Documents meet Databases: A System for Intranet Search

Christoph Mangold Holger Schwarz

Universität Stuttgart, IPVS
Universitätsstr. 38, D - 70569 Stuttgart
firstname.lastname@ipvs.uni-stuttgart.de

Abstract

In enterprise intranets, information is encoded in documents and databases. Logically, the information in both worlds is tightly connected, however, on the system level there is usually a large gap. In this paper, we propose a system to retrieve documents in the enterprise intranet. The system is an extension to common text search. It does not only consider the content of documents but also it exploits the enterprise databases to determine the documents' context.

1 Introduction

In enterprise intranets, information is encoded in documents and databases. Both kinds of information are crucial for the success of the enterprise. Both kinds of information are tightly connected as regards their content. However, both kinds of information are usually not integrated on the system level. For a discussion of this issue see, e.g. [1, 5]. In the enterprise, the missing links on the system level need to be compensated by employees. I.e., the missing links have to be established in the heads of the employees, which is a long-lasting and expensive process.

This paper introduces our system that connects documents with databases for the benefit of document retrieval in the enterprise intranet. Standard text search engines retrieve documents based on *content*. Our search engine additionally exploits document *context* that is retrieved from the enterprise's databases.

To model documents and their context we use a graph data model, called ContextGraph. We map databases to the ContextGraph such that graph nodes represent relational tuples, and values. Attribute and foreign-key relationships map to edges in the graph. Documents are represented by special document nodes. To determine the contexts of documents in the ContextGraph we use a specialized shortest paths algorithm.

International Conference on Management of Data
COMAD 2005b, Hyderabad, India, December 20–22, 2005

In Figure 1, we give an impression of a ContextGraph. The example shows the document Doc11 that is a Work Instruction. In the enterprise, the document is listed in a document management system (DMS). With its URL, the content of the document can be found in the file system (FS). The DMS not only contains the URL but also a link to the technician Bob Blue who maintains the work instruction. From the enterprise resource planning system (ERPS) we get that Bob Blue works in Team18 together with Ricky Red.

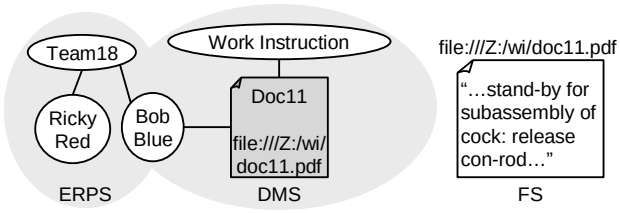

Figure 1: Small fraction of a ContextGraph.

Consider the following simple scenario: For a new product, some of the work instructions that are maintained by Ricky Red's team need to be adapted. In particular, the release of the con-rod has to be delayed. Hence, the responsible employee feeds the query "release con-rod" AND "Ricky Red" to the search engine. However, since Ricky Red is the team leader, he does not maintain a single work instruction. In this case, a standard text-based search engine yields results that *either* contain "release con-rod" *or* "Ricky Red". This behavior is of no use for the employee since the result set is potentially large. He might skim through the result set to filter relevant documents manually or alternatively investigate possible query modifications. The latter may involve expert knowledge from co-workers or information from databases.

Our system helps the employee to directly access the desired information. It is based on the notion that in this situation it is desirable to retrieve, e.g. Doc11 that

is maintained by a co-worker of Ricky Red and rank it as highly relevant. To achieve this, our approach uses information from the enterprise's ERPS.

In the rest of the paper we give an overview of the system in Section 2 and summarize in Section 3.

2 System Overview

In this section we discuss the ContextGraph and how we build it from the enterprise's databases. We give a brief overview of our ranking strategy, sketch the system architecture and finally present the user interface of the system.

2.1 The ContextGraph

As we motivated in the Introduction, we use enterprise's databases to determine the context of documents in the intranet. To achieve this, we derive the ContextGraph (CG) from the enterprise's databases (DB) using the following simple mapping: Each tuple in the DB becomes a node in the CG. Each attribute value belonging to the tuple also becomes a node that is connected to the tuple node. Tuple nodes are mutually interconnected according to foreign-key relationships in the DB. M:N-relationships in the DB are denormalized and represented as graph edges. A documents is represented in CG by a special node that contains the document's URL, only. We detail on the ContextGraph in [3].

To exploit the information represented in the ContextGraph, each document needs to be aware of its context. For two nodes $n_1, n_2 \in$ CG we define the *semantic distance* $\text{dist}(n_1, n_2)$ to be the length of the shortest path from n_1 to n_2. Our system permits to assign different semantic distances to different types of edges, but this is beyond the scope of the paper.

Let *cRange* be a positive value that denotes the context range. Then, we define the *context* of a node $n \in$ CG as the set of nodes that are reachable from n within distance *cRange*:

$$\text{Context}(n) = \{v \in \text{CG} \mid \text{dist}(n, v) \leqq cRange\}.$$

Likewise, the context of a document d is the context of the node n_d that represents d.

