Information Retrieval, 8, 521–545, 2005

Semantic Similarity Search on Semistructured Data with the XXL Search Engine

RALF SCHENKEL schenkel@mpi-sb.mpg.de
ANJA THEOBALD atb@mpi-sb.mpg.de
GERHARD WEIKUM weikum@mpi-sb.mpg.de
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. Query languages for XML such as XPath or XQuery support Boolean retrieval: a query result is a (possibly restructured) subset of XML elements or entire documents that satisfy the search conditions of the query. This search paradigm works for highly schematic XML data collections such as electronic catalogs. However, for searching information in open environments such as the Web or intranets of large corporations, ranked retrieval is more appropriate: a query result is a ranked list of XML elements in descending order of (estimated) relevance. Web search engines, which are based on the ranked retrieval paradigm, do, however, not consider the additional information and rich annotations provided by the structure of XML documents and their element names.

This article presents the XXL search engine that supports relevance ranking on XML data. XXL is particularly geared for path queries with wildcards that can span multiple XML collections and contain both exact-match as well as semantic-similarity search conditions. In addition, ontological information and suitable index structures are used to improve the search efficiency and effectiveness. XXL is fully implemented as a suite of Java classes and servlets. Experiments in the context of the INEX benchmark demonstrate the efficiency of the XXL search engine and underline its effectiveness for ranked retrieval.

Keywords: XML retrieval, semistructured data, ranked retrieval

1. Introduction

1.1. Motivation

XML is becoming the standard for integrating and exchanging data over the Internet and within intranets, covering the complete spectrum from largely unstructured, ad hoc documents to highly structured, schematic data. A number of XML query languages have been proposed, such as XPath, XML-QL, or the recently announced W3C standard XQuery. These languages combine SQL-style logical conditions over element names, content, and attributes with regular-expression pattern matching along entire paths of elements. The result of a query is a set of paths or subgraphs from a given data graph that represents an XML document collection; in information retrieval (IR) terminology this is called Boolean Retrieval.

This search paradigm makes sense for queries on largely schematic XML data such as electronic product catalogs or bibliographies. It is of very limited value, however, for searching highly heterogeneous XML document collections where either data comes from many different information sources without a global schema or most documents have an ad

hoc schema or DTD with element names and substructures that occur only in a single or a few documents. The latter kind of environment is typical for document management in large intranets, scientific data repositories such as gene expression data collections and catalogs of protein structures, and, of course, also for the Web. For example, a bank has a huge number of truly semistructured documents, probably much larger in total size than the production data held in (object-) relational databases; these include briefing material and the minutes of meetings, customer-related memos, reports from analysts, financial and business news articles, and so on. Here, the variance and resulting inaccuracies in the document structures, vocabulary, and document content dictate ranked retrieval as the only meaningful search paradigm.

The result of a query should be a list of potentially relevant XML documents, elements, or subgraphs from the XML data graph, in descending order of estimated relevance. This is exactly the rationale of today's Web search engines, which are also widely used for intranet search, but this technology does not at all consider the rich structure and semantic annotations provided by XML data. Rather state-of-the-art IR systems restrict themselves to term-frequency-based relevance estimation (Baeza-Yates and Riberto-Neto 1999) and/or link-based authority ranking (Brin and Page 1998, Kleinberg 1999).

This article presents a query language, coined XXL (for Flexible XML Search Language), and the prototype implementation of the XXL search engine, as steps towards more powerful XML querying that reconciles the more schematic style of logical search conditions and pattern matching with IR-style relevance ranking. Aiming at simplicity and with focus on simple but widely usable search templates, our approach has adopted a core of essential features from XQuery-style languages and has enhanced it with a similarity operator, denoted $\sim$, on element names and contents. The evaluation of similarity conditions is based on a quantified ontology for element names and contents in combination with term-frequency-based IR-style estimations for element contents. The assessments of "local" similarity tests are combined into "global" relevance rankings using simple probabilistic arguments.

Even though some of our results have been previously published (Theobald and Weikum 2000, 2002a, Schenkel et al. 2003), this article is the first that completely covers the XXL Search Engine in all its aspects, including the underlying ontological model, query evaluation, and implementation issues. It further presents new experimental results with the INEX benchmark that were not published outside the INEX community before.

1.2. XML data model

In our model, a collection of XML documents is represented as a directed graph where the nodes represent elements, attributes and their values. For identification, each node is assigned a unique ID, the `oid`. There is an directed edge from a node x to a node y if

- y is a subelement of x,
- y is an attribute of x,
- y contains the content of element x, or
- y contains the value of attribute x.

Additionally, we model a link from one element to another (this can be an ID/IDREF link, an XLink or an XPointer) by adding two directed edges in opposite directions between the corresponding nodes; while this model covers simple and extended XLinks, our system currently supports only simple XLinks. We call the resulting graph the *XML data graph* for the collection.

