

# Combining Ontologies and Document Retrieval Techniques: A Case Study for an E-Learning Scenario

Stephan Baumann, Andreas Dengel, Markus Junker, Thomas Kieninger  
German Research Center for Artificial Intelligence  
DFKI GmbH  
P.O. Box 2080, 67608 Kaiserslautern, Germany  
{baumann, dengel, junker, kieni}@dfki.de

## Abstract

*Ontologies are a perfect knowledge base for enriching standard document retrieval and classification techniques. In this paper we will describe such an approach for the application of e-learning. We choose a basic combination of ontologies, some simple heuristics and classic term, resp. document similarity measure to deliver the required functionality. Our system has a distributed architecture handling the linkage to a database-driven document management system with browser-suited output. First results are presented showing the advantages of automatic generation of queries based on ontologies.*

## 1. Overview

In the advent of large-scale intranet document management systems hidden knowledge sources are difficult to discover. For a special application such as training of newly hired employees it would be of benefit to generate some tutorial materials covering the relevant topics to such a candidate. Needless to say, from scratch only the attributes of a document management system could be used in a human task to retrieve appropriate materials. In our own working environment at the German Research Center for Artificial Intelligence we can find the main ingredients for building a seamlessly integrated system and thus fulfilling the retrieval and relevance-computing task in an automated way. We have actually a good understanding of required computer science and research know-how in our main working areas for researchers and software engineers. Furthermore, we have a proprietary database-driven document management system for the storage and retrieval of in-house and external paper-based and electronic documents. Beside these rudimentary information and knowledge sources we have developed different toolboxes for document retrieval and

classification in the past [4]. Making inferences in the ontological representation is a further major working area of our institute. In this paper we want to describe our prototypical setup which copes with the task of organizing document collections for an e-learning task in a comfortable and efficient manner. We decoupled the generic functionalities of the system explicitly from the declarative knowledge sources, namely the ontology modeling of the application domain. The attribute structure of the document management system is flexible for further adaption to new attribute value ranges as well as the introduction of further application-specific attributes. The heuristics are actually coded in a procedural way but will be described and coupled with the ontological representation in a formal way using a rule interpreter [2] in the next version. The actual prototype is therefore an ideal testbed for different future application areas. Research on different aspects describing the nature of relevance will be of special interest in future work.

## 2. Architecture

Our system is built in a typical client-server-style and contains the following components:

(1) Ontology editor and plug-ins: We use the Protégé 2000 tool for convenient design of ontologies. This tool and the underlying approach were developed at Stanford University, driven by needs in medical departments [7]. Protégé 2000 is a cross-platform tool with widespread usage in different areas. It is mainly its flexible plug-in-architecture and the powerful import and export of standard formats such as XML (eXtended Markup Language) and RDF(S) (Resource Description Framework) which makes it the tool of choice when it comes to ontology editors. Beside the definition of classes, automatically generated form editors allow the definition of the instances. We incorporated a special plug-in for convenient application-specific definition of know-how profiles. Two further plugins without user interaction are

used for access to information in (2) and (3), resp. the selection and scoring of the retrieved information in (4).

(2) Document management system: Our document management system is based on a standard relational database, here Informix/Solaris. The database tables reflect meta information such as date-of-origin, number-of-words, etc. as well as the document structure (title, author, keywords, body). Additional semantic information such as the projects involved in the generation of a document or manually applied document categories by the author are available. The unstructured full-text document content is stored as plain ASCII in the file system and is accessible via document filename identifiers.

(3) Document retrieval and classification tools: State-of-the-art document retrieval and classification approaches have recently been commercialized and successfully adapted to real-world problems. They are still lacking an in-depth ontological support, which is described in our approach. Nevertheless the basic techniques have a long-standing tradition in information retrieval. We used both an interface to the MindAccess development kit [3] as well as a TCW (Text Classification Workbench) and its submodules developed at our institute[4].

