Semantic Methods and Tools for Information Portals

Sudhir Agarwal[1], Peter Fankhauser[3], Jorge Gonzalez–Ollala[1], Jens Hartmann[1], Silvia Hollfelder[3], Anthony Jameson[2], Stefan Klink[4], Patrick Lehti[3], Michael Ley[4], Emma Rabbidge[4], Eric Schwarzkopf[2], Nitesh Shrestha[3], Nenad Stojanovic[1], Rudi Studer[1], Gerd Stumme[1], Bernd Walter[4], Alexander Weber[4]

[1] AIFB, University of Karlsruhe, D–76128 Karlsruhe, Germany
[2] DFKI, D–66123 Saarbrücken, Germany
[3] FhG IPSI D–64293 Darmstadt, Germany
[4] DBIS, University of Trier, D-54286 Trier, Germany

Abstract. The paper describes a set of approaches for representing and accessing information within a semantically structured information portal, while offering the possibility to integrate own information. It discusses research performed within the project 'Semantic Methods and Tools for Information Portals (SemIPort)'. In particular, it focuses on (1) the development of scalable storing, processing and querying methods for semantic data, (2) visualization and browsing of complex data inventories, (3) personalization and agent-based interaction, and (4) the enhancement of web mining approaches for use within a semantics-based portal.

1 Introduction

Information portals provide access to high quality information for specific communities. Often the content already exists before and is replicated from other web pages as, e. g., in our scenario, from different servers for bibliographic data. However, efficient management of knowledge is difficult in such portals, as there are typically several content providers who all have their specific focus on the data. The data are thus heterogeneous both in syntax and semantics. While the problem of syntactical mismatch is expected to decrease with upcoming standards like XML and RDF, semantic heterogenicity remains a challenge.

Ontologies are nowadays considered as an important building block to overcome this challenge. As inherent central component of an information portal, an ontology represents the formal model of all relevant information, containing among other things a hierarchy of terminology, relationships between the specific terms as well as axioms. The ontology supports the structured storage of the information, the (decentralized) integration of information into the portal, the deduction of additional, implicit information, the navigation in the available information and its personalized presentation, thereby facilitating the portal users to locate relevant information more easily.

This vision is following the general trend of using rich conceptual models (such as ontologies) for data management and presentation, which can be traced in several communities, as in the database community (e. g., [26]), the knowledge representation community (e. g., [10] [16]), or the information system community (e. g., [1, 8, 7, 12, 11, 23]).

The intention of the project *Semantic Methods and Tools for Information Portals (SemIPort)*[1] is to continue this trend; and to evolve a set of semantics-based methods and tools for representing and accessing information, by combining technologies from areas such as meta data, knowledge presentation (ontologies, logic formalisms), personalization and visualization with present web standards (XML(S), RDF(S) and OWL). The aim is to extend information portals for scientific communities with advanced features, while additionally offering possibilities to integrate own information. The specific research tasks within SemIPort are: (1) the development of scalable storing, processing and querying methods for semantic data, (2) visualization and browsing of complex data inventories, (3) personalization and agent-based interaction, and (4) the enhancement of web mining approaches for use within a semantics-based portal. These research tasks will be discussed in more depth in the subsequent sections.

For testing purposes, the developed approaches will be evaluated on data from the online bibliography DBLP, and the tools are planned to be integrated into the competency and service network portal *IO-Port* of the Gesellschaft für Informatik(GI) which is currently under construction within the project FachInformationsSystem Informatik (FIS-I). There, FIZ Karlsruhe, GI, and the universities of Karlsruhe, Munich and Trier aim at developing and running a competency and service network for the German computer science community in form of an Internet portal.

2 Querying and Integrating Ontology-Based Data Repositories

In the given scenario for information portals, an ontology is used as basis for the portal. This ontology serves as a view on the available data sources, which is presented to the user. Therefore it is transparent to the user, where and how the data is stored, which syntax is used for the data e.g. XML [5], RDF [19], OWL [9] or relational data and which concrete schema is used for a data source.

In the context of an information portal one needs to deal with two kinds of data sources. First such a portal normally has an own internal repository that integrates several data sources by importing and integrating them based on one internal schema. This schema of the internal repository will be closely related to the used ontology. Additionally, an information portal will also provide access to external data sources, which cannot be imported into the internal repository, e.g. because these data sources are not under control of the portal. Such external data sources are normally significantly less closely related to the ontology.

To query such data opens up several research topics and challenges. In order to allow the user to query the data sources on the basis of the knowledge of the ontology, a language for querying ontologies and their instances must be defined. Furthermore, an efficient query evaluation strategy for distributed external data sources as well as for the internal repository must be developed. It must also be analyzed which storage method is best suited for the internal repository.