Figure 1 shows the XML data graph for a collection of two XML documents similar to those in the INEX collection: a journal document with an XLink pointing to an article document. Each node that contains an element or attribute name is called an *n-node* (shown as normal nodes in figure 1), and each node that contains an element content or attribute value is called a *c-node* (dashed nodes in figure 1). To represent mixed content, we need a local order of the child nodes of a given element. Figure 1 shows an example for a sentence that is distributed over several *c*-nodes.

1.3. Outline

The rest of the article is structured as follows. We discuss related work in Section 2. In Section 3 we introduce our ontology model. Section 4 presents our query language XXL, and Section 5 describes architecture and core components of our search engine. Section 6 presents details of the query evaluation in the XXL Search Engine. Finally, Section 7 shows the effectiveness of our search engine with example results from the INEX benchmark.

2. Related work

Ranked retrieval on XML data has been quite popular in recent years. The first approaches carried over keyword-based search from the Web to XML data and did not provide structural constraints. Among the more recent retrieval engines of this kind are XRANK (Guo et al. 2003) and XSearch (Cohen et al. 2003). XRANK uses a two-dimensional proximity measure and a pagerank-like authority ranking to increase result quality for keyword-based queries; XSearch allows to restrict a keyword-based query to elements with certain tag names, but does not support more complex structural constraints.

The majority of recent approaches support both keyword-based and structural constraints. Extending existing XML query languages such as XML-QL (Deutsch et al. 1998) or XQuery (Boag et al. 2002) with text search methods has been suggested by Chinenyanga and Kushmerick (2001) Fuhr and Großjohann (2001) Hayashi et al. (2000) Theobald and Weikum (2000). The simultaneously developed languages XIRQL (Fuhr and Großjohann 2001) and XXL (Theobald and Weikum 2000) (the latter is our own approach) have been designed to support ranked retrieval. A restricted approach along these lines is Hayashi et al. (2000), which assumes advance knowledge of the document structure and provides similarity search only on element contents, not on element names. The ELIXIR system (Chinenyanga and Kushmerick 2001, Chinenyanga and Kushmerick 2002) provides an extension of XML-QL by content-based conditions that are evaluated using the WHIRL system (Cohen 1999). The TeXQuery project (Amer-Yahia et al. 2004), extends XQuery with extensive keyword-based search conditions on the content of XML elements. To our

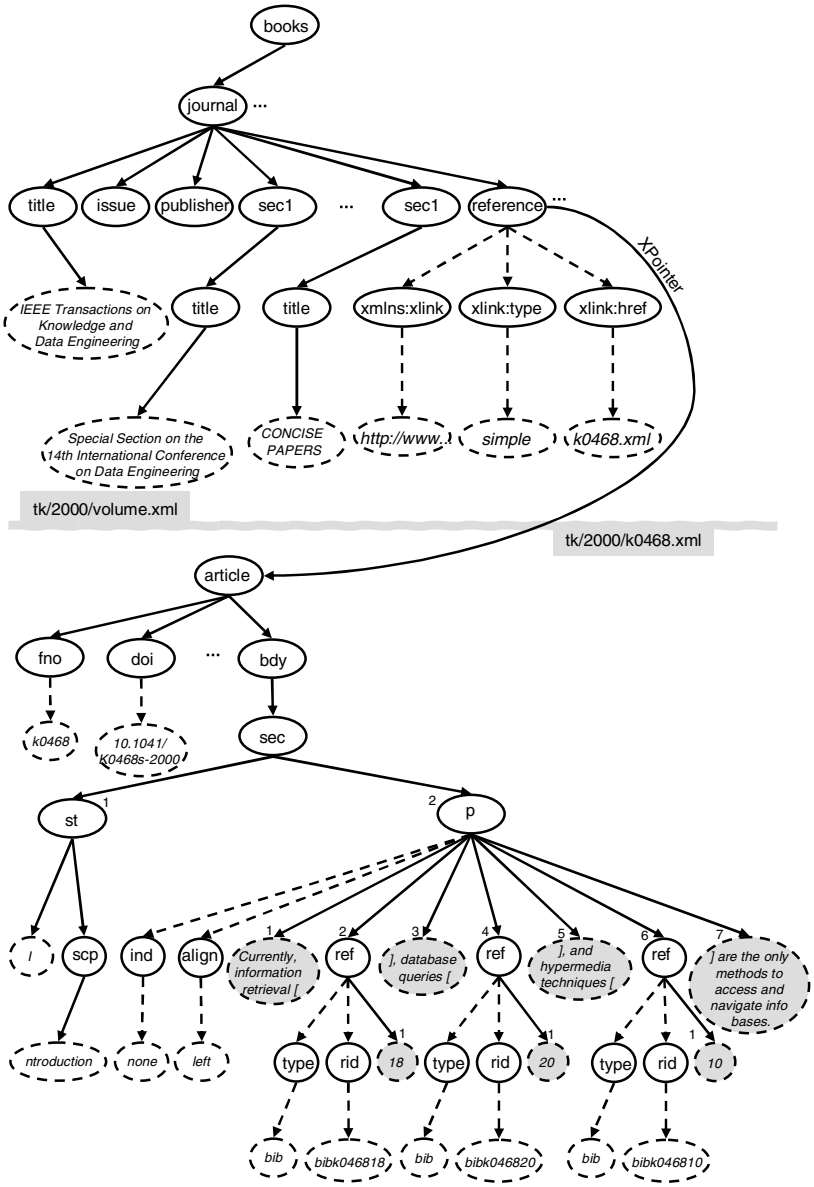

Figure 1. XML data graph.

knowledge, none of the above approaches uses ontological information in their similarity metrics.