(4) Selection and heuristic ranking: When we started with work on e-learning, different simple heuristics came to mind, which is more or less common sense when coping with the task of making relevant selections for document collections. The new thing about this work especially was how to handle the aspect of document relevance by means of completeness and depth of a given topic with respect to the reader's expertise level, e.g. novice vs. expert.

(5) E-Learning browser: The collected documents, which seem appropriate for the e-learning task at hand, are assembled in a document in HTML (HyperText Markup Language). The collection is presented as a hyperlinked guided tour through the relevant documents. The contained links allow for direct reading of the document content via the standard mime-type plug-ins (Acrobat-PDF, ghostview-ps, MS-Word, etc.).

Our system is deployed in a typical heterogeneous working environment with distributed access to the components involved, namely the ontology editor, the document management system, the document retrieval and classification components and the internet browser.

### 3. Components and their functionalities

The individual components of the system will be described with regard to their functional contributions to the entire system. Chapter 4 will go into the details how the components are linked together and exchange information and knowledge sources.

#### 3.1. Protégé 2000: ontology modeling

"An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary" [6]. In our application scenario we noted as terms the concepts of required know-how in the IT-domain and for the domain of research in document analysis and knowledge management. The relations consist of two types, namely is-a and part-of relations. Is-a-relations are used to indicate specializations of concepts (e.g. ORACLE is-a Database, table recognition is-a document analysis technique) while part-of-relations denote required subconcepts for the understanding of a given concept (e.g. configuration management part-of software engineering, OCR part-of text recognition). In this way we were able to set up an initial ontology describing our department's view on know-how about computer science and research concepts at different expertise levels. The ontology is manually updated and maintained by a responsible knowledge engineer. This work is performed using an interactive ontology editor. We used the Protégé 2000 tool, which is distributed by Stanford University. The options for integrating other know-how domains via importing RDF(S) files and the open plugin architecture were the major reasons for choosing Protégé 2000. Meanwhile Protégé 2000 is an established standard tool used in different areas worldwide. We used the plug-in architecture to build our own user-interface for convenient specification of know-how instances instead of using the build-in forms-editor. Furthermore the activation of commercial and non-commercial document classification workbenches are triggered by the Protégé 2000 server as well as the access to document databases.

#### 3.2. Retrieval on full-text/structured information

The document management system which we use is a custom development and thus provides the openness and flexibility that is needed in order to fully integrate it with the other components of our environment. In its first evolution step, our DMS (document management system) server consisted of a file-system-based repository that was managed and accessed solely by server applications of the archive system. For retrieval purposes the system relied on a hybrid combination of full text index and explicit meta-information that is stored in a relational database system. The database schema for the meta-information is defined generically in a proprietary format. This format allows the specification of the type (string, integer, date etc.) of each attribute, whether it is a single- or multiple value attribute (e.g. title vs. authors) and whether it is mandatory or optional. The generic definition is compiled into an appropriate database relation scheme, HTML-forms / pages for file-uploading,

querying and modifying the archive and the programs that perform the according tasks on the server-side. To achieve a plain text version from various document-formats, we integrated numerous conversion tools. Hence the full text search can be performed on PostScript, PDF, XML/HTML, Framemaker and Word documents. After the system has been used internally, the attached keywords for manual document categorization turned out to be significant. Although a list of previously assigned keywords could be launched to easily select appropriate words for a new document, users tended to use their own spelling in an appropriate field. Consequently, specific concepts could be found in various specifications: e.g. either English or German, singular or plural. We wanted to find a technological solution for this proof of missing discipline. The solution was the integration of a domain ontology which performed a query-expansion simultaneously and transparently. This approach has been expanded to drive the e-learning application as described in this paper.

