Query Language for Structural Retrieval of Deep Web Information

Stefan Müller[1], Ralf-Dieter Schimkat[1], and Rudolf Müller[2]

[1] University of Tübingen, WSI for Computer Science, Sand 13, D-72076 Tübingen, Germany
[2] University of Maastricht, Department of Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands

Abstract. Information provided on the Web is often hidden in databases and dynamically extracted by script pages according to some user input. Web information systems follow this concept when they must handle a huge amount of fast changing data that is somehow related (e.g. shoppers, route planers, yellow pages). In contrast to static pages it is very hard for traditional search engines to properly index pages with non-static content in a way that Web users can perform a precise search. The information that is provided through dynamic script pages is often called the Hidden or Deep Web. We present a retrieval approach that uses the structure of the database schemas and further schema information to calculate a similarity between a structured query and registered Deep Web information systems. Our retrieval process is a combination of structured and keyword-based retrieval. This combination should overcome the drawback of a solely keyword-based indexing method which is insufficient to exactly describe the content of the Deep Web. Dynamically and individually adjustable retrieval behavior further improves the useability of our approach.

1 Introduction

The World-Wide-Web (WWW) as a global repository for information is a big improvement for the quality of life of people around the world: Organizing a journey, buying or selling goods, or just searching detailed and up-to-date information becomes cheaper, faster, and easier. The main problem is to find suitable pages to perform the individual tasks. The research efforts taken to solve this problem brought up a lot of different technologies, e.g. to better describe the content of Web pages (RDF [15]), the Web services (UDDI [14]), and elaborated index and search strategies (pagerank used by Google [13]).

The WWW contains a lot of pages that are not explicitly made persistent in a Web-accessible manner. These pages are dynamically constructed by scripts or servlets according to some user input, e.g. travel destinations, or product lists of Web shops. The set of those pages is called the Deep Web. Information providers use dynamic pages when they have to handle a mass of related information that is rapidly changing over time. Describing this information in a way that traditional search engines can present them to an interested user is only possible, if the Uniform Resource Locator (URL)

is used to transmit user input to the Web site. Search engines feeded with keywords like "flight", "from", "Frankfurt", "to", "Hawaii" would probably not deliver a link to a travel agency, because "Frankfurt" and "Hawaii" is an information of the Deep Web and therefore not available for the index generation of search engines. A keyword description of script pages (the entry points to the Deep Web) can only describe very vague the Deep Web information and the real content of a Web site. Additionally the main part of the information provided by keywords relies on the relations between them and are not expressed. A search engine for Deep Web information should use this structural information, too.

It is interesting to see that the structural information (lost when a user's question is transformed to a set of keywords) is actually used to manage the information of the Deep Web: Most of the sites offering dynamic Web pages store their data within relational databases for which Entity-Relationship models (ER models [4]) accurately describe the content and relations. ER models are abstract enough to deliver information about the stored data without using the data itself. Imagine a search engine that indexes the internal ER models of Deep Web information systems and provides a front end to formulate queries in a structural manner like ER modelling. This paper presents a realization of this approach based on a framework for general model retrieval [12]. The paper is structured as follows: We briefly describe the components of the framework and their application in the domain of ER models in the next section. Section 3 evaluates the structural retrieval approach in a class room experiment. Section 4 shows how the structural query approach can be combined with the necessary keyword-based retrieval. Dynamic control of retrieval behavior is explained in section 5. Before we summarize and conclude this paper, section 6 gives an overview of other approaches to search for Deep Web information.

2 Structural Retrieval of ER models

The management of complex models like ER models is a very popular scientific field. Mapping, matching, or merging models is required in many situations, e.g. when companies want to share data contained in different kind of databases or documents. There is a need of more abstract data descriptions and operations to automatically perform the given tasks. Bernstein et al. described in [1] an algebra on a generalized view of complex models. Matching of models with respect to their inherent structure is one of the issues that have to be tackled.

Firestorm [10] is a framework for the retrieval of models according to their implicit graph structure. The details are presented in [11] and [12]. The core of the framework is a representation of models through three-layer graphs called Structured Service Models (SSM). This representation is based on the concepts of Structured Modelling [6], which is used to formulate mathematical

models. Three-layer graphs allow the design of algorithms that compute the graph similarity between different SSMs. After stating a query in a graphical way the system returns an ordered list of the models that are most similar to the query with respect to the graph structures.