Structural similarity of path patterns within XML documents has been investigated by some authors. Schlieder and Meuss (2000) extend the vector space model to term vectors

consisting of structured terms by generalizing tf*idf-based term weights to structured terms. The language ApproXQL (Schlieder 2002) considers structural similarity between query graphs and XML data graphs based on graph transformation and matching. Guha et al. (2002) apply a tree distance measure to implement approximate joins on XML data. Amer-Yahia et al. (2002) consider a broader set of transformation rules, including ontological-based generalization of tag names (but without discussing the problem of quantifying ontological relationships), edge generalization, and subtree promotion. However, in all these approaches, ranked results are computed based on the cost of the graph transformations; the general problem of embedding a query tree into a data tree is NP-complete. In contrast, our XXL search engine aims to exploit similarity conditions on element names and contents, which can be evaluated much more efficiently.

Index structures for the support of XML path queries have been intensively studied in the literature. The most efficient path indexes use encoding schemes for trees (e.g., Grust 2002, Zezula et al. 2003), but are inherently limited to tree-structured XML without links. Recently, index structures have been published that support XLinks and XPointers, e.g. APEX (Chung et al. 2002), the Index Definition Scheme (Kaushik et al. 2002), the $D(k)$ Index (Qun et al. 2003), HOPI (Schenkel et al. 2004), and the FliX framework (Schenkel 2004). All these approaches don't provide explicit support for ranked retrieval. Our XXL search engine can make use of all of them.

There are many recent papers on XML query processing and optimization. A special focus has been on the efficient evaluation of query twig patterns (see, e.g., Bruno et al. 2002, Jiang et al. 2003, Choi et al. 2003, Kaushik et al. 2004). The latter approach integrates a variant of Fagin's Threshold algorithm to return only the most relevant results. Another important aspect in the literature have been selectivity estimation for queries on XML (see, e.g., Aboulnaga et al. 2001, Chen et al. 2001, Wu et al. 2002) and statistical summaries of XML data (Polyzotis and Garofalakis 2002a, 2002b). None of these approaches considers similarity search.

The interest in ontologies has been recently revived with the recent discussion about the "Semantic Web". In contrast to the extremely ambitious early AI approaches toward building universal ontologies (see, e.g., Lenat and Guha 1990, Russel and Norvig 1995), more recent proposals are aiming at domain- or user-specific ontologies and are based on more tractable logics (see, e.g., Horrocks 2002, Staab et al. 2000). However, these publications do not consider the quantification of relationships which was first introduced by Rubenstein and Goodenough (1965) as a result of a manual annotation process. The first automatic approaches for term similarity concentrated on exploiting statistical correlations between terms (Lesk 1969), especially for the problem of query expansion (Qiu and Frei 1993, 1995). Early work on similarity measures for ontologies concentrated on the graph structure of the ontology (Rada et al. 1989, Sussna 1993, Leacock and Chodrow 1998, Wu and Palmer 1994, Hirst and St-Onge 1998, Richardson et al. 1994, Agirre and Rigau 1996, Lewis 2002). Since the mid of the 90ies, researchers started connecting both worlds, yielding similarity measures that take into account both the graph structure as well as statistics on a large corpus (Resnik 1995, Jiang and Conrath 1997, Lin 1998, Resnik 1999). A detailed comparison of similarity measures for WordNet can be found in Budanitsky and Hirst (2001), McHale (1998) and Jamasz and Szpankowicz (2003) compare measures

based on WordNet with similar measures for Roget's Thesaurus (Jamasz and Szpankowicz 2001).

3. Ontology-based similarity

Ontologies have been used as a means for storing and retrieving knowledge about the words used in natural language and relations between them. This section presents an overview of our ontological model with its quantified relationships; more details can be found in Schenkel et al. (2003).

In our approach we consider an ontological term t as a pair $t = (w, s)$ where w is a word over an alphabet Σ and s is the word sense (short: sense) of w, e.g.

```
t1 = (star, a celestial body of hot gases)
t2 = (heavenly body, a celestial body of hot gases)
t3 = (star, a plane figure with 5 or more points)
```

The *synset* syn(s) for a sense s is the set of all words with that sense, and we call the pair (syn(s), s) a *concept*. A concept collects all possible words for a sense. In order to determine which concepts are related, we introduce semantic relationships between concepts that are derived from common sense. We say that a concept c is a *hypernym* (*hyponym*) of a concept c' if the sense of c is more general (more specific) than the sense of c'. We also consider holonyms and meronyms, i.e., c is a *holonym* (*meronym*) of c' if c' means something that is a part of something meant by c (vice versa for meronyms).