### 3.3. Associative retrieval in the Vector Space Model

We explained how to cope with everyday retrieval problems using a database-centered approach in the previous chapter. Through automatic query expansion by usage of the ontology a lot of hurdles can be resolved. However the vocabulary problem still remains by means of incompleteness in the ontology. Furthermore successful techniques for finding similar documents in an associative way are not embedded in the former approach. In order to solve these problems we rely on a toolbox implementing a standard boolean retrieval model as well as techniques based in the VSM (Vector Space Model) for IR (information retrieval). A query in the boolean retrieval model consists of a boolean combination of tests on the occurrence of specific words. For instance, the query *skill or expertise and computer* tests whether a document contains one of the words skill or expertise as well as the word computer. Extended boolean retrieval models also allow the testing of specific phrases. For our scenario, the boolean retrieval model is not sufficient. The additional functionality which we integrated is based on the VSM. In this model, documents as well as queries are represented as vectors. The dimension of the vectors indicate specific terms, the value of a vectors component indicates the number of times the respective term occurs in the document/query to be represented. Typically, raw term frequencies in vectors are replaced by term weights. These term weights also take into account how often the respective term occurs in other documents of the same collection, which is, to some extent, an indicator of its importance in the document (see, e.g., the tfidf weight, [1]). Standard document retrieval based on queries in the VSM is done by defining a similarity measure between vectors. The

most frequently used measure here is the cosine-measure, which computes the angle between two vectors. Having a vector representing the query, the documents corresponding to the most similar document vectors are returned as answer documents. The first functionality we implemented using the vector space model is the computation of document similarity. Since queries and documents in the VSM are represented as vectors, so can the similarity between vectors representing documents also be computed. Generally speaking, those documents, which share many important words, will have a high similarity. We also derived a second functionality from the VSM. While in the original VSM, documents are represented by vectors, it is also possible to represent terms as vectors. In such a vector each position corresponds to a specific document in a given document collection. The value reflects how often the term to be represented occurs within the respective document. Similar to the standard VSM, now term vectors can be compared using a similarity measure. If two term vectors are similar with respect to this measure, this indicates that these terms are likely to occur in the same context/documents.

### 3.4. HTML: Structured presentation of results

After generation of the chosen document collection based on heuristics, term and document similarities, a standard HTML document is assembled for browser access. We can use the documents in the original formats by relying on the mime-type mechanism which triggers the appropriate browser plug-in for document reading. The philosophy behind the link structure is to guide the user from general concepts to special ones. Overview papers are presented first, the user has then the choice to navigate into the nested structure of documents covering special single topics.

## 4. Mapping relevant documents to the ontology

The process of building an information retrieval, resp. question-answering system is always related to the treatment of relevance. We have found that the framework and review of literature as stated by Mizzaro [5] is quite useful to gain a clear mathematical foundation of relevance for our scenario. Therefore we will sketch the four sets of this framework, which we will refer to. We used extracts of the original definition in the following:

(1) Problem set:

$PS = \{P, IN, R, Q\}$ . The problem  $P$  is to train the newly hired candidate for his job by providing a collection of useful documents. The information need  $IN$  to solve this problem is a vague, maybe incomplete description of the required know-how concepts as a mental representation. This need has to be expressed as a request  $R$ , e.g. in natural language *I need to know everything about JAVA beans and*

table recognition techniques. Finally  $R$  must be translated into a concrete query  $Q$  in the systems language, which is difficult by means of completeness.

(2) Information set:

$IS = \{S, D, I\}$ . The surrogate  $S$  consists of the document representations stored in the system. A document  $D$  is the physical entity the user receives as a result of his retrieval.  $I$  is the non physical information entity received when reading the document.  $S$  and  $D$  are stored in our database.  $I$  is in part represented in the document's title.

(3) Components set:

$Co = 2^{\{To, Ta, C\}} - \{\}$ .  $To$  is the subject the user is interested in. In our case "the know-how of software engineers and researchers in the IT domain". The Task  $Ta$  describes the actions the user will perform with the received documents, e.g. "read, understand and expand your knowhow". The context  $C$  covers all the aspects not pertaining to topic and task, e.g. "the user is a native English speaker and prefers English documents only".