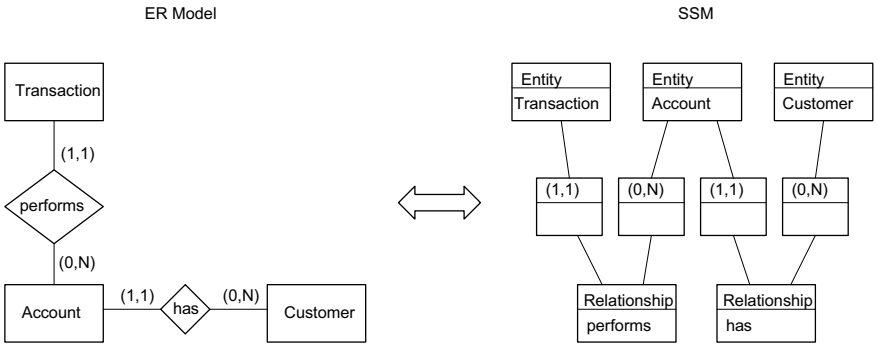

Fig. 1. Mapping between ER model and SSM

We extended the framework to the field of ER models by the definition of a one-to-one mapping between ER models and SSMs following the rules of [12]: Different kind of entities become different kind of nodes of the first layer. Relationships become nodes of the third layer and the relations between entities and dependencies between entities and relationships become nodes on the second layer with types derived from the cardinalities of the relations. Figure 1 shows a simple ER model and its corresponding SSM. The mapping rules enforce that SSMs always decompose into two bipartite graphs. Apart of the mappings we implemented graphical user interfaces used to state queries and to visualize the stored SSMs as natural ER models.

The server part of *Firestorm* remains merely unchanged. It stores SSMs in a relational database and provides algorithms to specify the similarity between SSMs. This similarity exploits the adjacency structure of SSM graphs. Given two graphs $G = (V, E)$ and $G' = (V', E')$, we look at partial mappings π of the nodes from G onto the nodes of G' which only map nodes onto nodes of the same type. The number q expresses the number of edges in G that are implicitly mapped onto edges in G', i.e. edges $(v, w) \in E$ such that $(\pi(v), \pi(w)) \in E'$. The *matching quality* realized by the mapping π is defined as $d(\pi) = 2q/(|E| + |E'|)$. The similarity between two graphs is then defined by the maximum over all matching qualities of mappings from G onto G'. Mappings are always one to one. The matching quality is a rational number between 0 and 1 with a value of 1 indicating that there exists a mapping between the library graph and the query graph that matches exactly all edges. As we can assume w.l.o.g. that there are no isolated nodes in a SSM, this is the case if and only if both graphs are isomorphic.

The authors of [11] showed that finding a mapping of optimal quality is *NP*-hard. But the three-layer structure of SSMs supports the design of heuristic and exact retrieval algorithms as well as a fast filter. The exact algorithm enumerates all feasible mappings of nodes from the second layer and solves for each of them the matching problems on layers 1 and 3 to optimality by using algorithms for weighted bipartite matching. The exact algorithm is polynomial for a fixed number of nodes on layer 2, and a reasonable algorithm for small numbers of nodes on layer 2. Filter and heuristic algorithms speed up the retrieval time tremendously by reducing the set of graphs to which the exact algorithm has to be applied. Further details of the algorithms are explained in [11].

