In contrast to general purpose crawlers, a **focused crawler** seeks to collect pages related to specific topics, which it learns from examples provided by the administrator. A federation of focused crawlers can cover each specific topic in more depth, and succeed in keeping crawls fresher. They start from example URLs on a certain *focus topic*, say, basketball, and explore outlinks with the goal of collecting pages about basketball and avoiding pages that are not about this topic.

We initiated the design and study of focused crawlers in 1999 [7], and several enhancements have been suggested since then [9, 15]. The key operation of a focused crawler is to estimate the worth of fetching a new page v based on its citation found on a page u that has already been fetched. Such decisions are made by training a *text classifier* [14] on the example URLs.

3.1 Fine-grained focused crawling

Under the coarse-grained model, a focused crawling system can avoid fetching irrelevant pages (pages not about the focus topic) by not crawling links that it finds on irrelevant pages. For example, the relevance of u to focus topic c_* (which, in a Bayesian setting we can denote by $\Pr(c_*|u)$) could be used as an estimate of $\Pr(c_*|v)$, the relevance of unvisited outlink to page v. The assumption made here is that if a page is not relevant to the focus topic, the chances that links found on that page point to relevant resources are remote (in other words, that pages across a hyperlink are more similar than two randomly chosen pages on the Web). For obtaining $\Pr(c_*|u)$ we use the RAINBOW naive Bayes classifier by McCallum and others [14]. The classifier is trained offline on examples taken from a taxonomy such as Yahoo! (`http://www.yahoo.com/`) or the DMoz Open Directory (`http://dmoz.org/`).

Consider, however, a Web page u which talks about various topics in general, but has a few links to very good resources/pages (v_i) on basketball. Because u relates to sundry topics, it may not be considered relevant by a focused crawler crawling pages on basketball, and the links to basketball resources (v_i) will remain unexplored. Conversely, a page may contain extremely relevant text but some links to irrelevant pages (see the examples in Figure 8). These irrelevant links would be eagerly crawled because they were found on a highly relevant page. Clearly, the judgments on relevance of v_i ($\Pr(c_*|v_i)$) must be based on something more than just $\Pr(c_*|u)$.

In the fine-grained focused crawler [6] we use **two** classifiers. In a standard focused crawler, the classifier was used to assign priorities to unvisited frontier nodes v. This no longer remains its function. The role of assigning priorities to unvisited URLs in the crawl frontier is now assigned to a new learner called the *apprentice*, and the priority of v is specific to the (u, v) link which leads to it. Meanwhile the role of the standard classifier becomes one of generating instances for the apprentice, as shown in Figure 1. We may therefore regard the baseline learner as a *critic* or a *trainer*, which provides feedback to the apprentice so that it can improve "on the job."

We use RAINBOW for the apprentice classifier as well, suitably encoding the context of each link (u, v) so that the bag-of-words classifier can make use of these encoded features effectively. After parsing the HTML and creating a DOM tree, its leaves are numbered from left to right. The specific `<a href...>` which links to v is an internal node a_v, and is the root of the subtree which contains the anchor text of (u, v). The subtree may contain other nodes representing tags such as `<em>` or `<b>`. We regard all textual tokens found in this subtree to be *at offset zero* w.r.t. the (u, v) link. Tokens outside the subtree of a_v are associated with negative and positive offsets as shown in Figure 2. For efficiency, we limit the maximum offset up to which we accept tokens to some **window** around a_v of radius $d_{\max}$. Special strings can now be prefixed to the text tokens to encode their location

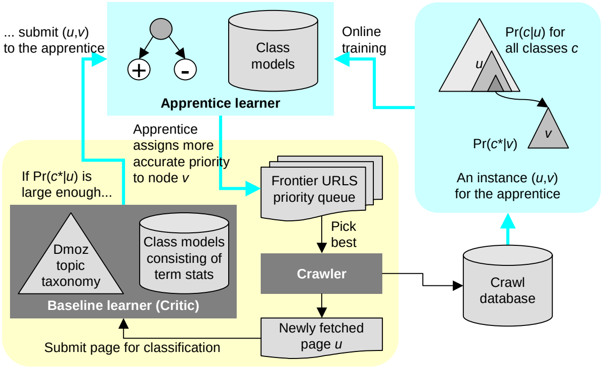

... submit (u,v)
to the apprentice
+
-
Class models
Apprentice learner
Online training
Pr(c|u) for all classes c
u
v
Pr(c*|v)
An instance (u,v) for the apprentice
Apprentice assigns more accurate priority to node v
If Pr(c*|u) is large enough...
Frontier URLS priority queue
Pick best
Dmoz topic taxonomy
Class models consisting of term stats
Baseline learner (Critic)
Crawler
Crawl database
Newly fetched page u
Submit page for classification

Figure 1: The apprentice is continually presented with training *cases* (u, v) with suitable features. Once trained, the apprentice helps assign unvisited nodes v better estimates of relevance.

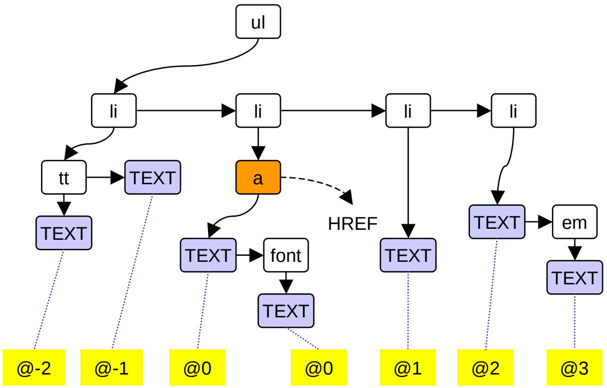


Figure 2: Numbering of DOM leaves used to derive offset attributes for textual tokens. '@' means "is at offset".

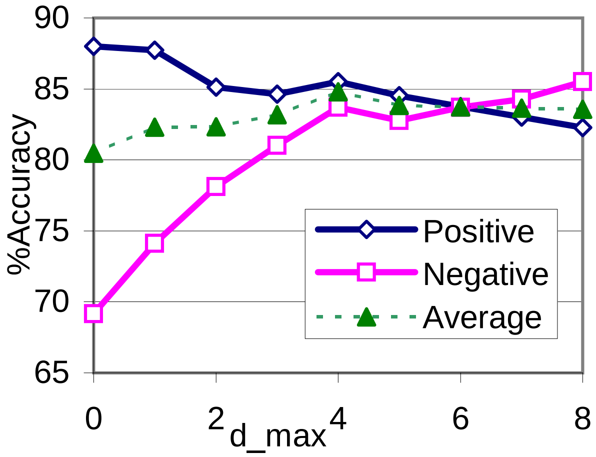


Figure 3: Apprentice accuracy visibly improves as $d_{\max}$ is increased, up to about 5–7, mainly because of an increased ability to correctly reject negative (low relevance) outlink instances.

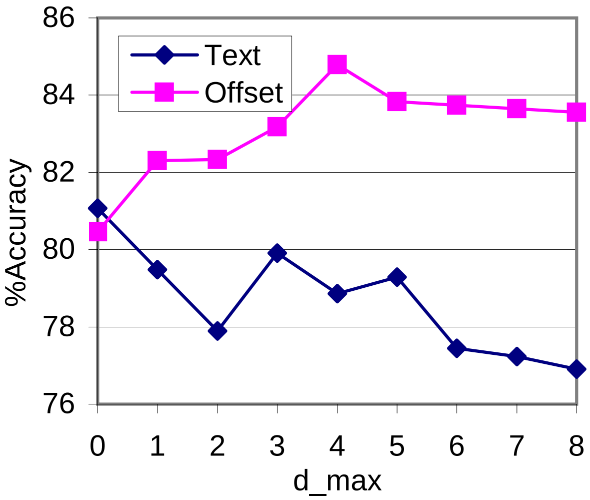


Figure 4: Using encoded offset as against plain text improves the apprentice's accuracy.

w.r.t. to the (u, v) link.

3.2 Experiments and Results

We experimented with focused crawls starting with examples of dozens of topics from roughly 450 topics chosen from DMoz. We excerpt some salient observations.

Window width: A key concern in the selection of the context of a link is the window size. In the typical example shown in Figure 3, we see that the accuracy of the apprentice classifier increases rapidly as $d_{\max}$ is increased above 0 (indicating that using only anchor text loses out on useful features) but the gains level out and may even fall for larger $d_{\max}$. Based on these observations we settled on a maximum window size of 5.

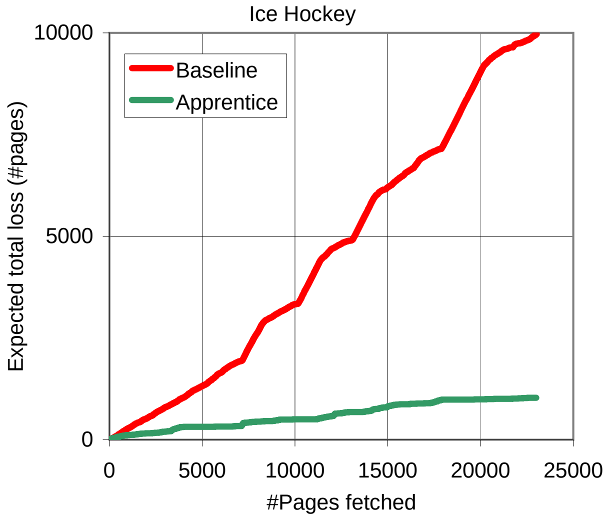


Figure 5: Guidance from the apprentice significantly reduces the loss rate of the focused crawler. In this chart the apprentice has been trained offline.