(4) Time points set:

$T = \{tp0, tn0, tr0, tq0, tq1, tr1, tq2, \dots, tqn, tf\}$ . The typical information retrieval scenario is dynamic. Queries, requests and information needs are changed during a session. At the beginning of this process the user has a problem  $P$  at time  $tp0$ , he perceives his initial  $IN$  at  $tn0$ , he expresses his request  $R$  at  $tr0$  and translates it to a query at  $tq0$ . His revisions take place until the problem is solved at time  $tf$ . We operate in our scenario in a static way ( $tp0 = tf$ ). The focus is to represent an optimal query  $Q$  with respect to an automatic generation and expansion of the request  $R$ . Optimal is meant to be a relevance which is as close as possible to the relevance for the user.

Relevance:  $Rel = PS \times IS \times Co \times T$ . Each element of  $Rel$  represents a relevance. We will explain in the subsequent sections how our technical approach tries to come up with optimal user-relevant document collections fulfilling the information need to solve the problem of "learning by documents" in an automated way. Obviously we will expand our system in the future to allow for individual relevance feedback to drive an automatic refinement of queries.

#### 4.1. Query expansion

As we have shown in chapter 3.3 term similarities are suited to expand the concept vocabulary of the underlying ontology on demand. In this way we try to map the non-formalized request  $R$  to a fully-fledged query  $Q$ . As a critical parameter we can select the maximum number of similar terms using a fixed threshold or a threshold for the term weights delivered by the IR tools. Furthermore we are generating boolean queries on the fly based on the assumption that part-of edges can be interpreted as every sub-concept should be contained in one document, while edges of the

type is-a allow for alternatives. If we start with the root of the ontology we have to expand for each son up to the maximum path length to get the most general documents. Vice versa we get specialized documents by incorporating each father of the concept under inspection up to the root. Again the maximum path length can be selected by a threshold to avoid unnecessary expansion in case of deeply nested ontologies. After having generated such a complex query the standard boolean query technique can be processed. In the formal model we build a competing query  $Q$  for  $R$  at  $tq0$ .

#### 4.2. Retrieval in the title or body passages

Using the query of 4.1. we can retrieve documents either containing the query in the title or the document's body. The database-centered system allows for access to the title contents. The queries on the document's body are processed with the information retrieval tools. Depending on the maximum path length, resp. the maximum number of similar terms one has to switch from title to body queries approximately over a threshold of 3. Titles containing more than 3 subconcepts or similar concepts are seldom. Quite often 2 or 3 concepts are explicitly stated in the title of scientific papers to express a new combination of specialization of well-known approaches. The best working approach is a mixture of title and text body retrieval checking for the occurrence of the concept under observation in the title and the father concepts up to the root in the document's body (to cover the correct context). The results are ranked according to the degree of matching subconcepts in the text's body. In the formal model these steps correspond to a different relevance ranking depending on a selection from  $S_{title}$  or  $S_{body}$ .

#### 4.3. Result expansion with document similarities

Working with document similarity allows for the extension from the initial seed documents to further semantic relevant documents. In this way we expand the initial results of our query. We discover similar documents based on the information retrieval metrics expanding vocabulary and ontological incompleteness in an implicit way. The number of hits is a good indicator for the nature of the document being very specific or general; this can be used to drive the expert-level-selection. General documents tend to generate further hits, specific documents not. In the formal model we build further queries  $Q$  for  $R$  by using the documents as queries at  $tq1$ .

#### 4.4. Heuristic result restriction

In order to cope with different expertise levels of the readers we use a simple heuristic incorporating such as-

pects. It is pragmatic in a way that it incorporates the author's common-sense understanding of preparing titles addressing a specific readership. In this way we can reduce the number of results to adapt to the user's knowledge. We use two sets of different keywords representing such aspects to refine the results: experts = {*professionals, detailed, advanced*} and rookies={*tutorial, novice, overview*}. The heuristic can be turned on/off and is incorporated automatically in the initial queries  $Q$ .