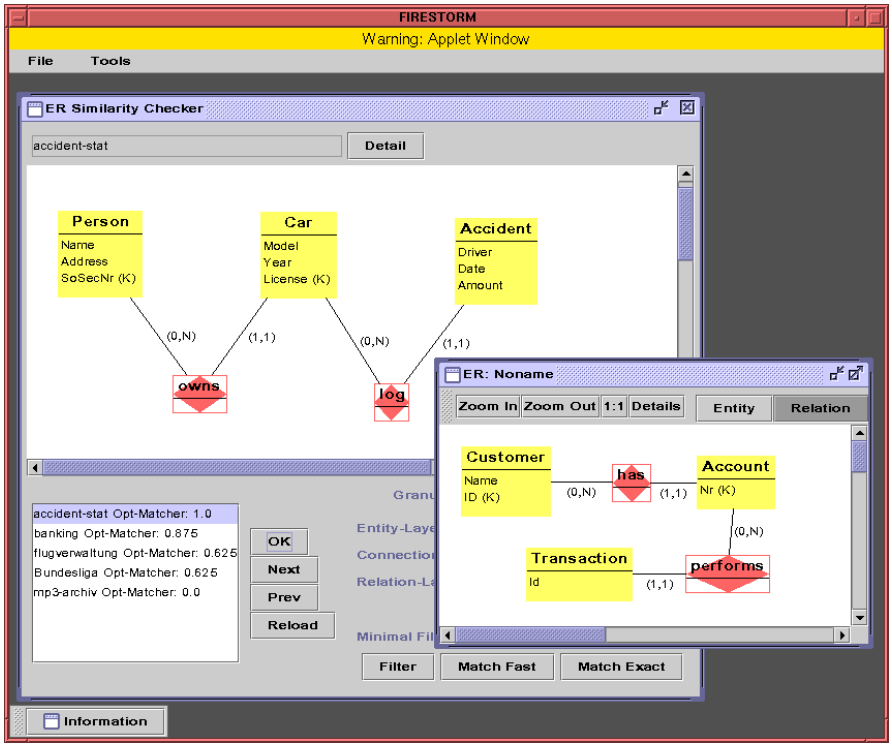

Fig. 2. Applet interface of *Firestorm*

As an example, figure 2 shows the search for a site that deals with information in the context of online banking. Customers have accounts and perform transactions. There is actually no online banking system registered in *Firestorm* and we get a top match for a system that logs car accidents because the graph structure is identical. Obviously this is not an answer we want to achieve. So there is a need to combine the structural retrieval with

context-related keyword search. Section 4 focuses on that task. Furthermore we added new functionality to *Firestorm* to fine-tune the retrieval algorithms with respect to the retrieval situation of a user. These functionalities are explained in section 5. But first of all, reporting experiences with a class room exercise will indicate that structural similarity of ER models is a good measure for the closeness of corresponding real world situations (if the context is predefined).

3 Evaluation of Structural Retrieval

This section will show how well the structures of SSMs capture the semantics of corresponding ER. Modelling is an art and one may argue that e.g. the same real world situation can be modelled in very different ways leading to different SSMs. The structural retrieval approach reduces the detection of similarity between different ER models to the graph similarity between their SSMs but the retrieval quality depends on semantic similarity between ER models. The restrictiveness in structure and available means to create ER models somehow limits the modelling freedom but a mathematical quality proof is out of sight.

As explained in the previous section *Firestorm* was extended to the domain of ER models. This means that the system can report the similarity between two different ER models. The focus of our working group at the University of Tübingen (Germany) is on database and information systems. Part of a database course is to learn how to create ER models that describe an aspect of the real world to be managed with a database. In this course *Firestorm* is used by students to create and manage their ER models.

The course contains three exercises that require students to design ER models for three different real world cases. These cases are formulated as natural language text. The text implies the naming of entities and relationships so that we can concentrate only on the structure. Some staff members evaluated the solutions of the students and assigned credit points according to how well the ER models represent the real world cases. For each exercise a staff member created a master model representing the expected solution. This made it possible to evaluate whether high similarity on the SSMs (respectively the corresponding ER models) reflects a certain similarity according to the semantics of the modelled real world cases, expressed by a high grade. A positive answer would promise to apply the structural retrieval approach for the domain of ER models.

These master models were used as queries to the system with the corresponding student models building the library. The system computed the (graph) similarity between each student model and the master model. The similarities were converted into credit points by simply multiplying the similarities with maximal credit points.

The results are encouraging. Figure 3 relates the credit points assigned by the staff members to the credit points assigned by *Firestorm*. The differences in credit point assignment are marginal so it seems that graph similarity between SSMs can be used to determine semantic similarity between ER models (as long as the context is clear).

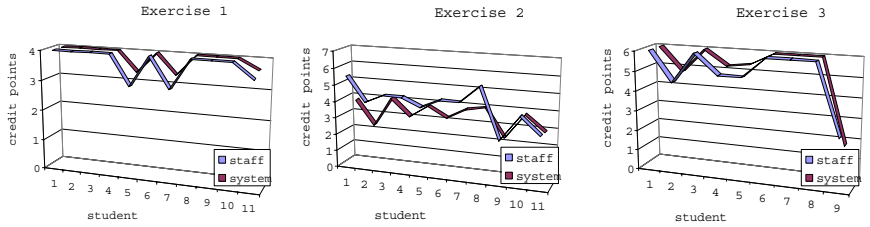

Fig. 3. Credit point assignment (staff vs. *Firestorm*)

4 Combining Structural and Keyword-based Retrieval

The results of the previous section are very promising but as the example usage of section 2 shows structural retrieval on its own cannot distinguish between different contexts. The application of *Firestorm* on the Web with thousands of sites offering heterogenous information needs a mechanism to influence the structural retrieval by something describing the context.