Based on these definitions we now define the ontology graph $O = (V_O, E_O)$ which is a data structure to represent concepts and relationships between them. This graph has concepts as nodes and an edge between two concepts whenever there is a semantic relationship between them. In addition, we label each edge with a weight and the type of the underlying relationship. The weight expresses the semantic similarity of two connected concepts.

To fill our ontology with concepts and relationships we use the voluminous electronical thesaurus WordNet (Fellbaum 1998) as backbone. WordNet organizes words in synsets (i.e., sets of words with the same sense) and presents relationships between synsets without any quantification.

For quantification of relationships we consider frequency-based correlations of concepts using large web crawls. In our approach, we compute the similarity of two concepts using correlation coefficients from statistics, e.g. the Dice or Overlap coefficient (Manning and Schuetze 1999). Figure 2 shows an excerpt of an example ontology graph around the first sense for the word "star".

For two arbitrary nodes u and v that are connected by a path $p = \langle u = n_0 \ldots n_k = v \rangle$, we define the similarity $\mathrm{sim}_p(u, v)$ of the start node u and the end node v along this path to be the product of the weights of the edges on the path:

$$\mathrm{sim}_p(u, v) = \prod_{i=1}^{\mathrm{length}(p)-1} \mathrm{weight}(\langle n_i, n_{i+1} \rangle)$$

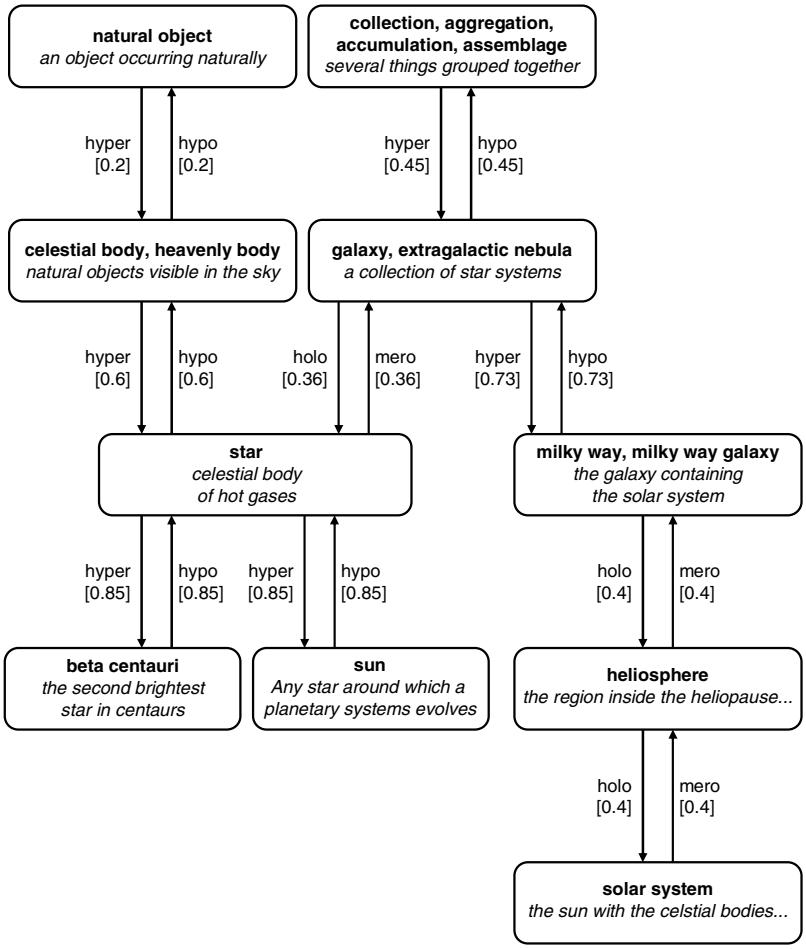

Figure 2. Excerpt of an ontology graph O with labeled edges.

where $weight(\langle n_i, n_{i+1}\rangle)$ denotes the weight of the edge $e = (n_i, n_{i+1})$. The rationale for this formula is that the length of a path has direct influence on the similarity score. We may additionally restrict the type of edges that are allowed on the path (e.g., allow only hyponym edges to have positive similarity only to more specific concepts). The similarity $\text{sim}(u, v)$ of two nodes u and v is then defined as the maximal similarity along any path between u and v:

$$\text{sim}(u, v) = \max\{\text{sim}_p(u, v) \mid \ p \text{ path from } u \text{ to } v\}$$

However, the shortest path (the path with the smallest number of edges) need not always be the path with the highest similarity. Thus, we need an algorithm that takes into account all possible paths between two given concepts, calculates the similarity scores for all paths, and

chooses the maximum of the scores for the similarity of these concepts. This is a variant of the single-source shortest path problem in a directed, weighted graph. A good algorithm to find similar concepts to a given concept and their similarity scores is a variant of Dijkstra's algorihm (Cormen et al. 2001) that takes into account that we multiply the edge weights on the path and search for the path with the maximal weight instead of minimal weight.