#### 4.5. Example

Let's have a look at a typical application scenario. The software development department is hiring two application programmers who should have a basic understanding of software engineering methods as well as the concept of object orientation. The programming languages of choice are object-oriented but the head of the department has no special priorities. He is willing to spend some time on initial training to learn programming languages from scratch or to go deeper into the details of a language. Having represented these constraints by annotating the ontology with expert levels (required level, candidate level) our system is able to generate different collections for two candidates having different expertise. First of all documents covering the aspects of programming languages are selected. A refinement selects documents such as a JAVA Tutorial for the task of learning JAVA from scratch. An advanced C++ programmer would be able to select also the C++ 4 Experts book. Documents containing software engineering and OO subtopics are ranked with low priority if the candidates are already aware of these concepts.

#### 5. Results

We made some initial tests based on our document collection consisting of 481 documents. These documents were collected by our department staff. The underlying ontology contains about 50 concepts of the computer science domain. In addition we specified a second domain containing 30 concepts about research in the area of information and knowledge management. We will give a short qualitative evaluation of these results for two typical profiles. For a senior researcher in the document analysis department being expert in table recognition and novice in concepts about workflow the system delivered the following documents: Exact matches in title (*Lexicon-Driven Information Extraction from a Document Analysis View, On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy, A Graph-Based Table Recognition System*), exact matches in title and heuristic (*An Overview of Workflow Management, Workflow Management in the Internet Age*) and by term expansion (*Workflow Management*

*Coalition - Terminology and Glossary, Workflow Management Coalition:Workflow Standards*). For a junior software engineer having basic java expertise the following documents have been selected: Exact match in title and heuristic (*Brewing Java: A Tutorial, Java GUI*), a partial title match and term expansion in the text (*Anwendungsentwicklung am Beispiel der JFC*), similar terms in text (*JavaX - An Approachable Examination Of Java, JavaBeans, JavaScript And All The Related Java Technology*) and pure document similarity (*Thinking in Java, 2nd Edition, Release 7*). A broader scope of results will be available at the website of our institute ([www.dfki.de](http://www.dfki.de)).

#### 6. Conclusion and future work

The basic algorithms and tools developed in the *Profiler* project will be expanded to cope with automated support for human resource management. We plan to implement some web crawlers, which can be used to collect information from academic and company sites. The job offering pages of large IT companies could be used to fill the required expertise levels in our ontology about IT concepts. Vice versa curriculum vitae are often available at academic sites and could serve as an input to look for appropriate candidates. The scope of our future work will remain focused to our limited application. Related work [8] covers a much broader range in basic research and applications. Nevertheless it justifies the practical impact of bridging subsymbolic approaches, i.e. IR methods with knowledge representation issues, i.e. ontologies.

#### References

- [1] R. Baeza-Yates et al. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [2] E. J. Friedman-Hill. *Pjess java expert system shell*. <http://herzberg.ca.sandia.gov/jess/>, 1999.
- [3] Insiders-Information-Management-AG. *Manual mindaccess 2.6*. <http://www.im-insiders.de>, 2001.
- [4] M. Junker et al. Text categorization using learned document features. *Hybrid Methods in Pattern Recognition*, World Scientific Publishing 2002.
- [5] S. Mizzaro. Relevance the whole (hi)story. *Journal of the American Society for Information Science*, 48(9):810-832, 1997.
- [6] R. Neches et al. Enabling technology for knowledge sharing. *AI Magazine*, 12 (3):36-56, 1991.
- [7] N. Noy et al. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 16(2):60-71, 2001.
- [8] OnToKnowledge-Consortium. *Project homepage*. <http://www.ontoknowledge.org>, 2002.