Another topic is the integration of distributed data sources. We will discuss how known approaches like data warehouses and mediator/wrappers can be enhanced with semantic methods, and how the latter can be extended with Semantic Web Services.

[1] http://km.aifb.uni-karlsruhe.de/semiport

2.1 Design of a Query Language

For the need for a query language that is only based upon an ontology, the so-called ”Semantic Web Query Language – SWQL” (pronounced ”swequel”) was designed. The syntax of SWQL is based on XQuery [4] – the currently emerging standard for an XML query language – but the underlying semantics has significantly changed. SWQL uses OWL [9] - the emerging standard for a Web Ontology Language – as its type system and a corresponding graph data model instead of tree based XQuery data model. This data model is therefore very similar to RDF data models.

The formal definition of the data model for OWL instances as used by SWQL is as follows. All parts of a graph are called items. Items are either nodes or properties (refer to line 1 in figure 1) A node is either an object or a literal (refer to line 2 in figure 1): The following functions are defined on Items, the function declarations use an XQuery syntax. Every Item of a graph has an associated type from the OWL ontology. The ”type” function returns this associated type as a qualified name as defined in XML Schema (refer to line 3 in figure 1). Every Property connects two nodes. The ”domain” of a Property is always an Object, the ”range” can be any Node (refer to line 4 and 5 in figure 1). All nodes $n in a graph may have Properties, where $n is the range of the Property. The ”toProperties” function returns a set of such properties. For Literals this set is always non-empty (refer to line 6 in figure 1). All Objects $n may have properties, where $n is the domain of the Property. The ”fromProperties” function returns a set of such Properties (refer to line 7 in figure 1). Every Literal has a value, which is an atomic type as defined in the XQuery 1.0 and XPath 2.0 specifications (refer to line 8 in figure 1). Finally, the function ”all” returns a set of all existing Nodes in the graph (refer to line 9 in figure 1).

```
1  Item ::= Node | Property
2  Node ::= Object | Literal
3  type($i as Item) as xs:QName
4  domain($p as Property) as Object
5  range($p as Property) as Node
6  toProperties($n as Node) as Property*
7  fromProperties($n as Object) as Property*
8  value($l as Literal) as xdt:anyAtomicType
9  all() as Node*
10 Publication()
11 Publication()/hasAuthor
12 Publication()[hasAuthor = "I.M.Author"]
```

Fig. 1. SWQL Design

SWQL borrows its control constructs from XQuery, but replaces XPath [3] (the language to select specific nodes from the XQuery data model) with SWQLPath, a language to select specific nodes from the SWQL graph data model. The two main constructs in SWQLPath are so-called NodeTests and PropertyTests. NodeTests filter a set of nodes for being an instance of a given OWL class. The following simple NodeTest query selects all nodes that are an instance of the class ”Publication” (refer to line 10 in figure 1).

PropertyTests are used for navigation via the edges of the graph. They check a set of current nodes, if they have a property that is an instance of the given property type and if yes, they return the corresponding node in the range of the property. The following query selects all authors of all publications (refer to line 11 in figure 1):

As in XPath, SWQLPath supports also so-called Predicates. Predicates filter a set of nodes based on some condition. The following query selects all publications, where ”I.M.Author” is one of the authors (refer to line 12 in figure 1):

The main construct of a SWQL query are FLWOR expressions as in XQuery. For example, the following FLWOR query selects the titles of all Publications, whose title contains ”Semantic Web” and which was written by ”I.M.Author”:

```
for $a in Publication() where
  contains($a/title, "Semantic Web")
  and $a/hasAuthor = "I.M.Author"
return $a/title
```

FLWOR expressions allow to define very complex and compositional queries. Besides of FLWOR expressions, SWQL also supports conditional (if-then-else) expressions, existential and universal quantification, user-defined functions, type switches and casts.

2.2 Data Integration

As already mentioned above, in the context of an information portal one has to deal with two kinds of data integration. One is the integration of data sources into the internal repository of the portal, the other is the integration of external data sources that cannot be imported. The former data integration concept is called a data warehouse approach, the latter a mediator-wrapper approach [14].

DataWarehouse Approach. The first step when setting up a data warehouse is the definition of the import and mapping of the data sources to the internal schema of the data warehouse. Based on the ontology information it may also be possible to infer additional information during the import step. One remaining challenge is the efficient evaluation of SWQL queries against this data warehouse. For this purpose, an appropriate index mechanism must be developed.

These indices should provide a way to find instances of a specific class (for the evaluation of NodeTests), find instances that can be reached by a specific property (for the evaluation of PropertyTests and full text indices (for the evaluation of filters). The main difference between these new indices for ontology based queries and index mechanisms in the field of relational or object-oriented query processing are as follows: Classes are not only in a hierarchy, but there might also exist equivalence statements between classes. Properties are first class objects, i.e. they are themselves structured within a hierarchy and equivalence statements might exist as well.