2.2 Ranking

The ranking of search results is a critical issue for search engines. The result quality of a search engine is considered good if and only if for a given query the most relevant documents appear at the top of the result list. Hence, a good ranking function distinguishes those documents that are most likely to be relevant for the query.

In general, ranking measures can be divided into measures for document importance and document relevance. Document importance is a query independent measure. In a scenario where users prefer new documents, the last change date is a good indicator for document importance. In contrast, document relevance measures the appropriateness of documents concerning a given query. In the common full text search scenario, document relevance is determined based on the content of the document.

Our approach extends the relevance measure to not only consider the content of a document but also its context. I.e., for a given search term, we do not only check its appearance inside the document, but also in the document's context. If a search term appears in the contexts of two documents it is a straightforward idea to boost the ranking of the document with smaller semantic distance to the search term.

We realize this by storing terms with different semantic distances in different parts of the index. When retrieving documents, we consider the distance of found search terms for the ranking of results.

2.3 System Architecture

In Figure 2, we give an overview of our system, for more details see [4]. To enable scalability, we chose an index based architecture. At indexing time, the indexer analyzes databases and documents and generates a data structure – the index – that can be searched efficiently at retrieval time.

At the bottom of Figure 2, the RDB Import component inputs data from relational databases and generates a ContextGraph. The Shortest Path component determines document contexts and graph distances by running a specialized shortest path algorithm on the ContextGraph [3]. For each document, the Contextualizer component uses the Document Import component to retrieve the document text from the respective URL. Then, it combines the document text with the results of the Shortest Paths component, i.e., the document context, and passes both to the Index Builder component. Finally, the Index Builder component is responsible to organize this information such that it can be stored in the index data structure.

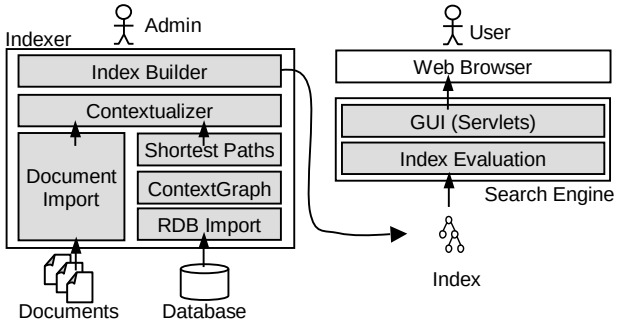

Figure 2: System architecture.

At query time, the user inputs a keyword query into a search form in his web browser. The browser sends the user query via web server and servlet engine to the Index Evaluation component. The latter is responsible to rewrite the user query according to the following constraints:

- Index structure. The query needs to fit the index structure that is determined by the Index Builder component.
- Ranking. Document relevance is partially computed at query time and needs to be considered when rewriting the query.
- User parameters. The user may set some parameters at query time that need to be reflected, see Section 2.4.

Subsequently, the Index Evaluation component accesses the index and retrieves relevant documents that are finally returned to the user.

In our implementation we use the Lucene search engine framework [2] as part of the Index Builder and as part of the Index Evaluation component.

2.4 User Interface

The user interface of our system is shown in Figure 3. The input field (a) takes user keywords. Right below, the (advanced) user may adjust the following parameters (b):

- Number of results. Results retrieved per query.
- Context range. The user can adjust the value of cRange, however cRange is bound by the value that has been chosen at indexing time. If the user sets cRange= 0 the search engine disregards context information and operates like a standard text search engine.
- Not range. Where context-based search is trivial for search terms connected with the boolean operators AND and OR, the usage of NOT poses the problem that too many documents might be pruned from the result set. We solve this problem by introducing the separate parameter NotRange that overrides the cRange parameter for negated query terms.
- Context influence. Determines the influence of context information to the ranking of results. The user may choose that results are computed based on content only, on context only, or on a combination of both.

The area in Figure 3 (c) shows the query results. It comprises three columns: The (colored) boxes in the first column give hints on the ranking of results. Here, dark boxes indicate relevant documents, light boxes represent lower ranks. Column two shows document links and column three contains a set of buttons that transfer the respective document with its context to the visual ContextGraph browser that is shown in Figure 3 (d). In the browser, the user may explore the ContextGraph interactively. It supports the expansion of the direct neighborhood of nodes. Furthermore, the user may apply edge filters to focus on certain information and various layout filters.

To elucidate the ranking value of search results, a left-click on the ranking box in the first column of Figure 3 (c) opens a window that explains how the value has been computed. Figure 4 shows a sample window. In this example the query is "Pleuel lösen" AND XLK1 (where "Pleuel lösen" is german for "release con-rod"). The ranking explanation for the found document shows that the document contains "Pleuel lösen" in its content (upper part) and XLK1 in both, the context and the content of the document (lower part).