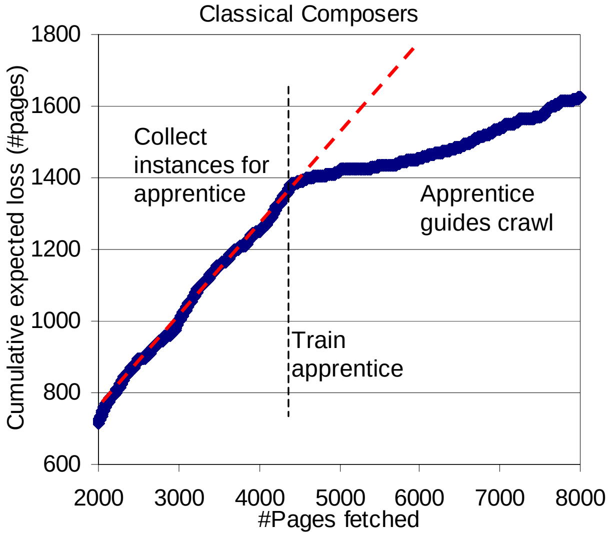


Figure 6: An example of training the apprentice *online* followed by using it for crawl guidance. Before guidance, loss accumulation rate is over 30%, after, it drops to only 6%.

Encoding DOM offsets in features: We verified that adding offset information to text tokens was better than simply using plain text near the link [4]. One sample result is shown in Figure 4. The apprentice accuracy decreases with $d_{\max}$ if only text is used, whereas it increases if offset information is provided. This highlights the importance of proper *feature extraction*.

Rate of acquiring relevant pages: The focused crawler's performance can be measured in terms to its *harvest rate* which is the fraction of relevant pages collected. If V is the set of nodes collected, the harvest rate is defined as $(1/|V|)\sum_{v \in V}\Pr(c_*|v)$. Alternatively, we can measure the *loss rate*, which is one minus the harvest rate, i.e., the (expected) fraction of fetched pages that must be thrown away. We present loss rates, because this is the part of the crawler's work which can be potentially avoided. Figure 5 contrasts the loss rate of the baseline and apprentice guided crawls. Figure 6 shows the sudden decrease in loss rate when the focused crawler switches from using the baseline classifier to the apprentice classifier once the latter has been sufficiently trained.

4 Topic Distillation

Once a focused crawler has collected a substantial corpus of the Web pages on a specified topic, we are usually interested in compiling a list of authoritative pages and a list of link collections pertaining to that topic [12]. Such lists can be computed using **topic distillation**, a process of discovering *authoritative* Web pages and comprehensive collections of links called as *hubs* which reciprocally endorse each other and are relevant to a given topic. Kleinberg's HITS [11] and the PageRank algorithm [2] underlying Google are the most well known topic distillation algorithms. In this paper we concentrate on HITS.

A key property of HITS is the diffusion of endorsement or popularity across cocited siblings. From the equations shown in the Figure 7 we notice that the increase in the authority score of v_1 gives rise to the increase in authority score of v_3 within the next two iterations. We call this phenomenon *authority diffusion*.

Because HITS models the hub page u as a single node with a single hub score, high authority scores could diffuse from a relevant authority v_1 to a less relevant cocited authority v_3. This makes HITS and related algo-

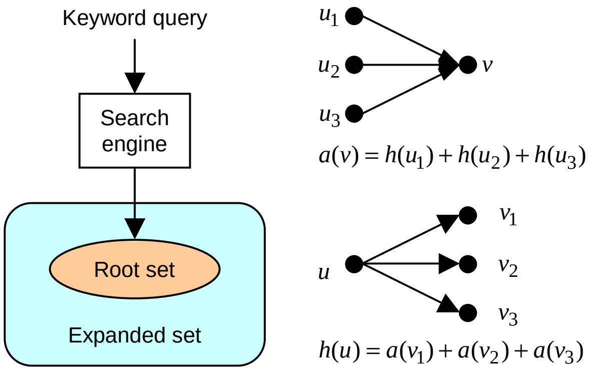


Figure 7: Hyperlink induced topic search (HITS).

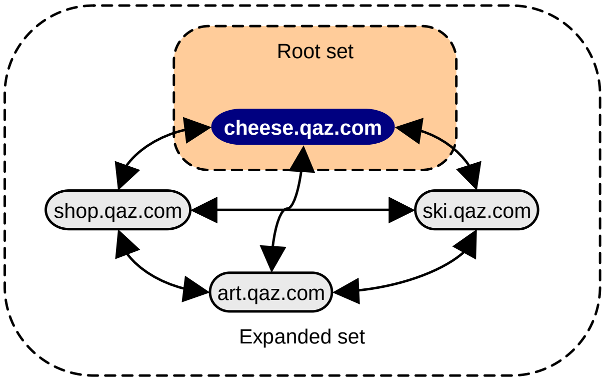


Figure 8: Spammed or mixed hubs can confound topic distillation.

rithms vulnerable to unintended *topic drift* and deliberate *clique attacks*. Topic drift may happen if v_1 is a studio homepage, v_3 is a page about Oscar nominations, and only one topic is of interest to the user. Clique attacks are perpetrated by site designers aiming to improve their clients' search engine ratings by artificially constructing dense hyperlink neighborhoods that appear like prestige-conferring citations (Figure 8). These links imply no semantic coherence or editorial judgment, which are crucial for the success of topic distillation. Banners, navigational panels, advertisements, and "Web ring" links add to the woes of traditional distillation algorithms.

4.1 Fine-grained topic distillation

Our enhancement to the standard topic distillation algorithm is largely contained in one step: the authority-to-hub score transfer. Recognizing that a single hub score per page may result in unwanted score diffusion, we represent each page using its DOM tree. To simplify the description for this paper, `<a href...>` nodes (Figure 4) are made *leaf* nodes, removing their subtrees. These leaves are called **microhubs**.

HITS in effect aggregates all microhub scores on a page and makes the sum the hub score for the whole page. We would like to aggregate microhub scores selectively. If we believe that some scores belong to relevant DOM subtrees, we would like to add them up and propagate them back to the authority nodes as in HITS. But we would *not* like to aggregate over microhub scores which we believe lead to irrelevant authorities.

There are (at least) two kinds of clues as to whether a DOM subtree is relevant. First, if a disproportionately large fraction (compared to the global average) of outlinks in a DOM subtree link to known good authorities, we should collapse the hub score of the subtree. Second, if the text contained in a subtree is highly relevant to the query (compared to neighboring subtrees), that subtree could represent a single hub.

Informally, our model of hub generation enables our algorithm to find a *frontier* across each hub's DOM tree. Subtrees attached to the frontier are made individual nodes in the distillation graph. Thus the hub score of the entire page is *dis-aggregated* at this intermediate level. The key question is: how do we determine such a frontier for each page? Owing to space limitations, we will take up this question with regard to the first clue above: the proximity of outlinks to good authorities.

Finding the best frontier: Our goal is to recover the *simplest* frontier which best explains any non-uniformity in the microhub scores. We wish to avoid choosing either one large and redundant frontier near the leaves and explain the many small and homogeneous subtrees very accurately, or choosing a short frontier near the root with a few large subtrees which are harder to model since they contain many diverse microhub scores.

```
$\mathbf{h} \leftarrow \mathbf{Ea}$
for each document DOM root $u$ do
    frontier $F \leftarrow \mathbf{segment}(\mathbf{u})$
    for each frontier node $v \in F$ do
        $h(v) \leftarrow \sum_{w \in L_v} h(w)$
        for each $w \in L_v$ do
            $h(w) \leftarrow h(v)$
        end for
    end for
end for
$\mathbf{a} \leftarrow \mathbf{E}^{\mathbf{T}}\mathbf{h}$
normalize $\mathbf{a}$ so that $\sum_u a(u) = 1$
```

Figure 9: One iteration of the fine-grained topic distillation algorithm. L_v is the leaf microhubs in the DOM subtree rooted at v.

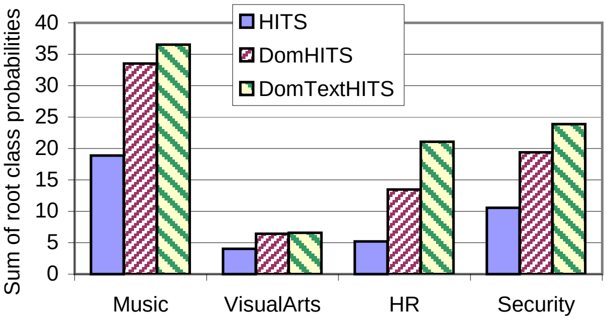


Figure 10: Authorities reported by DOMTEXTHITS have the highest probability of being relevant to the Dmoz topic whose samples were used as the root set, followed by DOMHITS and finally HITS. This means that topic drift is smallest in DOMTEXTHITS.