Furthermore, as words may have more than one sense, it is a priori not clear in which sense a word is used in a query or in a document. To find semantically similar words, it is fundamental to disambiguate the word, i.e., to map it to the concept that corresponds to its current sense. In our work we compute the correlation of a context of a given word and the context of a potential appropriate concept from the ontology (e.g., the concepts that contain the word). Here, the context of a word are other words in the proximity of the word in the query or document, and the context of a concept is built from the words of closely related nodes of the concept in the ontologies (like hypernyms and hyponyms). After removing stopwords, we determine the most similar concept out of the candidate concepts by computing the correlation of the contexts using one of the correlation measures discusses above or (as we do in our implementation) by computing the cosine similarity of the corresponding term vectors.

Note that the information in a query may not always be sufficient for a successful disambiguation, for example when a user submits only a single keyword. In such a situation, it may be helpful to ask the user select the "right" concept out of the candidates, maybe by presenting her all possible candidates together with a subset of results for the query with that concept. However, we have not yet implemented this solution in the XXL search engine (but the COMPASS search engine (Graupmann et al. 2004) delevoped in our group provides a similar user interaction for disambiguation).

4. The Flexible XML Query Language XXL

The Flexible XML Search Language XXL (Theobald and Weikum 2000, Theobald and Weikum 2002a) has been designed to allow SQL-style queries on XML data. We have adopted several concepts from XML-QL (Deutsch et al. 1998), XQuery (Boag et al. 2002) and similar languages as the core, with certain simplifications and resulting restrictions, and have added capabilities for ranked retrieval and ontological similarity search. As an example for an XXL query, consider the following query that searches for publications about both information retrieval and databases:

```
SELECT $T              // output of the XXL query
FROM   INDEX           // search space
WHERE  ~article AS $A  // search condition
  AND  $A/~title AS $T
  AND  $A/#/~section ~ "IR & DB"
```

The `SELECT` clause of an XXL query specifies the output of the query: all bindings of a set of element variables. The `FROM` clause defines the search space, which can be a set of URLs or the index structure that is maintained by the XXL engine. The `WHERE` clause specifies

the search condition; it consists of the logical conjunction of *path expressions*, where a path expression is a regular expression over *elementary conditions* and an elementary condition refers to the name or content of a single element or attribute. Regular expressions are formed using standard operators like '/' for concatenation, '|' for union, and '*' for the Kleene star. The operator '%' is a wildcard for any single element, and '#' (which is equivalent to `(%)*`) stands for an arbitrary path of elements. Each path expression can be followed by the keyword `AS` and a variable name that binds the end node of a qualifying path (i.e., the last element on the path) to the variable, that can be used later on within path expressions, with the meaning that its bound value is substituted in the expression.

In contrast to other XML query languages we introduce a new operator '∼' to express semantic similarity search conditions on XML element (or attribute) names as well as on XML element (or attribute) contents. As an example for this, consider the elementary condition ∼`article` in the example query. It is satisfied by elements with the name `article`, but also by other elements with names that are semantically similar to `article`, like `book`, `report`, and so on. The relevance for such elements corresponds to the semantic similarity of their tag name to `article` as reported by the ontology.

The result of an XXL query is a subgraph of the XML data graph, where the nodes are annotated with local relevance probabilities called similarity scores for the elementary search conditions given by the query. These similarity scores are combined into a global similarity score for expressing the relevance of the entire result graph. See Section 6 for more details on this process.

5. The XXL search engine

5.1. Architecture

The XXL Search Engine (Theobald and Weikum 2002b) is a client-server system with a Java-based GUI. It uses an Oracle 9i database for storing XML documents (see Secion 5.3). The architecture of our search engine is depicted in figure 3, it consists of the following core components:

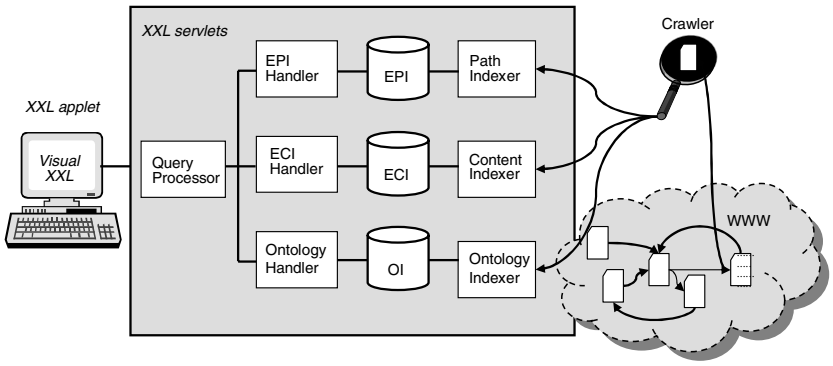

Figure 3. Architecture of the XXL search engine.