A meaningful context description is given by the names of the elements (e.g. Entities, Relationships) that compose an ER model. To calculate the structural similarity the algorithms map nodes on nodes and count the number of implicitly overlapping edges. For all algorithms we can define weights per edge without changing the algorithm structure. Structural retrieval can then be combined with some kind of keyword-based retrieval in the following way:

1. The system fills a matrix that keeps the closeness between node names for each combination of nodes. If the system only allows the usage of names of a controlled vocabulary or names with a given semantic (e.g. by RDF [15]) the normalized string distances (e.g. edit distance) express the closeness by a value between 0 and 1 with a value 1 indicating that two names have an identical meaning.
2. The retrieval algorithms use the values of the matrix to identify the weight of realized edge mapping which is the mean value of the corresponding node mapping matrix values.

The similarity between structural identical graphs that have no node names in common will get much lower.

5 Dynamic Recall/Precision Control

Like in any other retrieval system, the retrieval quality can be measured in terms of *Recall* and *Precision*. Recall is the quotient of the number of relevant ER models contained in the result set by the number of all relevant ER models of the system with respect to a query. Precision is the quotient of the number of relevant ER models contained in the result set and the size of the result set with respect to a given query. Recall and precision are no objective measures because each user has its own definition of relevance. In the following we use recall and precision in a weaker sense defining that all ER models are relevant with respect to a query if the similarity is more than a certain threshold. The threshold can be defined by a user just before starting a retrieval action. Additionally we implemented a functionality to control the behavior of the structural retrieval algorithms.

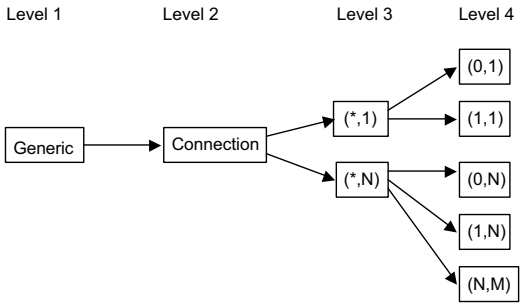

Fig. 4. Node type tree of second layer

The control functionality relies on a type tree of SSM nodes. As we know from section 2 the elements of ER models are mapped onto SSM nodes of different types. We define a type tree that starts with a generic root type and and specializes the types along the tree paths. Figure 4 shows the subtree of the second layer SSM node types for the domain of ER models. Following the root type the second level defines the types of all nodes on the second layer. These nodes represent connections between entities and relationships. The third level of the tree distinguishes between `(*,N)` connections and `(*,1)` connections that represent two different kinds of connection cardinalities in (min, max) notation. The fourth level of the tree captures the most specific cardinality distinction (`(0,1)`,`(1,1)`,`(0,N)`,`(1,N)` and `(N,M)`).

Assume that the SSM of the query model and the SSMs of all library models only contain nodes of the most specific type tree levels. Before a user triggers a retrieval action the user specifies for each layer which tree level should be used by the algorithms for the similarity computation, in other words which subtypes can be matched to each other. This means e.g. that if a users chooses the third tree level for the second layer the algorithms can

map a node of type `(0,N)` on a node of type `(1,N)`. This is impossible if the user chooses the fourth tree level for the second layer.

The effect of this mechanism is that for a certain threshold choosing a higher type level reduces the size of the result set. Some of the relevant models may not appear in the result set but the precision of the remaining models is higher. On the other hand setting a low type level leads to a large result set with possibly a lot of irrelevant models. But irrelevant models can contain a lot of hints that help to specify the type levels in a way that the results meet a user's expectations. With this functionality a user can adjust the retrieval system towards the individual sense of relevance and quality.

6 Related Work

Searching for information that is hidden in the Deep Web is not a new task. Sites providing information generated by scripts out of data stored in databases exist nearly as long as the Web exists. There are a number of search engines that index those sites and query them according to some user input. We look at some of those engines and relate them to the structural retrieval approach.