We strike a balance between these extreme choices by designing comparable units of cost for defining a frontier model (the **model cost**) and generating microhub scores from a frontier model (the **data cost**), and minimizing the sum of these two costs, following Rissanen's Minimum Description Length (MDL) principle [16].

For any node u, we can model the microhub scores H_u at the leaves of the tree rooted at u using a distribution Θ_u. The probability of a specific microhub score h w.r.t. this distribution is denoted $\Pr_{\Theta_u}(h)$. Let the global distribution at the root DOM node be Θ_0. Let the model cost Θ_u for a node u on the frontier, encoded relative to the global distribution Θ_0, be denoted $L(\Theta_u|\Theta_0)$. The data cost for the set H_u of microhub scores in the subtree rooted at u is approximately $-\sum_{h \in H_u} \log \Pr_{\Theta_u}(h)$ [8].

Finding the best frontier is NP-hard. The following greedy strategy worked quite well: at each node u, we compare two options:

- Make u a frontier node, paying $L(\Theta_u|\Theta_0)$ and $-\sum_{h \in H_u} \log \Pr_{\Theta_u}(h)$.
- Expand u to its children $\{v\}$, paying $\sum_v L(\Theta_v|\Theta_0)$ and $-\sum_v \sum_{h \in H_v} \log \Pr_{\Theta_v}(h)$.

We greedily pick the locally better option and continue until the frontier can no longer be improved by pushing it down. When a node is not worth expanding, we say it is **pruned**. The DOM segmentation steps above are integrated into the standard topic distillation algorithm by inserting a call to the **segment** routine after the $\mathbf{h} \leftarrow \mathbf{Ea}$ step and before the $\mathbf{a} \leftarrow \mathbf{E}^{\mathbf{T}}\mathbf{h}$ step (Figure 9).

We can use a similar segmentation technique for the text associated with DOM subtrees. The combined evidence from microhub scores and text can lead to a more reliable hub segmentation than using any one by itself. We call this algorithm as DOMTEXTHITS; the details of this algorithm have been reported elsewhere [5].

4.2 Experimental study

Our system can be used with ad-hoc queries as well as predefined topics; all we need is a subgraph of the Web with a significant fraction of the pages being relevant to a broad topic. We used the set of 28 queries used in the ARC/Clever system and other researchers [1, 4]. We also picked 20 topics from a simplified version of the DMoz topic hierarchy (`http://dmoz.org`). There were about 4 million non-local `HREF`s and over 15 million fine-grained nodes and links in our data sets. It was easily evident that our system was less susceptible to topic drift and clique attacks than HITS. We do not have space to report anecdotes, so we instead focus on some quantitative measurements.

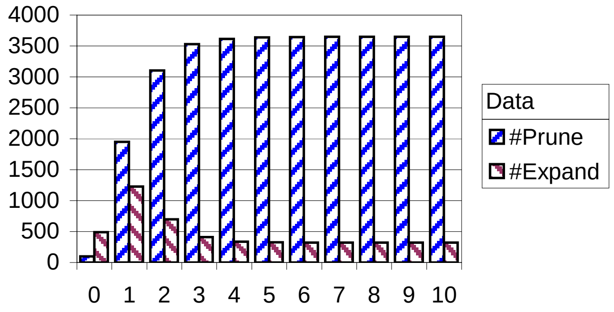


Figure 11: Our microhub smoothing technique is highly adaptive: the number of nodes pruned vs. expanded changes dramatically across iterations, but stabilizes in the end.

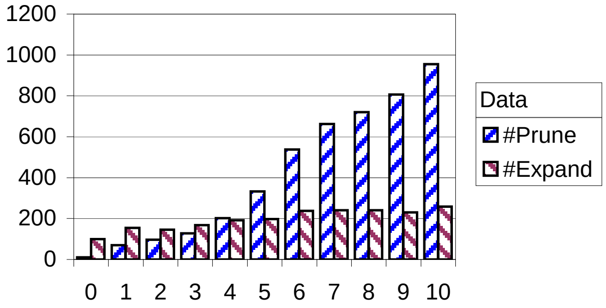


Figure 12: For some queries for which HITS showed high drift, our algorithm continues to expand a relatively larger number of nodes in an attempt to suppress drift.

DMoz and classification: First we trained the RAINBOW classifier on all the topics of DMoz. Next we picked some topics and used the example URLs in DMoz to seed the graph to be analyzed. As in HITS, we also included pages distant from these seed pages by one hyperlink. This graph is subjected to topic distillation. The 40 top authorities from the distillation algorithm (HITS, DOMHITS or DOMTEXTHITS) are submitted to the classifier. For each authority document d (which may not belong to the examples compiled by DMoz), the classifier returns a Bayesian estimate of $\Pr(c|d)$. These are added up as the *expected number of relevant authorities* among the top 40. Figure 10 clearly shows that the topical "purity" is best upheld by DOMTEXTHITS, followed by DOMHITS, HITS being the worst in this regard.

Pruning vs. expansion: In Figures 11 and 12 we plot relative numbers of nodes pruned vs. expanded against the number of iterations. Queries which do not have a tendency to drift look like Figure 11. Initially, both numbers are small. As the system bootstraps into controlled authority diffusion, more candidate hubs are pruned, i.e., accepted in their entirety. Diffused authority scores in turn lead to fewer nodes getting expanded. For queries with a strong tendency to drift (Figure 12), the number of nodes expanded does not drop as low as in low-drift situations. It is significant that the algorithm adapts to low- and high-drift pages automatically. For all the 28 queries, the respective counts stabilize within 10–20 iterations.

5 Summary and ongoing work

We believe that the application of machine learning to DOM structure will be valuable in bridging the gap between a schema-less, chaotic view of the Web and a relatively structured data model. In this paper we have illustrated our claim with two examples. In ongoing work, we are extending the fine-grained crawler to exploit link patterns across pages and longer paths. We are also trying to speed up our fine-grained topic distillation by factoring out site-level templates common to many pages.

References

[1] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, Aug. 1998. Online at `http://www.henzinger.com/monika/mpapers/sigir98_ps.ps`.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://decweb.ethz.ch/WWW7/1921/com1921.htm`.

[3] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *WWW 10*, Hong Kong, May 2001. Online at `http://www10.org/cdrom/papers/489`.

[4] S. Chakrabarti, B. E. Dom, S. Ravi Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *IEEE Computer*, 32(8):60–67, Aug. 1999.

[5] S. Chakrabarti, M. M. Joshi, and V. B. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *SIGIR*, volume 24, New Orleans, USA, Sept. 2001. ACM.

[6] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW*, Hawaii, may 2002. ACM.

[7] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31:1623–1640, 1999. First appeared in the 8th International World Wide Web Conference, Toronto, May 1999. Available online at `http://www8.org/w8-papers/5a-search-query/crawling/index.html`.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[9] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 527–534. Morgan Kaufmann, 2000. Online at `http://www.neci.nec.com/~lawrence/papers/focus-vldb00/focus-vldb00.pdf`.

[10] B. Kahle. Preserving the internet. *Scientific American*, 276(3):82–83, Mar. 1997. Online at `http://www.sciam.com/0397issue/0397kahle.html` and `http://www.alexa.com/~brewster/essays/sciam_article.html`.

[11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[12] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Semi-automated Web taxonomy construction. Research Report, 2002. Online at `http://www.verity.com/pdf/taxonomy.pdf`.

[13] S. Lawrence and C. Lee Giles. Accessibility of information on the Web. *Nature*, 400:107–109, July 1999.

[14] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Software available from `http://www.cs.cmu.edu/~mccallum/bow/`, 1998.

[15] J. Rennie and A. McCallum. Using reinforcement learning to spider the web efficiently. In *ICML*, 1999. Online at `http://www.cs.cmu.edu/~mccallum/papers/rlspider-icml99s.ps.gz`.

[16] J. Rissanen. Stochastic complexity in statistical inquiry. In *World Scientific Series in Computer Science*, volume 15. World Scientific, Singapore, 1989.

[17] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[18] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Online at `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

Query- vs. Crawling-based Classification of Searchable Web Databases

Luis Gravano
gravano@cs.columbia.edu
Columbia University

Panagiotis G. Ipeirotis
pirot@cs.columbia.edu
Columbia University

Mehran Sahami
sahami@epiphany.com
E.piphany Inc.

1 Introduction

The World-Wide Web is one of the main channels through which people currently exchange information. Unfortunately, this information is not characterized in a way that would make its semantics readily understandable by computers, which complicates building value-added services on top of the existing information. An ambitious effort that aims to facilitate the development of such services is the so-called "Semantic Web." According to Berners-Lee et al. [1]:

> "The Semantic Web will bring structure to the meaningful content of web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users."

Classification is a useful way to structure web resources. Directories such as *Yahoo!* manually organize pages in a classification scheme that helps users locate information of interest. In principle, each resource could use the Semantic Web infrastructure to categorize itself in a classification scheme of interest. However, since a single, universal taxonomy is unlikely to fit the needs of all users (and applications), a variety of classification schemes will continue to emerge. With a heterogeneity of classification schemes, the self-characterization of resources might prove to be unwieldy. Hence, tools to automatically categorize web resources remain of key importance.