- The *Crawler*: This is a standalone, multithreaded Java application that imports XML and HTML files into the search engine. It can either operate on a local filesystem or on the Web, in both cases following links found within the documents. The documents are decomposed into database tables (see Section 5.3 for details). The crawler uses a simple timestamp-based heuristics to decide if a document that was encountered before has changed in the meantime. Newly found or updated documents are automatically fed to the index subsystems.
- The *Element Content Index (ECI)*: The ECI contains all terms that occur in the content of elements and attributes, together with their occurrences in documents; it corresponds to a standard text index with the units of indexing being elements rather than complete documents.
- The *Element Path Index (EPI)*: The EPI contains the relevant information for evaluating simple path expressions that consist of the concatenation of one or more element names and path wildcards #.
- The *Ontology Index (OI)*: The OI implements the ontology graph presented in Section 3.
- The *Query Processor*: This component makes use of the three indexes to efficiently and effectively evaluate XXL queries.
- The *Visual XXL Interface*: This Java applet provides a user friendly, browser-based interface for formulating XXL queries. Additionally, it presents the XML fragments that are the result of a query.

More details on the index structures are provided in Section 5.2. Important aspects of the query processor are highlighted in Section 6.

5.2. Index structures

The XXL Search Engine provides appropriate index structures, namely the element path index (EPI), the element content index (ECI), and the ontology index (OI), that support the evaluation process.

5.2.1. The ontology index The OI supports finding words that are semantically related to a given word, using the techniques presented in Section 3. It delivers related words in descending order of similarity so that they can be consumed as needed by the query evaluation. Additionally, the search can be limited to certain edge types: For finding similar tag names, all edge types may be used, whereas for finding similar content, search may be restricted to hyponym edges to avoid topic drift. Finally, the OI provides methods for disambiguation, i.e., find the current sense of a word among all candidate senses, given the context of the word.

5.2.2. The element content index The ECI supports the evaluation of complex logical search conditions using an inverted file and a B+–tree over element names. Given an atomic formula, the ECI returns elements whose content is relevant with respect to that atomic formula together with a relevance score. The current implementation makes use of Oracle's fulltext search engine *Oracle Text* (Oracle Corp. 2004). It provides, among

other features, index support for database columns with textual content and automatic stemming and stopword removal. For a given combination of terms, Oracle Text returns rows that contain all of the terms, in descending order of a tf/idf-based score. Since in our database schema, each row (which is a "document" for Oracle Text) corresponds to the content of a single element, this score is in fact a tf/ief score, i.e., includes the inverse *element* frequency of the terms. This is much more appropriate to XML documents with their explicit structure than simply considering the frequency of a term in a complete XML document (as the standard tf/idf measure does). Note that even though Oracle Text provides a build-in ontology, we make use of the OI to expand queries in order to have full control over the generated score.

5.2.3. The element path index The EPI provides efficient methods to find children, parents, descendants and ancestors of a given node, to test if two arbitrary nodes are connected, and to determine the distance of two nodes. It provides a general Java interface that in principle allows all existing path indexes to be applied.

When the XML data graph forms a tree, we use the well-known pre- and postorder scheme by Grust (2002) and Grust and van Keulen (2003). This path index computes the *pre order* $pre(e)$ and the *post order* $post(e)$ for each element e of a single XML document without links, by traversing the document in depth-first order. All XPath axes can be evaluated using these numbers, e.g., there is a path from x to y iff $pre(x) < pre(y)$ and $post(x) > post(y)$.

However, if the document collection contains links, this scheme can no longer be applied as the XML data graph no longer forms a tree. For such settings that occur frequently with documents from the Web, the XXL Search Engine provides the *HOPI* index (Schenkel et al. 2004) that utilizes the concept of a 2-hop cover of a graph. This is a compact representation of connections in the graph developed by Cohen et al. (2002). It maintains, for each node v of the graph, two sets $L_{\text{in}}(v)$ and $L_{\text{out}}(v)$ which contain appropriately chosen subsets of the transitive predecessors and successors of v. For each connection (u, v) in the XML data graph G, we choose a node w on a path from u to v as a *center node* and add w to $L_{\text{out}}(u)$ and to $L_{\text{in}}(v)$. We can efficiently test if two nodes u and v are connected by checking $L_{\text{out}}(u)$ and $L_{\text{in}}(v)$: there is a path from u to v iff $L_{\text{out}}(u) \cap L_{\text{in}}(v) \neq \emptyset$. The path from u to v can be separated into a first hop from u to some $w \in L_{\text{out}}(u) \cap L_{\text{in}}(v)$ and a second hop from w to v, hence the name of the method. If we additionally store the distances of u to the nodes in $L_{\text{in}}(u)$ and $L_{\text{out}}(u)$, we can use this information to compute the distance of two nodes u and v by first computing the intersection $I := L_{\text{out}}(u) \cap L_{\text{in}}(v)$ and then choosing the minimum, among all $i \in I$, of $\text{dist}(u, i) + \text{dist}(i, v)$.