Duplicate Detection. While dealing with information sources of similar domains, one encounters two main problems regarding the identities of real world individuals. Firstly, the data provided by one information source may represent the same real world individual as the data provided by another information source, even if their syntax is different. For example, in two different bibliographic databases “Carl Friedrich Gauß” and “C. F. Gauss” may represent the same person. Secondly, the data of different information sources with same syntax may represent different real world individuals. For example, in two different bibliographic databases “C. Müller” may represent two different persons.

We are currently developing semantics-based heuristics for the domain of bibliographic databases and, which simulate the behavior of human experts. We consider the title of a publication, the publisher, year of publication, conference name, co-author relationships, editors etc. to detect the identity of persons (authors or editors) and articles.

Mediator–Wrapper Approach. The mediator-wrapper approach for the integration of external data sources has to deal with different problems. It consists of a mediator component, which is the interface to the user and several wrapper components that encapsulate the functionality to access a specific data source. The mediator has several tasks to fulfil, at first it has to analyze the incoming queries in order to determine all relevant sources for this query. Then the query must be localized for every relevant source in order to contain only concepts and properties that exist in the corresponding source. If the data sources contain equivalent instances e.g. information about the same publication, but with different kind of additional information, the single query results must be joined at the mediator. Therefore the queries may need to be rewritten in order to return all needed join predicates as well. These queries are then distributed and the results are merged or joined within the mediator. The mediator might also need to do some post processing (e. g., duplicate detection as discussed in Section 5) on the remaining result in order to answer the full original query.

The wrapper components need to translate the SWQL queries to a query language that can be answered by the data source. The results of these local queries must then be translated to instances of the SWQL data model in order to return these to the mediator. The wrapper might also need to do some post processing, if the local query language is not as powerful as SWQL.

2.3 Semantic Web Services

In a web scenario – and hence in particular for information portals –, the mediator/wrapper approach can be complemented by Web Services. Web Services are platform–independent, programming–language–independent and via world wide web accessible machine interfaces that can be accessed by authorized users. Web services offer a standard medium for communication between hetergeneous systems. Besides supporting information integration in the backend, Web Services play a second, important role for information portals: portals can offer their services as machine interfaces by using web services in addition to the user interface for human consumption (e.g. HTML), thus making new business models possible.

The data that is transmitted via Web Services is usually encoded in XML and thus does not have machine understandable semantics. Web Services are described with so called web service description languages such as WSDL, which is the current W3C standard. WSDL however has certain limitations. For example, it is not possible to specify the functionality and other characteristics of a Web Service in a machine understandable way. The ultimate consequence of the limitations on the data description level and on the service description level is that a significant amount of intellectual effort is needed to incorporate Web Services in any application.

Ontologies play an important role in solving the first problem, i.e. describing the schema and data with machine understandable semantics. To solve the second problem a web service description language is needed that allows to specify on a semantic basis the functionality and other characteristics like security and quality of a Web Service and also its relations to other Web Services.

Most of the currently available web services are simple web services with short execution time. Upon receiving the values for the input parameters they select some

information from their repository, or perform some calculations or do both. That is, most of the current web services realize a relation between a set of input parameters and a set of output parameters. Since the semantics of data and schema and hence of the input and output parameters is understandable for machines, the functionalities of such simple short-lived web services can be specified by a relation in the ontology of web service provider.

Often, the input parameters of a web service must fulfill certain constraints to enable successful execution of the web service. For example, validity of a credit card can be seen as Boolean attribute of a concept CreditCard. The value of such an attribute can then be specified by the rule "if expiration date is greater than or equal to current date then true else false". The constraints that the output parameters fulfill after the execution of a web service can be specified analogously. Logics like description logics and F-logic allow to specify such rules and axioms for ontologies formally with machine understandable semantics. Hence, the functionality of such web services can also be specified by a relation in the ontology (with appropriate rules and axioms) of a web service provider.

In order to invoke a web service, it is not enough if a requester machine knows merely the semantics of the input parameters. A requester machine must also know, how it can construct the required input parameter from the data it has. For example, consider a concept Person with a property Name in the ontology of a web service provider. The property Name contains the full name of a person, e.g. "Albert Einstein". Further, consider a concept Scientist with properties FirstName and LastName in the ontology of a service requester. FirstName contains the first name of a scientist, e.g. "Albert" and LastName his last name, e.g. "Einstein". Now, if the web service provider offers a web service that has an input parameter of type Name, the service requeser must know that the value of Name is equal to the concatenation of the values of FirstName and LastName with a blank in between. The Web Ontology Language (OWL, [9]), which is gradually becoming the standard ontology language, offers very few simple constructs to define ontology mappings. String manipulation operations and arithmetic operators are not part of it. Hence, there is a need for a high level language similar to a programming language that allows to specify mappings like simple computer programs that can be automatically executed by machines. Similarly, the values of the output parameters of a web service can be converted to the instances of appropriate concepts in the requester's ontology.