The web contains pages that both regular search-engine crawlers can reach and retrieve in order to index, and documents that search engines ignore. In particular, "hidden-web" databases contain documents that are only accessible through a search interface. Links to documents in a hidden-web database are typically generated only as a response to a dynamically issued query to the database's search interface. As a result, valuable web-accessible content remains largely ignored by current search engines, as the following example illustrates.

Example 1: Consider the medical bibliographic database CANCERLIT® from the National Cancer Institute's International Cancer Information Center, which makes medical bibliographic information about cancer available through the web [1]. If we query CANCERLIT for documents with the keywords `lung AND cancer`, CANCERLIT returns 67,518 matches, corresponding to high-quality citations to medical articles. The abstracts and citations are stored locally at the CANCERLIT site and are not distributed over the web. Unfortunately, the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] The query interface is available at `http://www.cancer.gov/search/cancer_literature/`.

high-quality contents of CANCERLIT are not "crawlable" by traditional search engines. A query[2] on Google that finds the pages in the CANCERLIT site with the keywords "lung" and "cancer" returns only 186 matches, which do not correspond to CANCERLIT abstracts but rather to other pages in that domain. This illustrates that the valuable content available through CANCERLIT is not indexed by traditional crawlers.

Another scenario in which the contents of a database are not accessible to crawlers is when the database is using a `robots.txt` file to instruct crawlers not to retrieve any files. Technically, in this case a crawler might be able to reach and fetch the pages on the site; however, a crawler implementing proper "netiquette" would not do so. Hence, in this case the database can also be considered uncrawlable. Such is the case with the *PubMed* database [3].

In previous work [8], we proposed a classification method for hidden-web databases. Our technique uses automatically learned queries that are strongly associated with the categories of interest (e.g., the query "lung AND cancer" is associated with the category "Health"). These queries consist of very few keywords, and are adaptively sent to a database that we want to classify. Using only the *number of matches* generated for the queries associated with each category, the algorithm detects the topic distribution in the database and classifies the database accordingly, *without retrieving any documents from the database*. The result of the query probing process is an accurate classification of the database, incurring only a small cost[4].

Our classification method was designed for hidden-web databases, where querying is the only way to access their documents. Interestingly, the same method can be applied to classify any web site that offers a search interface over its pages, whether or not its contents are directly accessible to a crawler. For example, although the news articles at the CNN Sports Illustrated [5] news site are accessible through links, the keyword search interface that is offered can be used alone to classify this site. As an alternative classification approach for such a database, we could use a standard crawler to download all (or a fraction) of the pages at the site, classify the pages using a *document* classifier, and decide how to classify the *site* based on the document class distribution.

The focus of this paper is to compare these two classification approaches, namely the query-based approach that we introduced in [8] and the crawling-based approach outlined above. We present evidence that our query-based approach works best in terms of both classification accuracy and efficiency. In a nutshell, the crawling-based approach can lead to unstable classification decisions, while requiring large amounts of data to be retrieved when classifying large databases. The rest of this paper is organized as follows. Section 2 provides a definition of database classification. Then, Section 3 gives a brief overview of both our query-based algorithm for this task and a crawling-based algorithm. Section 4 reports an experimental comparison of the query- and crawling-based approaches in terms of their accuracy and efficiency. Finally, Section 5 concludes the paper.

2 Classifying Searchable Web Databases

Similar to well-known web directories like *Yahoo!*, other directories *manually* classify hidden-web databases into classification schemes that users can browse to find databases of interest. An example of such a directory is InvisibleWeb [6]. Figure 1 shows a small fraction of InvisibleWeb's classification scheme. The classification of a database into such a scheme is determined by the database contents.

Example 2: Consider topic category "*Basketball.*" *CBS SportsLine* has a large number of articles about basketball and covers not only women's basketball but other basketball leagues as well. It also covers other sports like football, baseball, and hockey. On the other hand, *WNBA* only has articles about women's basketball. The

[2] The query is `lung cancer site:cancer.gov`.

[3] `http://www4.ncbi.nlm.nih.gov/PubMed/`

[4] A prototype implementation is available at `http://qprober.cs.columbia.edu`.

[5] `http://www.cnnsi.com`

[6] `http://www.invisibleweb.com`

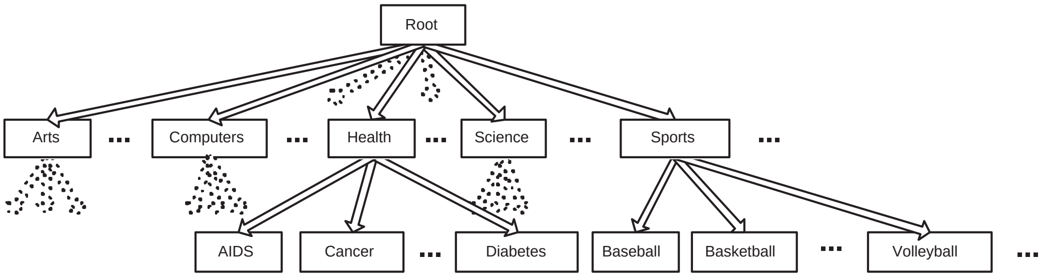


Figure 1: Portion of the InvisibleWeb classification scheme.

way that we will classify these sites depends on the intended usage of the classification. Users who prefer to see *only* articles relevant to basketball might prefer a *specificity-based* classification and would like to have the site *WNBA* classified into node "*Basketball*." However, these users would not want to have *CBS SportsLine* in this node, since this site has a large number of articles not related to basketball.

To define the database categorization problem, we introduce the notion of topic "specificity" of a database. The *Specificity*(D, C_i) of a database D for a given category C_i is the fraction of the documents stored in D that are about C_i [7]. So, a "specificity-based" classification of a database would focus on high-specificity categories. To formalize this notion, we introduce a *specificity threshold* τ_s, with values ranging between 0 and 1. For example, for $\tau_s = 0.25$ we classify a database into a given category only if at least 25% of the database documents are relevant to this category. More generally, we define the classification of a database in a hierarchical classification scheme as follows:

Definition 1: Consider a hierarchical classification scheme C with categories $C_1, \ldots, C_n$, and a searchable web database D. The *ideal classification of D in C* is the set *Ideal(D)* of categories C_i that satisfy the following conditions:

- *Specificity*$(D, C_i) \geq \tau_s$.
- *Specificity*$(D, C_j) \geq \tau_s$ for all ancestors C_j of C_i.
- *Specificity*$(D, C_k) < \tau_s$ for all children C_k of C_i.

where $0 \leq \tau_s \leq 1$ is the given threshold.

Of course, this definition requires a priori knowledge of the distribution of database documents across categories. Unfortunately, this information is rarely available, hence we need a way to estimate it.

3 Query- and Crawling-based Classification Algorithms

We now describe two approaches for classifying searchable web databases, one based on simple queries (Section 3.1) and one based on web crawling (Section 3.2).

[7] In a hierarchical classification scheme, if C_i is not the hierarchy root then this fraction is computed with respect to D's documents in C_i's parent category [8]. Also, please refer to [8] for a complementary notion that focuses only on the absolute number of database documents in each category.

3.1 Query-based Classification

In [8], we presented a query-based algorithm to approximate the *Ideal(D)* categorization of a database D. Our strategy works as follows. First, we train a document classifier over a given classification scheme using machine learning techniques. Second, we construct queries from the document classifier, which we use in turn to probe the database to obtain the category distribution information needed for database categorization. Specifically, we follow the algorithm below:

1. Train a rule-based document classifier with a set of preclassified documents.
2. Transform the learned classifier rules into queries.
3. Adaptively issue these queries to databases, extracting the number of matches for each query.
4. Classify databases using the number of query matches.

Our approach exploits rule-based document classifiers [5] to estimate the number of documents associated with each category. Such a document classifier has rules to assign categories to a document based on the words in the document. For example, the following rules are part of a classifier for the three categories *"Sports,"* *"Health,"* and *"Computers"*:

IF ibm AND computer	THEN Computers
IF jordan AND bulls	THEN Sports
IF cancer	THEN Health

The first rule classifies all documents containing both the words "ibm" and "computer" into the category *"Computers."* Similarly, the third rule classifies all documents containing the word "cancer" into the category *"Health."* Note that all the rules contain conjunctions of words in their antecedents.

To simulate the behavior of a rule-based classifier over all documents of a database, we map each rule of the classifier into a boolean query that is the conjunction of all words in the rule's antecedent. Thus, if we send a query probe to a database, the query will match (a close approximation to) the set of documents in the database that the associated rule would have classified into its corresponding category. For example, we map the rule `IF jordan AND bulls THEN Sports` into the boolean query `jordan AND bulls`. We expect this query to retrieve mostly documents in the *"Sports"* category. Now, instead of retrieving the documents themselves, we just keep the number of matches reported for this query (it is quite common for a database to report the total number of documents matching a query in the results page with a line like "X documents found"), and use this number as a measure of how many documents in the database match the condition of this rule.