As the optimal choice of center nodes is NP-complete, Cohen et al. (2002) apply some heuristics to find an approximation of the optimal solution, but their algorithm does not scale well for very large graphs because it requires that all connections in the graph are computed in advance. The HOPI index provides a divide-and-conquer algorithm for building the 2-hop cover of an XML data graph that first partitions the graph into fragments whose connections fit into memory, then builds the cover for the partitions and finally joins the partial covers into the cover for the complete graph. More technical details of HOPI, including experimental results with real-life and synthetic data, can be found in Schenkel et al. (2004).

For heterogeneous XML collections, none of these path indexes is perfectly suited. Therefore, the XXL Search Engine provides the *FliX* framework (Schenkel 2004) for indexing paths that supports large, heterogeneous document collections with many links, using the existing path indexes as building blocks. It first divides the document set into carefully chosen fragments (so-called meta documents). After that, an index is built for each meta document, using the "best" available indexing strategy given the characteristics of the meta document. XPath axes like descendants or ancestors are then evaluated first on the local indexes (which will probably return the "best" results, i.e., elements that are connected with short paths). After that, results spanning multiple meta documents are evaluated by following links between meta documents at run-time.

5.3. Implementation issues

Our prototype implementation of the XXL Search Engine stores XML data in an Oracle 9i database with the following relational database schema (denoting primary keys as underlined columns):

```
URLS (urlid, url, lastmodified)
NAMES(nid, name)
NODES(oid, urlid, nid, pre, post, depth)
EDGES(oid1, oid2)
LINKS(oid1, oid2)
CONTENTS(oid, urlid, nid, content)

LIN (oid1, oid2, distance)
LOUT(oid1, oid2, distance)
```

Here, NODES, EDGES and CONTENTS store the actual XML data, URLS contains the urls of all XML documents known to the system, and LINKS holds the links between XML documents. Each element is identified using its unique oid value. Note that, for each element, the corresponding entry in CONTENTS stores the concatenated contents of the element and all its (transitive) subelements; for the INEX collection, this table contains about 1.1 gigabytes of data which is about two times the size of the original (XML) data. While this replication increases the overall storage usage, it heavily decreases query evaluation time. Additionally, this is the only way to apply Oracle Text to the contents which would be impossible without this kind of replication.

LIN and LOUT store the L_{in} and L_{out} sets used by the HOPI index. Here, a tuple (v, w, d) in LIN represents the fact that $w \in L_{\text{in}}(v)$, and the shortest distance of v and w is d. The pre and post order numbers as well as the depth of the node in the tree are augmented to the NODES table. The Ontology Index is represented by the following three relational tables:

```
CONCEPTS (cid, concept, description, freq)
WORDS (cid, word)
RELATIONSHIPS(cid1, cid2, type, freq, weight)
```

Currently the index stores hypernym, hyponym, holonym, and meronym edges, but it can be easily extended to support new edge types. The entries in the ontology index are extracted from the well-known electronic thesaurus WordNet (Fellbaum 1998). Frequencies and weights are computed as shown in Section 3.

6. Query processing in the XXL search engine

As mentioned in Section 4, an XXL query is of the form `SELECT S FROM F WHERE W1 AND ...AND Wk`. The evaluation of the search conditions in the `WHERE` clause consists of the following main steps:

- The XXL query is decomposed into subqueries `W1`, ...,`Wk`.
- A global evaluation order for evaluating the various subqueries is chosen.
- For each subquery, first a local evaluation order for evaluating the components of the subquery is chosen. Then, subgraphs of the data graph that match the query graph are computed, exploiting the various indexes to the best possible extent.
- The subresults are combined into the result for the original query.

6.1. Query decomposition

As an example for an XXL query, consider the following XXL query that asks for the titles of scientific articles about information retrieval and databases:

```
SELECT $T
FROM   INDEX
WHERE ~article AS $A
 AND  $A/~title AS $T
 AND  $A/#/~section ~ "IR & database"
```

The Where clause of an XXL query consists of a conjunction `"W1 And ...And Wn"` of subqueries `Wi`, where each subquery has one of the following types:

- `Pi`, i.e., a regular path expression,
- `Pi AS $A`, i.e., a regular path expression where the node at the end of the path is bound to variable `$A`,
- `Pi ~/LIKE/=/<>/</> condition`, i.e., a regular path expression with an additional content condition.

where each `Pi` is a regular path expression over elementary conditions, `$A` denotes a element variable to which the end node of a matching path is bound, and `condition` gives a content–based search condition using a binary operator. In our example, the first two subqueries are