Web services seen as standard communication medium among heterogeneous information sources allow a significant degree of automation. Together with ontology mapping rules, they promise even more automation thus saving a lot of intellectual effort that is currently needed while integrating heterogeneous information sources available on the world wide web.

3 Visualizing and Browsing Complex Data

The search for helpful information can be a difficult task especially within complex data. One of the parts within the SemIPort-Project is to help and lead the user with a sophisticated user interface. In the first project phase, we developed a prototype of

an interface, and are currently testing it on the DBLP database. In a later stage, it is intended to apply it to the portal being developed in the FIS-I project.

This prototype, namely *DBL-Browser*[2], is an essential interface to the data and plays a central role in the search task. Within the DBL-Browser a smart search function and all developed textual and graphical visualization models are integrated. Furthermore, the possibility of data acquisition by some authorized user is given. The DBL-Browser is based on a central platform which is being developed in parallel to fit to the growing user and application requirements.

3.1 Platform Basis

An essential part of the visualizing and browsing framework shown in Figure 3.1 is the platform basis, which provides testing for several information retrieval and visualization techniques without changing the user interface. All requests coming from the user interface are answered by the platform basis and it in turn supplies all the data used for visualization. Rapid development and high performance are important issues.

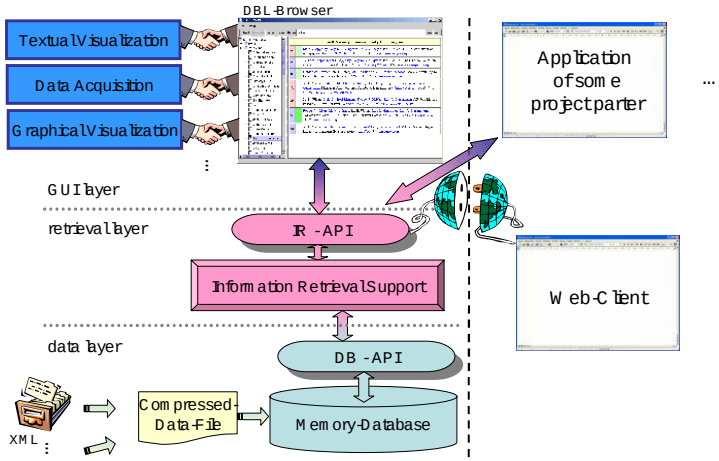

Fig. 2. Modules of the system basis

Main Memory Database. The underlying data is represented first as a plain XML file. For efficiency and performance reasons this file is converted into a compressed format which is then read into the main memory where the compressed data is directly used within specially developed data structures. An adapted version of Huffman coding [27] and other complex methods are used for the compression of text parts or numbers, respectively.

With these techniques it is possible to load the complete data of the DBLP database with more than 400.000 bibliographic entries into the main memory of a computer. It requires just 60 MB without any losses and without changing the structure of the data.

[2] **D**igital **B**ibliographic **L**ibrary Browser

Information Retrieval Support. The information retrieval module supports the applications with well-defined search functionalities and plays the role of an interface to the underlaying data. In the current state, this interface is implemented as a java API, but in a later stage of the project a mediator-wrapper approach may be integrated to use the SWQL approach described before (see also section 2.1). Furthermore, it is planned to integrate more complex data mining functions (see also Section 5). With these, a set of efficient algorithms will be available for the visualization and browsing modules. On this foundation, a variety of visualization tools can be developed.

User Interface. One example is the DBL-Browser which is used to access and browse through the complete DBLP data on a stand-alone computer (or laptop) without the necessity of a network (see Section 3.2). Another example is the planned integration of a data acquisition module, which will not only visualize the existing (static) data but gives the opportunity to change and extend the data base by authorized users.

The user interface has a modular design and makes use of public interfaces for the easy integration of more visualization and browsing tools. The advantage of this structure is also used by the client/server architecture where a browser is on the client-side and the complete retrieval package with the underlaying data is on the server-side.

3.2 Browsing Complex Data

The first application to give a comfortable and easy-to-use access to the complete DBLP data is the DBL-Browser. It sits on top of the platform basis and offers some additional functions, which are not realized in the current web page of the DBLP because of efficiency reasons. These functions ease the interaction with the data and help the user find the desired information in a faster way. The Browser is mainly divided into three fields. **(A)** the search field (see Figure 3) for direct access, **(B)** The tree-like navigation within the structured views and **(C)** The textual and graphical visualization of the data.