The query-probing results provide a good approximation of the document distribution across the categories in a database. Specifically, we can approximate the number of documents in a category as the total number of matches for the query probes associated with the category. Using this information, we can then classify the database according to the specificity threshold τ_s, since we have approximations to all the necessary category-frequency information. For example, in Figure 2 we have listed the detected specificity for the top-level categories of a classification scheme for five searchable web databases. There is a clear peak in the detected specificity for the category where each database would have been classified by a human. Using these numbers, we automatically derive the correct classification of the database. Previous experimental results [8] showed that our method had on average over 75% precision (i.e., on average over 75% of the categories that we produce for a database are correct) and over 75% recall (i.e., on average we discover over 75% of the *Ideal* categories for a database) for a set of 130 real, autonomous web databases. The average number of queries sent to each database was 185, and no documents needed to be retrieved from the databases. Furthermore, the number of words per query ranged between just one and four words. Further details of our algorithm and evaluation are described

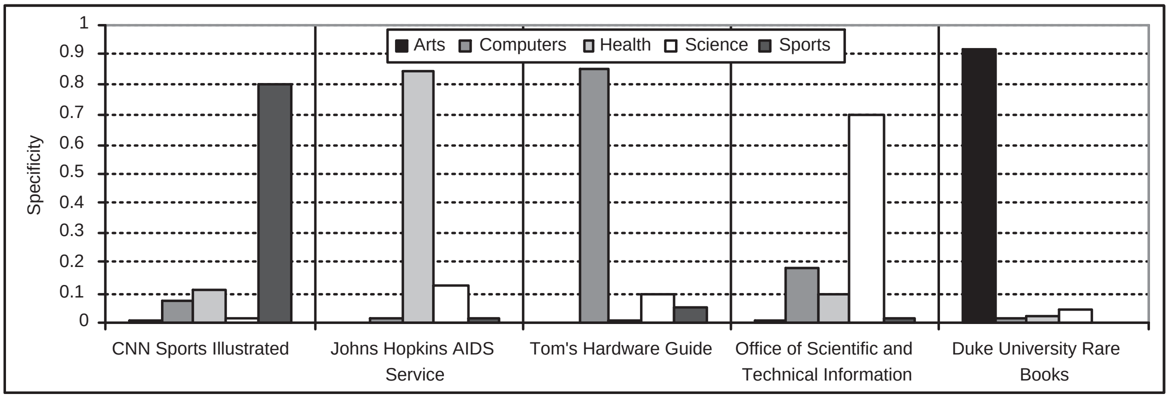


Figure 2: Distribution of documents in the top-level categories for five searchable web databases.

in [8]. (See [2, 3, 10, 6, 7] for other related work relevant to database classification.) As we discussed in the introduction, our technique can be also applied to the classification of any database that offers a search interface for its contents, no matter if its contents are "hidden" or not.

3.2 Crawling-based Classification

As described in Section 2, the categorization of a database is determined by its distribution of documents across categories. We now describe a classification approach that categorizes a crawlable database by simply retrieving its documents using a web crawler and classifying them with a previously-trained document classifier. This algorithm works as follows to classify a web database:

1. Train a rule-based document classifier with a set of preclassified documents.
2. Using a crawler, download all documents from the web database.
3. Classify each retrieved document into a set of categories using the document classifier from Step 1.
4. Classify the database using the number of documents classified into each category from Step 3.

Note that this crawling-based classification approach could be applied both to web sites that are crawlable through "traditional" crawlers (e.g., [4]) and to hidden-web databases as novel crawlers for them are developed [9].

4 Experimental Evaluation

To compare the accuracy and efficiency of the query- and crawling-based classification approaches, we use the five real web databases with crawlable content that are listed in Table 1. For evaluation purposes, we manually classified the five databases by inspecting their contents.

To implement the query-based algorithm of Section 3.1, we constructed a simple "wrapper" around each database. Given a keyword-based query, the wrapper sends it to the database, and extracts *only* the number of matches that the query produces. For each database, we measured the classification accuracy in terms of precision and recall, the amount of information that was transmitted over the network, and the time needed to complete the classification process.

URL	*Brief Description*	*Category*
`http://www.cnnsi.com/`	CNN Sports Illustrated	Sports
`http://www.tomshardware.com/`	Tom's Hardware Guide	Computers
`http://hopkins-aids.edu/`	Johns Hopkins AIDS Service	AIDS
`http://odyssey.lib.duke.edu/`	Duke University Rare Books	Literature
`http://www.osti.gov/`	Office of Scientific and Technical Information	Science

Table 1: The searchable web databases used in the experiments.

To implement the crawling-based algorithm of Section 3.2, we used the GNU Foundation's *wget* tool[8] to crawl and download the contents of each database. We classify each downloaded page *individually* using the same document classifier that the query-based approach needs to generate query probes. Hence at each stage of the crawling process, we know the category distribution of the pages already downloaded, from which we can derive a preliminary classification of the database. We measured the classification accuracy in terms of precision and recall at different crawling stages, to examine the speed with which the crawling-based approach reached the correct classification decision. Additionally, we measured the amount of information that was transmitted over the network, and the time needed to complete the classification process.

Table 2 lists the evaluation results for the two approaches over the databases in our data set. Except for one case, the time needed to complete the classification process and the amount of information transmitted over the network was orders of magnitude larger for the crawling-based approach. Additionally, the crawling-based approach needed extra local processing power to locally classify the documents. When the number of retrieved pages is large, this can create a significant local overhead.

One significant advantage of the query-based approach is the fact that its execution time is largely independent of the size of the database, so this approach can scale to databases of virtually any size. Additionally, no documents are retrieved during querying, which speeds up the classification algorithm and minimizes the required bandwidth for the classification. Finally, the query-based approach gives a better bound on completion time, since the maximum number of queries sent to a database depends only on the given classification scheme and is usually small. In contrast, a crawler might spend a lot of time crawling a large site in order to determine a final classification.

One question that remains to be answered is whether the crawling-based approach can be improved by requiring only a small portion of a site to be downloaded. To understand how fast the crawling approach can reach the correct classification, we measured the precision and recall of the classification results when different fractions of the database are crawled and classified. Our experiments reveal that often a significant fraction of a web database may need to be retrieved before a correct classification decision can be made. For example, as can be inferred from Table 3, the crawler started crawling parts of the CNN.SI that were about specific sports (cycling in this case). Consequently, early in the crawling process, the category distribution was wrongly biased towards this sport and the site was incorrectly classified under this category, rather than being classified under the

	Crawling-based Classification			*Query-based Classification*		
Database	*Time*	*Files*	*Size*	*Time*	*Queries*	*Size*
CNN Sports Illustrated	1325 min	270,202	8 Gb	2 min (-99.8%)	112	357 Kb (-99.9%)
Tom's Hardware Guide	32 min	2,928	105 Mb	3 min (-90.6%)	292	602 Kb (-99.7%)
Johns Hopkins AIDS Service	13 min	1,823	17 Mb	1 min (-92.3%)	314	723 Kb (-95.7%)
Duke University Rare Books	2 min	3,242	16.5 Mb	3 min (+50.0%)	397	1012 Kb (-93.8%)
Office of Scientific and Technical Information	210 min	30,749	416 Mb	2 min (-99.0%)	174	423 Kb (-99.8%)

Table 2: The performance of crawling- and query-based classification for five databases.

[8] `http://www.gnu.org/software/wget/wget.html`

% Crawled	**10%**	**30%**	**50%**	**60%**	**70%**	**100%**
CNN Sports Illustrated	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling $P = 1.0$ $R = 0.09$	**Sports** $P = 1.0$ $R = 1.0$	**Sports** $P = 1.0$ $R = 1.0$
Tom's Hardware Guide	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$
Johns Hopkins AIDS Service	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$
Duke University Rare Books	Poetry Texts Classics History Photography $P = 0.6$ $R = 0.6$	Poetry $P = 1.0$ $R = 0.2$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$
Office of Scientific and Technical Information	Biology $P = 1.0$ $R = 0.33$	Root $P = 0.25$ $R = 1.0$	Root $P = 0.25$ $R = 1.0$	**Biology** $P = 1.0$ $R = 0.33$	**Biology** $P = 1.0$ $R = 0.33$	**Biology** $P = 1.0$ $R = 0.33$

Table 3: The crawling-based classification and associated precision (P) and recall (R) for five databases after crawling different fractions of each database (specificity threshold $\tau_s = 0.4$).

more general "Sports" category. Only after crawling 70% of all pages did this approach yield the right category for this database. A similar observation holds for *Duke's Rare Book Collection* and the *Office of Scientific and Technical Information*. On the other hand, the classification of *Tom's Hardware Guide* and the *Johns Hopkins AIDS Service* was remarkably accurate in the crawling process and converged to the correct result very fast. This happened because these two sites contain documents that are relatively homogeneous in topic. Hence, even the first few pages retrieved were good representatives of the database as a whole.

In summary, the crawling-based approach is prone to producing wrong classification decisions at early stages of the crawling. A crawling-based approach could produce reasonable classification results early on only if crawling could somehow guarantee that the documents crawled first reflected the real topic distribution in the entire database. However, currently no crawler can perform such a traversal and it is doubtful that any will ever be able to do so, since to build such a crawler presupposes knowledge of the distribution of documents at the sites. In contrast, our query-based method manages to detect the correct classification of all these databases using only a fraction of the time and data required for the crawling-based approach. Hence, these first experimental results suggest that the query-based approach is a better alternative for the classification of web databases, both in terms of classification accuracy and efficiency.