3 Summary

In this paper we introduced our approach to exploit database information for the benefit of intranet document retrieval. We presented a search engine that considers not only unstructured document content but also structured information from the enterprise's databases. We introduced the ContextGraph as an abstraction of databases and explained our ranking strategy. We gave an overview of the system architecture and finally introduced the user interface.

References

[1] S. Chakrabarti. Breaking through the syntax barrier: Searching with entities and relations. In *ECML 2004*, volume 3201 of *LNCS*, page 9. Springer-Verlag, 2004.

[2] O. Gospodnetić and E. Hatcher. *Lucene in Action*. Manning Publications, 12 2005.

[3] C. Mangold, H. Schwarz, and B. Mitschang. Symbiosis in the intranet: How document retrieval benefits from database information. In *COMAD 2006*, 2006.

[4] C. Mangold, H. Schwarz, and B. Mitschang. u38: A framework for database-supported enterprise document-retrieval. In *IDEAS'06*, 2006.

[5] A. Somani, D. Choy, and J. C. Kleewein. Bringing together content and data management systems: Challenges and opportunities. *IBM Systems Journal*, 41(4):686–696, 2002.

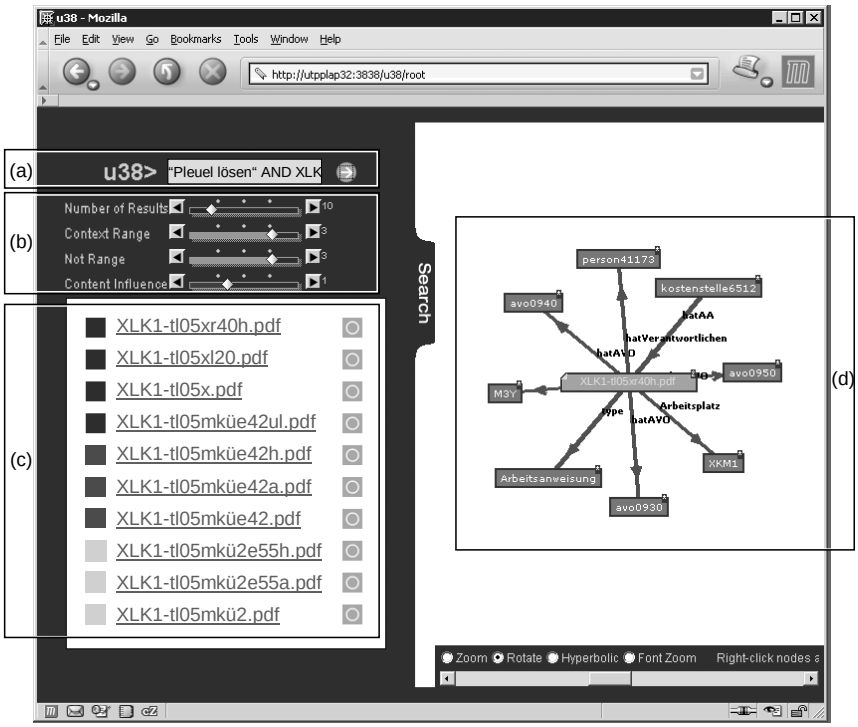

Figure 3: User Interface. (a) Input field. (b) Parameter adjustment. (c) Result panel. (d) Visual browser.

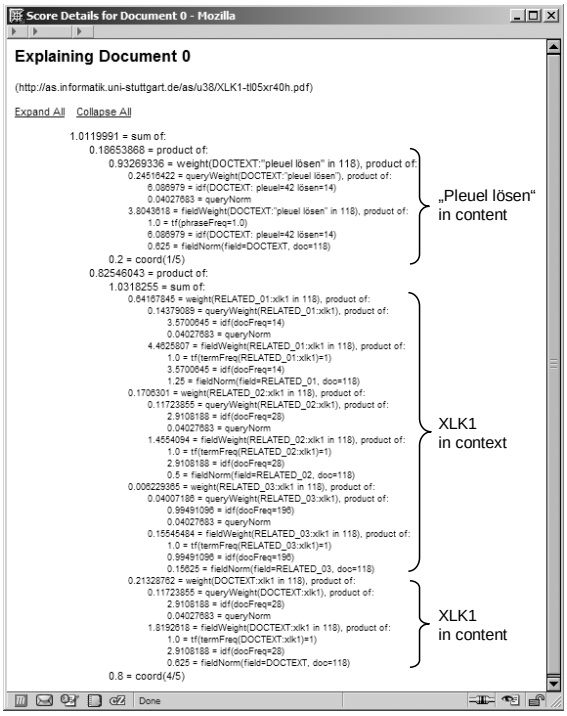

Figure 4: Sample explanation for result ranking.