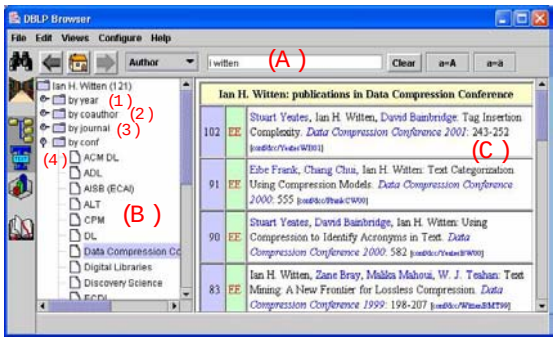

Fig. 3. Example view of the browser application

Search Field (A) The search field looks like the search field of a classical internet browser. Besides the usual 'back', 'forward', and 'home' buttons for navigating through the browsing history, the search field gives the user the possibility to search directly for an author. This can be done case (in)sensitive or (not) regarding umlauts (a=ä). If the query is ambiguous or if several authors can be found with this name, the user can select the appropriate author out of a list of possible candidates.

Navigation (B) serves for the navigation within the thematically structured views and for selecting between several view levels. The thematic views to the retrieved data

gives the user the possibility to structure the data with different criteria, i.e. publication years **(1)**, coauthors **(2)** or publications in certain journals **(3)** or conferences **(4)**. With the selection of relevant views and view levels – as it is known by the Windows-Explorer tree view – the user can easily select desired publications.

3.3 Visualization.

The main part of the application is the visualization **(C)**. The retrieved data can be visualized in a textual or in a graphical way. The user can, at any time, switch between the style of visualization used to display the data.

Textual Visualization. The textual visualization of the author pages corresponds in a large extent to the web pages of the DBLP[3]. The mark-up 'EE' at the left of a publication indicates the presence of a link to an electronic edition. To the right are the bibliographic data of the retrieved publications. First, all authors are listed in the sequence given in the original paper. With a single click on a coauthor a new query is generated and all publications are searched and visualized analogously. With this behavior the user can easily overview coauthor networks. Directly behind the authors, the complete title is listed. By clicking on a word, a search is initiated which returns all publications where this word occurs in the title. After the title, the kind of publication (journal or conference) and the page numbers are listed. By clicking on the journal/conference the users switch to the table of contents view for that publication, where they can then search for more publications of interest.

Graphically Enhanced Visualization. The DBL-Browser does not only use textual visualization to communicate data to users. The browser will also include a range of graphical visualizations aimed at helping the users quickly find the information they desire. Currently there are some basic visualizations which show the textual data in a graphical form – like the number of publications the author has published over the years, or in different journals.

A more advanced graphical visualization shows connections between authors. This is referred to a *co-author graph* and exposes underlying relationships in the data. For example, the frequency that authors have worked together. Alternatively this style of graph is used to graphically show where an author has published most frequently.

More complex visualizations show the conferences and journals in *relation* to each other – so the connection between different streams of publications are *visible*. With similar technologies it is possible to show which conferences or journals where en vogue in which period, or to show the rise and fall of different topics.

4 Personalized and Agent Based Interaction

Advances in user interfaces as well as search and browsing technologies have simplified gaining access to the vast amount of information in today's digital libraries and portals. But many issues remain if we aim for information portals which are not only easy to use, but also effective in providing the information and, going a step further, facilitating gaining the knowledge we actually need.

[3] see also http://dblp.uni-trier.de

Interaction with information does not end with retrieving relevant documents [25], but also includes making sense of the retrieved information, organizing collected materials for near or long-term use, and sharing insights with colleagues. With the work described in this section, our goal is to create an interaction environment which facilitates these tasks by combining different techniques already developed for each separate task, and leveraging them by providing a common representation of the user's interests and current working context. In particular, the envisioned system consists of a tool for organizing personal document collections, an automatic recommender, an information filter, and a browsing assistant. Sharing data about the user among these different components will, for example, allow the recommender to increase the accuracy of its recommendations by taking the user's current working context in the document manager into account. Thus, in addition to directly supporting the user by the provided functionality, the envisioned system can utilize the additional data about the user to provide contextualized and personalized support.