5 Conclusion

In this paper we have summarized a query-based classification for searchable web databases [8]. This algorithm is the state-of-the-art for the classification of databases with "uncrawlable" content. Interestingly, the algorithm can also be used for crawlable databases, as long as they expose a search interface. In such cases, we could alternatively classify the databases by crawling their contents and classifying the retrieved documents. We have argued in this paper that our query-based approach significantly outperforms the crawling-based approach. The query-based approach is in some cases orders of magnitude more efficient than its crawling-based counterpart both in execution time and in utilization of network resources. Additionally, the query-based algorithm reaches the correct classification decision quickly, while the crawling-based approach depends greatly on how well the crawling order reflects the actual topic distribution in the database. Please refer to `http://qprober.cs.columbia.edu` for more details about our query-based algorithm, as well as to access a prototype system demonstrating our approach.

Acknowledgments

Luis Gravano and Panagiotis G. Ipeirotis were funded in part by the National Science Foundation (NSF) under Grants No. IIS-97-33880 and IIS-98-17434.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[2] James P. Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 479–490, 1999.

[3] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[4] Junghoo Cho, Héctor García-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, 1998.

[5] William W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 115–123, 1995.

[6] Ron Dolin, Divyakant Agrawal, and Amr El Abbadi. Scalable collection summarization and selection. In *Proceedings of the Fourth ACM International Conference on Digital Libraries (DL'99)*, pages 49–58, 1999.

[7] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion*: Intelligent fusion from multiple, distributed search engines. *The Journal of Universal Computer Science*, 2(9):637–649, September 1996.

[8] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, pages 67–78, 2001.

[9] Sriram Raghavan and Héctor García-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB 2001)*, 2001.

[10] Wenxian Wang, Weiyi Meng, and Clement Yu. Concept hierarchy based text database categorization in a metasearch engine environment. In *Proceedings of the First International Conference on Web Information Systems Engineering (WISE'2000)*, pages 283–290, 2000.

Classification and Intelligent Search on Information in XML

Norbert Fuhr
University of Dortmund, Germany

Gerhard Weikum
University of the Saarland, Germany

1 Introduction

XML will be the method of choice for representing all kinds of documents in product catalogs, digital libraries, scientific data repositories, and across the Web. This observation creates high expectations that XML will be a major catalyst in constructing the "Semantic Web". However, merely casting all documents into XML format does not necessarily make a document's semantics explicit and more amenable for effective information searching. Rather, to fully leverage XML on a global scale, significant progress is needed on the following issues:

1. providing an easy-to-use yet powerful and efficient search language that combines concepts from current XML pattern-matching languages (e.g., XPath, XQuery, etc.) with ontology-backed information-retrieval-style search result ranking,

2. extracting more semantics from existing document collections by constructing structural and ontological skeletons (e.g., in the form of DTDs or XML schemas) that describe the data at a higher semantic level and can also facilitate new forms of indexing for efficiency, and

3. classifying existing documents according to a given thematic or personalized, hierarchical ontology to make searching more effective (e.g., exploit relevance feedback) and efficient (e.g., limit the search focus).

CLASSIX, a joint project of the Universities of Dortmund and the Saarland in Germany, addresses these three issues. We describe our approaches for each of these topics in the remainder of this paper.

2 Query languages for information retrieval of XML documents

Looking at the broad variety of XML applications and systems that are currently under development, one can see that there are in fact two different views on XML:

- The *document-centric view* focuses on structured documents in the traditional sense (based on concepts from electronic publishing, especially SGML). Here XML is used for logical markup of texts both at the macro level (e.g. chapter, section, paragraph) and the micro level (e.g. MathML for mathematical formulas, CML for chemical formulas).

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

- The *data-centric view* uses XML for exchanging formatted data in a generic, serialized form between different applications (e.g. spreadsheets, database records). This is especially important for e-business applications (e.g. for exchanging orders, bills).

Unfortunately, most approaches for XML query languages (including the W3C working group on XML query languages with its proposal of XQuery) have concentrated on the data-centric view, thus providing little support for information retrieval of XML documents.

In contrast, our work focuses on the document-centric view. For information retrieval of XML documents, we take into account the intrinsic imprecision and vagueness of IR and the resulting need for ranked lists as search results (as opposed to result sets or bags in traditional database querying). For this purpose, we are developing the query language XIRQL (Xml IR Query Language), which extends the XPath part of the (proposed standard) query language XQuery by the following features:

- weighting and ranking,
- relevance-oriented search,
- data types with vague predicates,
- structural relativism.

Below, we describe each of these concepts in detail.

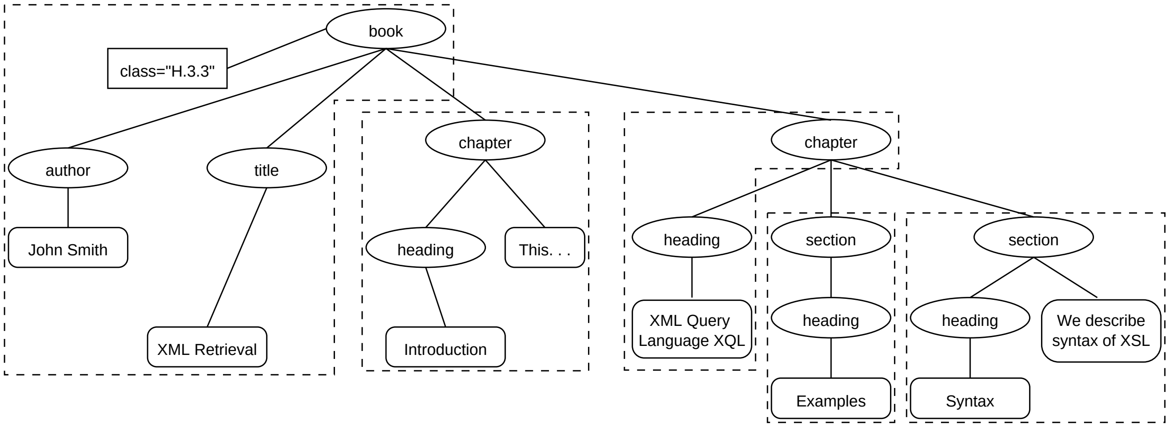


Figure 1: Example XML document tree

Weighting and ranking. IR research has shown that document term weighting as well as query term weighting are necessary tools for effective retrieval in textual documents. So query conditions referring to the text of elements should consider index term weights. Furthermore, query term weighting should also be possible, by introducing a weighted sum operator (e.g. 0.6 · ”XML” + 0.4 · ”retrieval”). These weights should be used for computing an overall retrieval score for the elements retrieved, thus resulting in a ranked list of elements.

The basic idea for assigning indexing weights to document terms is that the weight of a term depends on its context. So we split up a document into disjoint contexts which we call index nodes; based on the DTD, index nodes are specified by giving the names of those elements that form the roots of important and “semantically coherent” subtrees of XML documents. Figure 1 shows an example where index nodes are marked as dashed boxes. For each term in such a context, the indexing weight is computed by using standard weighting functions like e.g. tf·idf.

For retrieval, we use a probabilistic retrieval model where we treat all term occurrences within the same index node as a single probabilistic event, whereas occurrences of other terms in the same node or of the same term in other nodes are regarded as being independent events. When a query condition matches a term occurrence within an index node, the event representing the corresponding term-node pair (along with its weight) is returned as answer to this condition.

As an example, for the document in Figure 1, a query searching for the word 'syntax' occurring in a heading would match the rightmost heading element. Since the word 'syntax' also occurs in the body of the corresponding section, which is part of the same index node, these additional occurrences affect the weight of the term in the index node, and thus also the weight of the match.

By using the concept of event keys and event expressions from [9], we can compute result probabilities in a consistent way, even for complex combinations of query conditions. Due to the (assumed) independence of events, our approach always yields point probabilities, thus yielding a linear ranking of results.

Relevance-oriented search. The query language should also support traditional IR queries, where only the requested content is specified, but not the type of elements to be retrieved. In this case, the IR system should be able to retrieve the most relevant elements; following [4], we assume that this should be the most specific element(s) that satisfy the query. In the presence of weighted index terms, the tradeoff between these weights and the specifity of an answer has to be considered, e.g. by an appropriate weighting scheme.

For this purpose, we introduce the concept of augmentation. The index weights of the most specific index nodes are given directly. For retrieval of the higher-level objects, we have to combine the weights of the different text units contained. When propagating indexing weights to the higher-level objects, they are downweighted (multiplied by an augmentation weight), such that, in general, more specific results get higher retrieval weights.

In addition, since not all elements of a document may be reasonable answers for relevance-oriented queries, we restrict the set of possible answers to the roots of index nodes. For example, consider the relevance-oriented query 'syntax $\wedge$ example'. In the document shown in Figure 1, there is no single index node matching this query; however, the rightmost chapter satisfies all conditions, when we propagate the weights of the two query terms up to this level. In contrast, a query for 'XSL' would yield the highest weight for the last section, whereas the comprising chapter would be returned with a lower weight.

Data types and vague predicates. The standard IR approach for weighting supports vague searches on plain text only. XML allows for a fine grained markup of elements, and thus, there should be the possibility to use special search predicates for different types of elements. For example, for an element containing person names, similarity search for proper names should be offered; in technical documents, elements containing measurement values should be searchable by means of the comparison predicates $>$ and $<$ operating on floating point numbers. Thus, there should be the possibility of having elements of different data types, where each data type comes with a set of specific search predicates. In order to support the intrinsic vagueness of IR, most of these predicates should be vague (e.g. search for measurements that were taken at about 20°C).