One of the most popular commercial Deep Web search engines is *LexiBot* [3] from BrightPlanet. It searches among 2200 databases and provides a query interface for simple text or boolean queries. The users can adjust the search strategies and result presentation to its own preferences.

Also from BrightPlanet is the free search engine *CompletePlanet* [2]. It contains the addresses of about 103000 databases organized in a directory structure with categories and subcategories. The search interface allows simple text input.

The *LII* [9] is a free search engine with an annotated, searchable subject directory of about 9000 Web resources. The resources are selected and evaluated by librarians for their usefulness to users of public libraries. Simple text and boolean operators are the means to state queries.

The search engines of IntelliSeek (*ProFusion* [8] and *InvisibleWeb* [7]) resemble much the other mentioned directory-based search engines.

Neither of the presented search engines use the database structure of Deep Web information systems. They describe the content of those systems with keywords and organize them (by hand or automated) within a directory. Query interfaces require simple text and boolean queries. They are easier to use than our query interface that is based on ER models. But especially the structural dependencies between different information units captures much more semantics and therefore allows a more precise search.

7 Summary and Conclusion

This paper describes a structural retrieval approach to search for Web sites that provide information contained in the Deep Web. Therefore we extend the general model retrieval framework *Firestorm* to the domain of ER models. Influenced by the special requirements of the Web we describe a technique to combine structural and keyword-based retrieval. A functionality for the dynamic control of retrieval behavior based on type trees provides the users with means to adjust the system with respect to the individual preferences according to recall and precision.

Users have to state their queries to the system by specifying ER models to describe the desired information in structural correlation. Many users are unfamiliar with ER modelling but it is a standardized and powerful way to express dependencies between information objects. The simple graphical user interface should allow users with basic knowledge to specify their own ER models.

Another drawback of the approach is the absence of ER models that Web sites need to register with the system. Most of the target sites store their data within relational databases. If they do not have ER models that describe their databases a tool could probably create a model out of the database schema.

Apart of the sandbox evaluation presented in section 3 the structural retrieval approach has yet to prove its applicability under the real world conditions of the Web. Therefore we hope that many Web sites register to our system to provide Web users a promising way to search for information of the Deep Web.

Looking into the future the ongoing hype of the Semantic Web [5] brings up many applications that require tools to handle semantic descriptions of information. The semantic description is mostly done following the principles of RDF [15]. RDF definitions resemble graphs that show the dependencies among the individual elements. The management of RDF definitions can benefit a lot by an RDF extension of *Firestorm*.

References

1. P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A vision of management of complex models. *SIGMOD Record*, 29(4):55–63, 2000.
2. BrightPlanet. *CompletePlanet*, 2000. http://www.completeplanet.com/.
3. BrightPlanet. *LexiBot*, 2001. http://www.lexibot.com/.
4. P. P. S. Chen. The entity-relationship model — toward a unified view of data. *Proceedings of the 1th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Kerr (ed), pp.173*, 1975.
5. S. Decker. *The Semantic Web Community Portal*, 2001. http://-www.semanticweb.org/.
6. A. M. Geoffrion. An introduction to structured modeling. *Management Science*, 33(5):547–588, 1987.

7. IntelliSeek. *InvisibleWeb – The Search Engine of Search Engines*, 2001. http://-www.invisibleweb.com/.
8. IntelliSeek. *ProFusion*, 2001. http://www.profusion.com/.
9. Library of California. *Librarians' Index to the Internet*, 2001. http://-www.lii.org/.
10. S. Müller. *FIRESTORM – FIrst a REtrieval SysTem for Operations Research Models*, 2002. http://firestorm.informatik.uni-tuebingen.de/ermodeling.
11. S. Müller and R. Müller. Retrieval of service descriptions using structured service models. In *Proceedings of the 10th Annual Workshop on Information Technologies and System*, pages 55–60, 2000.
12. S. Müller and R. Schimkat. A general, web-enabled model retrieval approach. In *Proceedings of the International Conference for Object-Oriented Information Systems (OOIS 2001)*, 2001.
13. L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*, 1998. http://citeseer.nj.nec.com/-page98pagerank.html.
14. UDDI community. *Universal Description, Discovery and Integration of Business for the Web (UDDI)*, 2002. http://www.uddi.org/.
15. World Wide Web Consortium (W3C). *Resource Description Framework*, Feb. 1999. http://www.w3.org/TR/REC-rdf-syntax/.