4.1 Document Manager

The document manager aims at supporting the user in managing and making sense of a personal document collection. To specify the requirements for this tool, one has to understand how people work with their documents. This is a well researched issue, and from the studies published in the literature (e.g. [21, 17, 22]) and interviews with students and researchers at the DFKI, we derived the following main requirements for the document manager:

- People use spatial organization to express relationships between (physical) documents, so the document manager should provide means to spatially organize document representations.
- When collecting documents, users often cannot immediately decide where to file or how to classify these documents. The tool should facilitate easy creation of informal structures out of newly discovered documents.
- People tend to organize their documents into hierarchical structures. The tool should support this activity.
- People annotate documents so it is easier for them to remember what was important about the information at a later time. Hence, the tool should allow users to associate an annotation with each document.
- Annotations are also used to express more specific relationships between documents (e.g., "Document A describes a user evaluation of the system presented in document B"). The document manager should explicitly support the definition and browsing of these kinds of relationships.
- The tool should support the user in finding documents in the personal collection.

These requirements led us to the adoption of a two-dimensional, zoomable plane as the central interface component of the document manager similar to systems like, e.g., NaviQue [13]. Representations of documents can be positioned and grouped freely on this plane, which facilitates spatial and informal organization.

In addition to associating text files containing in-depth notes with a document, the user has the possibility to associate objects in the personal document collection with

concepts from an underlying ontology, and define relations between those objects. Thus, the user can easily create a personal knowledge base. Defining relations is as simple as a drag-and-drop interaction. For example, if the user wants to express that document A describes an improved version of an algorithm originally described in document B, the user drags and drops the representation of A on top of B, which will initiate a dialog in which the user can select an appropriate relation from the underlying ontology.

This personal knowledge base simplifies browsing and querying the personal document collection, but it is also a valuable source of information about the user's knowledge and current interests.

We plan to integrate the document manager and the DBL–browser (see Section 3) to provide the user with a single tool to query, browse, and make sense of the contents of a bibliographic database of scientific literature and the related information in the personal document collection.

4.2 Recommender Systems

The user will at different times have different recommendation needs. In the context of a portal to scientific information, these will include:

- Finding current publications related to what the user is working on.
- Finding publications similar to those the user currently works with.
- Finding a certain type of publication (e.g., an empirical study) related to a given document or working context.
- Finding documents pointing to new ideas and solutions.

To fulfill these recommendation needs, different recommendation strategies are required. To this end, we plan to implement knowledge-based, content-based, and collaborative-filtering recommendation components, and combine these to a hybrid recommender [6] as appropriate for the specific recommendation need. For example, if the user is interested in documents similar to those she is currently working with, using a combination of a knowledge-based and a content-based recommender is more sensible than using a collaborative recommender, since the latter is more likely to recommend documents only marginally related to what the user is thinking about. But this latter characteristic is valuable when the user is interested in new ideas.

To support the user in evaluating documents while she is browsing the information portal, we plan to provide a *browsing assistant* (see, for example, Lieberman et al. [20]) which accompanies the user and recommends items on the page the user is currently visiting. Since the assistant has to rely on uncertain evidence (e.g., the time the user stays on a specific page) to adapt its user profile, inaccurate recommendations should not interfere with the user's task. Because of this, the assistant will not modify the structure of the page the user is looking at, but only highlight its recommendations. In addition, the assistant will always display a description of the current representation of the information need, so the user can quickly decide whether the recommendations are likely to be accurate or not. Finally, the user will have the ability to quickly deactivate the assistant.

4.3 The User Model

The system will maintain an internal representation of the user's interests in order to be able to better fulfill the user's information needs. This user model is the central means of interaction between the different proposed components: The document management tool will adapt the model on the basis of the user's interaction with the document collection, and hand it to the recommender whenever the user formulates a recommendation need. Then, when the user starts browsing the information portal, the browsing assistant fetches the profile from the recommender and adapts it according to the user's navigational behavior.

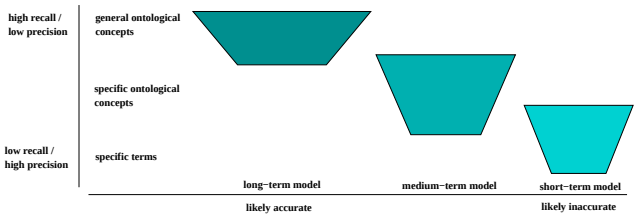

Fig. 4. Three layer user model

The proposed user model consists of three layers (see Figure 4.3):

- A general, fairly accurate, only slowly adapting long-term profile. This layer represents the user's general interests in terms of research areas.
- A more specific medium-term profile which represents the user's current working context. It consists of the most specific research areas covering the documents in the context, concepts associated with these documents (such as authors and conferences), as well as a representation of the documents themselves.
- A very specific, quickly adapting, but possibly less accurate short-term profile. It represents the information need during the user's current interaction with the system, and contains the same kind of information as the medium-term profile, but puts a stronger emphasis on document terms than on more general ontological concepts.

The proposed user model is primarily based on the interaction with the personal knowledge base the user creates with the provided document manager (see above), but also takes the interaction with the information portal into account by utilizing results from the semantic usage mining component of the OntoManager, which is described in the next section.