Whereas XML schema focuses on those properties of data types that can be checked at the syntactic level (i.e. structure and domain of possible values), we assume that the difference between many data types is visible at the semantic level only; thus, we characterize data types by their sets of vague predicates (such as phonetic similarity of names, English vs. French stemming). In principle, data types with vague predicates generalize text indexing methods for all kinds of data. Thus, the consideration regarding the probabilistic interpretation of weights apply here as well.

Structural relativism. Database-oriented XML query languages are closely tied to the structure of XML documents, but it is possible to use syntactically different XML variants to express the same meaning. For example, a particular information could be encoded as an XML attribute or as an XML element. As another example, a user may wish to search for a value of a specific datatype in a document (e.g. a person name), without bothering about the element names.

For this purpose, we support query conditions that ignore the difference between elements and attributes of a specific name. Based on our notion of datatypes, we allow for searches covering all elements of a specific datatypes. We are investigating further possibilities of generalizations (e.g. by introducing hierarchies over elements, similar to `rdfs:subPropertyOf` in RDF schema).

Implementation. As a first prototype, we have implemented the retrieval engine named HyREX[1]. This system accepts XIRQL queries, translates them into an internal path algebra [7] [8] and processes these expressions based on inverted files; the current version performs efficient retrieval for document collections up to the gigabyte range.

3 Metadata

Metadata generally adds to the value of raw XML data by making it easier to interpret the data semantics and reason about relationships within the data. In the CLASSIX project we are pursuing a pragmatic approach toward constructing personal or domain-specific ontologies as a metadata backbone that can guide the user and an XML search engine in refining queries. In contrast to a pure logical representation of ontological relationships, we aim to quantify the similarity of related "concepts" as a basis for computing similarity scores between XML (sub-)documents and queries. This way we can capture, for example, that synonyms are closer to each other than hyper- and hyponyms.

Ontology construction. There is a trend toward organizing XML documents according to domain-specific ontologies with meaningful, carefully chosen element names. However, it cannot be expected that all documents for a given domain will eventually correspond to a standardized XML schema; rather we still expect substantial schematic variety, for three reasons: 1) not every author of a (high-quality) document is willing to adopt standardized terminology, 2) ontologies are evolving over time, and 3) there is some "natural" diversity of ontologies for the same or overlapping domains.

For these reasons, we treat ontogical metadata as an optional tool for searching XML data, rather than firmly relying on its existence and unambiguity. In the CLASSIX project, we are building application-specific ontology indexes as follows. When a crawler traverses XML documents from the Web or an intranet, all element names that are not yet in the application's ontology index are added as nodes to a graph. The edges between nodes represent semantic relationships; currently we distinguish only between synonym and (different kinds of) hypernym/hyponym edges. The position of a new element name in the index graph is determined by calling WordNet [6], a comprehensive thesaurus put together by cognitive scientists. We extract the concept description (i.e., the "word sense" in WordNet terminology), all synonyms, and all hypernyms and hyponyms for the given word. In addition to WordNet, we are also considering other sources of authoritative ontological information such as XML-based directories or lexicons for the domain of interest. When the crawler encounters a document collection whose entire data corresponds to the same DTD or XML schema, this step needs to be performed only once.

After an element name has been added to the index graph and its edges to and from already existing nodes have been constructed, the second step is to quantify the new relationships by computing a similarity weight between adjacent nodes. Here again, we are currently pursuing a fairly pragmatic approach, but plan to investigate better theoretical foundations as well. The current approach is based on extracting co-occurence frequencies from various sources like Google, Yahoo, and other Web search engines and directories. The similarity weight

[1] `http://ls6-www.informatik.uni-dortmund.de/ir/projects/hyrex/`

between two nodes increases in proportion to the frequency of finding two corresponding words (or, more precisely, synonym sets) in the same document.

Query refinement. The ontology index is exploited to expand or narrow the search scope of a user query, automatically or in combination with a user feedback step. This is important in two cases: when the original query is "overspecified" using overly specific search conditions that would lead to very few results or even no matches at all, or when the query is "underspecified" using overly broad or even ambiguous search terms and conditions. In the first case we would consider generating a refined query with additional search conditions based on hypernyms and their synonyms; in the second case we would add selected hyponyms (and possibly more related terms when useful for disambiguation). An additional option to consider would be the re-adjustment of term weights in the query.

As an example, consider a query with a similarity search condition $\approx$*region* $\sim\sim$ *"India"*. Here *region* is an element name that needs to be matched, with the additional condition that the element content contains the term *"India"*. The unary similarity operator $\sim$ denotes that the element name does not need to occur literally but should rather be matched "semantically", and the binary similarity operator $\sim\sim$ denotes an analogous approximation for finding the specified term in the element content. To allow for approximate matches, the original element name *region* could be expanded into the disjunction *region* | *country* | *continent* and the search for *"India"* should consider also related terms such as *"Asia"*, *"Bangladesh"*, *"Tamil Nadu"*, etc., provided that the similarity scores obtained from the ontology index are above some threshold. The ranking of the search results reflects the similarities between the terms in the query and the found documents. For example, a document with an element name *continent* and the word *"Asia"* in its content will be ranked lower than a document with an element named *region* and containing the term *"India"*.

Implementation. The XXL prototype system (fle*x*ible *X*ML search *l*anguage), an XML crawler and search engine, includes basic support for ontology-based similarity search [15, 16]. The system constructs an ontology index from XML element names seen during a crawl by looking up WordNet entries as sketched above, and expands queries with synonyms and hyper-/hyponyms. The ranking of search results reflects the corresponding proximity measures. Preliminary experiments indicate that this approach can significantly improve the precision and recall of searching XML document collections [15].

4 Automatic Classification

XML documents can be searched more effectively and efficiently when they are organized in a more explicit way. For this purpose, we are investigating how we can automatically classify newly crawled documents into a hierarchical taxonomy. In other words, we not only build ontologies at the metadata level, but want to populate the ontological concepts or categories of interest with actual data. For simplicity, we so far concentrate on trees of topics that reflect the user's or user community's interest profile. Typically, such an application-specific taxonomy would be an order of magnitude smaller than say the complete directory structure of a Yahoo-like service. Each node in the tree would be a priori populated with a number of prototypical documents derived from user bookmarks, surf trails, and other user profiling information. These documents serve as training data for the classifier.

Rather than viewing crawling and classification as two separate stages of a data organization engine, we interleave these two steps following a paradigm known as *focused crawling* [3]. A focused crawl starts from the training documents of the taxonomy's categories as seeds and then traverses a, hopefully small, fraction of the Web with focus on these topics of interest. The classification is invoked for each visited document, which is either added to one or more categories when the classification test is positive or discarded in the negative case.

The accuracy of the classification step depends on three key aspects:

- the mathematical model and algorithm that are used for classification,
- the feature set upon which the classifier makes its decisions, and
- the quality of the training data that is initially categorized by human users and from which the classifier derives parameters for its decision model.

These three issues are discussed next.

Supervised learning for hierarchical taxonomies. The most natural way of classifying a new document is a top-down procedure starting from the root of the taxonomy [5]. For each category that is considered a classifier is invoked and returns a yes-or-no decision and possibly a confidence measure of the decision. When the documents fits with more than one category, either the one with the highest confidence measure is chosen or the document is classified into multiple categories. Then the classification proceeds with the children of the categories to which the document has been added.

This procedure requires a binary classifier for each category, a test that decides whether a document belongs to the category or not. To this end we are using a supervised learning method known as support vector machines (SVM) [2]. Unlike simpler classifiers such as Naive Bayes, this technique takes all correlations in the underlying feature space (i.e., the term vector space of the documents) into account, by learning an optimal hyperplane that separates positive from negative training documents in the feature space (possibly with some outliers on the “wrong” side of the hyperplane). The decision procedure is a very efficient scalar product computation, testing on which side of the hyperplane a document lies, and the distance from the hpyerplane can be interpreted as a confidence measure for the classification of a document.

An SVM classifier needs both positive and negative training data for each topic. So far we treat all positive training data from a given category’s siblings in the taxonomy (i.e., the competing categories given that a document has already been classified into the parent category) as negative training data for the given category. This seemingly natural approach can be expected to work well for a “closed-world” document collection such as the popular Reuters collection in the TREC benchmark, where each document that will ever be passed to the classifier is known to belong to at least one of the categories. With Web data or documents from a highly diverse intranet, however, this assumption is not valid. We are currently investigating to what extent additional, explicit negative training data should be incorporated and how this can be done without imposing too much of a burden on the human user.

Feature selection. For efficiency as well as accuracy typically only a subset of the documents’ terms are used as components of the feature space on which the classifiers are based [10, 17]. A good feature discriminates competing topics (i.e., siblings in the taxonomy tree) from each other. Therefore, feature selection has to be topic-specific; it needs to be performed for each topic in the tree individually. As an example, consider a taxonomy with topics mathematics, agriculture, and arts, where mathematics has subcategories algebra and stochastics. Obviously, the term “theorem” is very characteristic for math documents and thus an excellent discriminator between mathematics, agriculture, and arts. However, it is of no use at all to discriminate algebra versus stochastics. A terms such as “field”, on the other hand, is a good indicator for the topic algebra when the only competing topic is stochastics; however, it is useless for a classifier that test mathematics versus agriculture.