As an example for the utilization of this user model, take a possible strategy for recommending related material: To fulfill this recommendation need, the recommender could use a cascade of knowledge-based, collaborative, and content-based modules. First, the relevant concepts of the ontology provided by the user model are used to focus on the associated set of documents. Next, documents used by users with similar profiles are selected from this set. Finally, the set of relevant terms stored in the profile is used to identify those documents in the reduced set which are most closely related to the user's work.

5 Discovering New (Meta-)Data and Relationships

In an ontology-based information portal, ontologies support the process of "indexing" content of an information resource – so called semantic annotation – and the navigation

through the knowledge repository – so called conceptual navigation. However, ontologies, as a conceptual model for the given business domain, should react to all changes in the business environment. This includes accounting the modification in the application domain or in the business strategy; incorporating additional functionality according to changes in the users' needs; organizing information in a better way etc. For example, if the underlying ontology is not up-to-date, then the effectiveness of the searching for information decreases significantly. Hence, in order to be usable, an ontology-based application should be continually adapted to the changes in its environment. Such changes can be discovered with (web) mining techniques

Since we consider portals that are ontology-based, we can exploit the semantics for improving the mining results. This vision of *Semantic Web Mining* has been described in [2]. There are three ways which are commonly distinguished in (Semantic) Web Mining, and which can also be used to discover the change of patterns in content, structure, and usage of the portal:

- **Usage Mining**: Identifying regularities in the usage of information resources, which enable the discovery of rarely used/overused topics in the portal.
- **Structure Mining**: Identifying regularities in the structure (mainly linkage) of information resources, which enable the determination of the most/less important resources in the portal.
- **Content Mining**: Identifying regularities in the content of information resources, which enable the discovery of topics which are well/insufficiently covered by the portal.

We discuss an approach which combines these three mining approaches in order to make an ontology-based application more useful. We present a tool for managing changes in the ontology, based (i) on the discovery of changed content/usage patterns, and (ii) on the output of a focussed crawler which gathers relevant pages from the web.

5.1 OntoManager

The OntoManager has been designed to provide methods that support ontology managers in managing and optimizing the ontology according to the users' needs. The system incorporates mechanisms that assess how the ontology (and by extension the application) is performing based on different criteria, and then enable to take action to optimize it. One of the key tasks is to check how the ontology fulfills the perceived needs of the users. By tracking users' interactions with the application in a log file, it is possible to collect information that can be used to assess what the main interests of the users are. In this way, we avoid asking the users explicitly, since they tend to be reluctant to provide the feedback via filling questionnaires or forms. Moreover, the entries in the log file are enriched with the information about the content/semantics of the visited pages. More information about such *Semantic Logs* can be found in [24]. Conceptually, the *OntoManager* consists of three modules:

- The *Data Integration Module* that collects data from different, possibly distributed logs in case an ontology-based application is deployed on several web servers; preprocesses data by transforming disparate data into meaningful information. This

module also covers the cleaning and validation of the data for achieving the required quality; and organizes them in a way that enables a fast and efficient access to the data.
- The *Visualization Module* that makes the integrated usage data more useful for human beings by presenting the data in a comprehensible visual form: Graph-, Table- or Bar-based
- The Analysis Module that provides guidance for adapting the ontology with respect to the users' needs. Two analysis are performed:
  - *Ontology Evolution* provides guidance in the process of modifying the ontology and ensures the consistency of the updated ontology. This module keeps track of the changes and has the possibility to undo any action taken upon the ontology. The OntoManager imports the functionalities related to the ontology evolution process that we elaborated in [18]. We omit the details here.
  - *Ontology-based Crawling* fills newly created concepts with the most promising instances that can be found in the web or an intranet. Ontology-based Web Crawler is described in more detail in the next section

5.2 Ontology–based Web Crawling

The process of identifying resources on the web that are relevant for a topic (web crawling) is a difficult task, which should be focused in order to reduce the search-space and search-time. The approach of our web crawler consists of several interchangeable and expandable modules. The *Crawler Module* retrieves desired web resources which are then processed (mainly text-processing) by the *Preprocessing Module*. The processed web resources are indexed and stored in a database (*Indexing Module*). Whereon stored resources are being **semantically analyzed and rated** in the context of a given ontology. The calculated relevances are used to feed the container of URLs which should be retrieved next by the *Crawling Module*. In the following we describe important aspects of our Ontology-based web crawling approach.

Knowledge Representation. We use ontologies as formal representation of background-knowledge to calculate relevances of web resources. We select between two kind of ontologies in our approach. Firstly, we use a *Web-Ontology* describing existing objects and corresponding properties of the environment (WWW) in which the knowledge is kept. For instance, the web consists of *hosts*, *documents*, *hyperlinks*, etc. Secondly, we use a *Domain-Ontology* representing knowledge about a specific domain, e.g. the domain *Computer Science*. A first prototype of an ontology for bibliography for the computer science domain was developed within the SemIPort project. For the ontology modeling and evolution engineering process we used the KAON framework.

Semantic Relevance of Documents. The huge amount of information and corresponding heterogenicity sources requires a detailed analysis and rating towards an identification and extraction mechanism. Hence, a semantic relevance computation for resources is needed. In contrast to information extraction methods, like webbase [15], which are calculating relevances after downloading all resources[4], our crawler assigns

[4] In particular, downloading a large amount of data and then applying algorithms like HITS or PageRank, exemplary.

relevances at runtime based on the actual knowledge and the given background-knowledge. As main result, we can reduce requirements for storage capacities, computing time and net traffic which leads to a smart web crawler runable on *normal* PCs.

The search strategy of the crawler focuses on the most promising resource which has been calculated previously by the *Computation Module*, whereby the calculation of a relevance for a resource is calculated on its content. Therefore we apply well-known text processing operations, like Porter's stemming algorithm, on the content and determine the semantic relevance of the crawled documents. Furthermore, we calculate relevances for each outgoing link of a document. These relevances are used to rank the links which should be retrieved next.

6 Outlook

In this paper, we discussed several ways to enhance information portals with semantics–based techniques. The approaches we presented have now to undergo further experimental evaluation. The information inventory available in the IO-Port prototype is supposed to be used for testing the methods and tools developed in the SemIPort project. Furthermore, the results gained from the SemIPort project shall be integrated into the further development of the IO-Port portal.

Acknowledgement

This work is supported by the German Ministry for Education and Research (bmb+f) in the SemIPort project.

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. Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards semantic web mining. In Ian Horrocks and James A. Hendler, editors, *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*, pages 264–278. Springer, 2002.
3. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, and Jérôme Siméon. XML path language (XPath) 2.0. http://www.w3.org/TR/xpath20/, May 2003. W3C working draft.
4. Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. XQuery 1.0: An XML query language. http://www.w3.org/TR/xquery/, May 2003. W3C working draft.
5. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/REC-xml/, October 2000. W3C Recommendation.
6. Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
7. 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.
8. 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.

9. Mike Dean, Guus Schreiber eds., Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/, March 2003. W3C working draft.
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):361–382, 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. George W. Furnas and Samuel J. Rauch. Considerations for Information Environments and the NaviQue Workspace. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 79–88. ACM Press, 1998.
14. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal: Very Large Data Bases*, 10(4):270–294, 2001.
15. Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, and Andreas Paepcke. WebBase: a repository of Web pages. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):277–293, 2000.
16. A. Hotho, A. Maedche, S. Staab, and R. Studer. SEAL-II — the soft spot between richly structured and unstructured knowledge. *Journal of Universal Computer Science (J.UCS)*, 7(7):566–590, 2001.
17. Alison Kidd. The Marks are on the Knowledge Worker. In *Conference Proceedings on Human Factors in Computer Systems (CHI'94)*, pages 186–191. ACM Press, 1994.
18. B. Motik N. Stojanovic L. Stojanovic, A. Maedche. User-driven ontology evolution management. In *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW'02*. Springer, 2002.
19. Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, February 1999. W3C Recommendation.
20. Henry Lieberman, Christopher Fry, and Louis Weitzman. Exploring the Web with Reconnaissance Agents. *Communications of the ACM*, 44(8):68–75, 2001.
21. Thomas W. Malone. How Do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, (1):99–112, 1983.
22. Catherin C. Marshall and Frank M. Shipman III. Spatial Hypertext and the Practice of Information Triage. In *Proceedings of the Eighth ACM Conference on Hypertext*, pages 124–133. ACM Press, 1997.
23. G. Mecca, P. Merialdo, P. Atzeni, and V. Crescenzi. The (short) Araneus guide to website development. In *Second Intern. Workshop on the Web and Databases (WebDB'99) in conjunction with SIGMOD'99*, May 1999.
24. J. Gonzalez N. Stojanovic, L. Stojanovic. On enhancing searching for information in an information portal by tracking users' activities. In *First International Workshop on Mining for Enhanced Web Search (MEWS 2002), held in conjunction with WISE 2002, Singapore, 2002*. IEEE, 2002.
25. Andreas Paepcke. Digital Libraries: Searching Is Not Enough. *D-Lib Magazine*, 2(5), 1996.
26. G. Wiederhold and M. Genesereth. The conceptual basis for mediation services. *IEEE Expert*, 12(5):38–47, Sep.-Oct. 1997.
27. Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. second edition, 1999.