To compute the best features for a topic-specific classifier we are using information-theoretic measures, specifically, the cross-entropy between feature frequencies and topics (a special case of the Kullback-Leibler divergence, which measures the differences between multivariate probability distributions, the joint distribution of features and classes versus independent distributions in our setting) [12]. As feature candidates we are studying individual terms as well as term pairs that occur in the same XML element, XML element names, and also pairs of parent-child element names. A widely open issue in this context is to what extent additional, external sources can provide good features; for example, the neighbors of a hyperlinked document, the DTD of an XML document, or a thesaurus for a given category may yield features that do not occur in the training documents themselves.

Online training. An inherent difficulty in automatic classification is the scarceness of good training data. Motivated by some initial work on using automatically classified documents as additional training data [13], we are pursuing an approach with continuous online training. For this purpose, a set of the most characteristic documents of a topic, coined *archetypes*, are determined in two, complementary ways. First, a link analysis procedure, using a variant of Kleinberg's HITS algorithm [11, 1], is initiated. This procedure yields a ranking of *authorities* for each topic, documents that contain high quality information relevant to the topic. The second source of topic-specific archetypes builds on the confidence of the classifier's yes-or-no decision for a given node of the taxonomy tree. Among the automatically classified documents of a topic those documents whose yes decision had the highest confidence measures are selected as archetypes. Archetypes and the original training documents are then fed into the classifier for online re-training.

Implementation. The above considerations have been implemented in a prototype focused crawler coined BINGO! (for bookmark-induced gathering of information) [14]. BINGO! is completely implemented in Java and uses Oracle8i as an underlying storage engine. We are currently working on a first round of large-scale, long-running experiments to evaluate the viability and benefits of our approach.

5 Conclusion

This paper has discussed several key issues in exploiting the potential role of XML as a catalyst toward a "Semantic Web". We are investigating each of these issues, aiming to understand how they can contribute to better organization and more effective search of data. In addition, we are studying the interplay of these various aspects, striving for synergies among IR and DB querying concepts, ontological metadata, and machine learning techniques for automatic classification.

References

[1] K. Bharat, M. Henzinger: Improved Algorithms for Topic Distillation in a Hyperlinked Environment, ACM SIGIR Conference, 1998.

[2] C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery Vol.2 No.2, 1998.

[3] S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, WWW Conference, 1999.

[4] Y. Chiaramella, P.Mulhem, F.Fourel: A Model for Multimedia Information Retrieval, FERMI ESPRIT BRA 8134, University of Glasgow, 1996.

[5] S. Dumais, H. Chen: Hierarchical Classification of Web Content, ACM SIGIR Conference, 2000.

[6] C. Fellbaum (Editor): WordNet: An Electronic Lexical Database, MIT Press, 1998.

[7] N.Fuhr, K.Großjohann: XIRQL: A Query Language for Information Retrieval in XML Documents, Proceedings ACM-SIGIR Conference. pp.172-180, 2001.

[8] N.Fuhr, K.Großjohann: XIRQL: An XML Query Language Based on Information Retrieval Concepts. (Submitted for publication) `http://ls6-www.informatik.uni-dortmund.de/ir/publications/2002/Fuhr_Grossjohann:02.html`

[9] N. Fuhr, T. Rölleke: A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems, ACM Transactions on Information Systems, Vol.14 No.1, 1997.

[10] D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, International Conference on Machine Learning (ICML), 1997.

[11] J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM Vol.46 No.5, 1999.

[12] C.D. Manning, H. Schuetze: Foundations of Statistical Natural Language Processing, MIT Press, 1999.

[13] K. Nigam, A. McCallum, S. Thrun, T. Mitchell: Text Classification from Labeled and Unlabeled Documents Using EM, Machine Intelligence Vol.39 No.2/3, 2000.

[14] S. Sizov, S. Siersdorfer, M. Theobald, G. Weikum: The BINGO! Focused Crawler: From Bookmarks to Archetypes, Demo Paper, International Conference on Data Engineering (ICDE), San Jose, 2002.

[15] A. Theobald, G. Weikum: Adding Relevance to XML, 3rd International Workshop on the Web and Databases (WebDB), Dallas, 2000, LNCS 1997, Springer, 2001.

[16] A. Theobald, G. Weikum: The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking, 8th International Conference on Extending Database Technology (EDBT), Prague, 2002.

[17] Y. Yang, J.O. Pedersen: A Comparative Study on Feature Selection in Text Categorization, International Conference on Machine Learning (ICML), 1997.

DATA ENGINEERING

CALL FOR PAPERS

19th International Conference on Data Engineering
Sponsored by the IEEE Computer Society
March 5 - March 8, 2003
Bangalore, India

http://www.aztecsoft.com/icde2003

The ICDE 2003 International Conference on Data Engineering will be held in Bangalore, India, the center of Indian software activity with a variety of historical and natural attractions within striking distance.

ICDE 2003 aims at providing a premier forum for

- presenting new research results
- exposing practicing engineers to evolving research, tools, and practice and providing them with an early opportunity to evaluate these
- exposing the research community to the problems of practical applications of data engineering
- promoting the exchange of data engineering technologies and experience among researchers and practicing engineers

ICDE 2003 invites research submissions on all topics related to data engineering, including but not limited to those listed below.

Topic	**(Vice-chair)**
Indexing, access methods, data structures	(*David Lomet*, Microsoft)
Query/trans. processing & optimization	(*Don Kossmann*, Tech. Univ. Munich)
Data warehousing & OLAP	(*Divy Agrawal*, UC, Santa Barbara)
Mining (Data, Text, & Web)	(*Soumen Chakrabarti*, IIT Bombay)
Semi-structured Data, Metadata & XML	(*Jerome Simeon*, Bell Labs)
WWW & Databases	(*Anindya Datta*, Chutney Tech.)
Middleware, Workflow & Security	(*Gustavo Alonso*, ETH Zurich)
Database engines	(*K.Y.Whang*, KAIST, Korea)
Database applications & experiences	(*Laura Haas*, IBM)
Distributed, Parallel, Mobile Databases	(*Tamer Ozsu*, Univ. Waterloo)
Temporal, Spatial, Scientific, Statistical, Biological Databases	(*Gultekin Ozsoyoglu*, Case Western Res. Univ)

Conference Officers

General Co-chairs	*Deepak Phatak*, IIT Bombay
	Marek Rusinkiewicz, Telcordia
Program Co-chairs	*Umesh Dayal*, HP Laboratories
	Krithi Ramamritham, IIT Bombay
Organizational Chair	*Anand Deshpande*, Persistent Systems, Pune
Steering Committee Liaison	*Erich Neuhold*, GMD-IPSI, Darmstadt
Local Arrangements	*V. R. Govindarajan*, Aztec Software, Bangalore
	Jayant Haritsa, IISc. Bangalore
Publicity Chairs	*Fabio Casati*, HP Laboratories
	Kamal Karlapalem, IIIT, Hyderabad
	Kam-Yiu Lam, City Univ. HK

***RIDE 2003* -- *on* Multi-Lingual Information Management**
-- March 10-11, Hyderabad, India!

For details, please visit the conference website
http://www.aztecsoft.com/icde2003

All Submissions

Due	**June 14, 2002**
Decision notification	**Sept. 10, 2002**
Camera-ready versions due	**Oct. 10, 2002**

Only research paper submissions must be done via the conference website. (The rest should be sent to the appropriate chair(s), see website for details.) Research paper length must not exceed 25 double spaced pages (including figures, tables, etc.), must be in a font equal to or greater than 11 pt, and in pdf.

Publication and Publication Awards
All accepted papers will appear in the Proceedings published by the IEEE Computer Society. Authors of selected papers will be invited to submit extended versions for publication in a special section of the IEEE Transactions of Knowledge and Data Engineering.

An award will be given to the best paper. A separate award honoring K.S. Fu will be given to the best student paper. Papers eligible for this award must have a (graduate or undergraduate) student listed as the first and contact author, and the majority of the authors must be students. Such submissions must be marked as student papers at the time of submission.

Industrial Program The conference program will include a number of papers and invited presentations devoted to industrial developments. Papers intended for this program should be clearly marked as industrial track papers.
Chairs *C. Mohan*, IBM Almaden Research Center
N. Ramani, HP Labs (India)

Panels Panel proposals must include, an abstract, an outline of the panel format, and relevant information about the proposed panelists.
Chairs *Sham Navathe*, Georgia Tech.
Hongjun Lu, HKUST, Hong Kong
Gerhard Weikum, Univ. Saarbruecken

Advanced Technology Seminars Seminar proposals must include an abstract, an outline, a description of the target audience, duration (1.5 or 3 hours) and a short bio of the presenter(s).
Chairs *Alex Buchmann*, Tech. Univ. Darmstadt
S. Sudarshan, IIT Bombay
Vijay Kumar, Univ. Missouri, Kansas City

The deadline for industrial program papers and proposals for panels & advanced technology seminars is the same as for research paper submissions. Submission guidelines will be posted on the conference web site.

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903