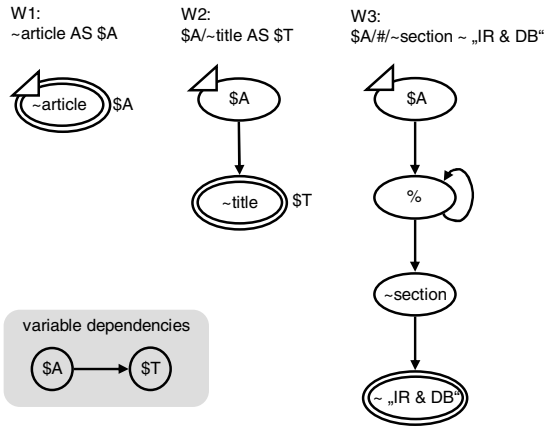

Figure 4. Non-deterministic finite state automata for the subqueries of the example XXL query.

of the second form, whereas the third subquery is of the third form. From the definitions of variables we derive the *variable dependency graph* that has an edge from `$V` to `$W` if the path bound to `$W` contains `$V`. We require the variable dependency graph of a valid XXL query to be acyclic. In our example, the variable dependency graph contains an edge `$A`→`$T` because `$A` is used in the definition of `$T`.

Each subquery corresponds to a regular expression over elementary conditions which can be described by an equivalent non-deterministic finite state automaton (NFSA). Figure 4 shows the automata for the subqueries of the example query and the variable dependency graph for this query. Note that the path wildcard operator `#` in the third subquery has been converted to the equivalent form `(%)*`.

6.2. *Global evaluation order*

To evaluate an XXL query, we first choose an order in which its subqueries are evaluated. This order must respect the variable dependency graph, i.e., before a subquery that defines a variable is evaluated, all subqueries that define variables used in this subquery must be evaluated. As this may still leave us some choices how to order subqueries, we estimate the selectivity of each subquery using simple statistics about the frequency of element names and search terms that appear as constants in the subquery. Then we choose to evaluate subqueries and bind the corresponding variables in ascending order of selectivity (i.e., estimated size of the intermediate result).

In our example, we may compare the frequency of element names that are similar to `article` with the combined frequency of the terms `IR` and `database` in either the whole collection or in the content of elements whose name is similar to `section`. This is done by first computing all element names with similarity above a given threshold and then querying Oracle Text for the number of elements with these names that contain the terms. As there will probably be much more articles in the collection than sections that contain the terms,

we may choose to first evaluate the last subquery, i.e., find sections that contain the terms, and then the first subquery, i.e., limit the results to sections within articles. The second subquery can be evaluated last, as it is only needed to compute the results that are returned to the user.

6.3. Subquery evaluation

Each subquery is mapped to its corresponding NFSA. A result for a single subquery, i.e. a *relevant path*, is a path of the XML data graph that matches a state sequence in the NFSA from an intial state to a final state. For such a relevant path, the relevance score is computed by multiplying the local relevance scores of all nodes of the path. In addition, all variables that occur in the subquery are assigned to one node of the relevant path.

A result for the query is then constructed from a consistent union of the variable assignments and a set of relevant paths (one from each subquery) that satisfies the variable assignments. The global relevance for such a result is computed by multiplying the local relevances of the subresults.

The local evaluation order for a subquery specifies the order in which states of the subquery's NFSA are matched with elements in the XML data graph. The XXL prototype supports two alternative strategies: in top-down order the matching begins with the start state of the NFSA and then proceeds towards the final state(s); in bottom-up order the matching begins with the final state(s) and then proceeds towards the start state.

As an example, we show how the NFSA shown in figure 5 is evaluated in top-down order on the data shown in that figure:

Step 1: The first elementary search condition contains a semantic similarity search condition on an element name. Thus, we consult the ontology index to get words which are similar to `article`, yielding the word `paper` with $sim(paper, article) = 0.9$. The first part of our result graph is therefore an n-node of the data graph named `article`, and it is assigned a local relevance score of 0.9.

Step 2: To be relevant for the query, a node from the result set of Step 1 must also have a child node with name `bdy`. As a result of Step 2, we consider result graphs formed by such nodes and their respective child.

Step 3: The next state in the NFSA corresponds to a wildcard for an arbitrary path in the data graph. Explicitly evaluating this condition at this stage would require an enumeration of the (possibly numerous) descendants of candidate results found so far, out of which only a few may satisfy the following conditions. We therefore proceed with the next condition in the NFSA and postpone evaluating the path wildcard to the next step. The following condition is again a semantic similarity condition, so we consult the ontology index to get words which are similar to `section`. Assume that the ontology index returns the word `sec` with a similarity score of 0.95. There are no n-nodes in the data that are named `section`, but we can add n-nodes named `sec` to our preliminary result with a local relevance score of 0.95.

Step 4: In this step we combine the results from steps 2 and 3 by combining n-nodes that are connected through an arbitrary path. The local relevance score for the results corresponds

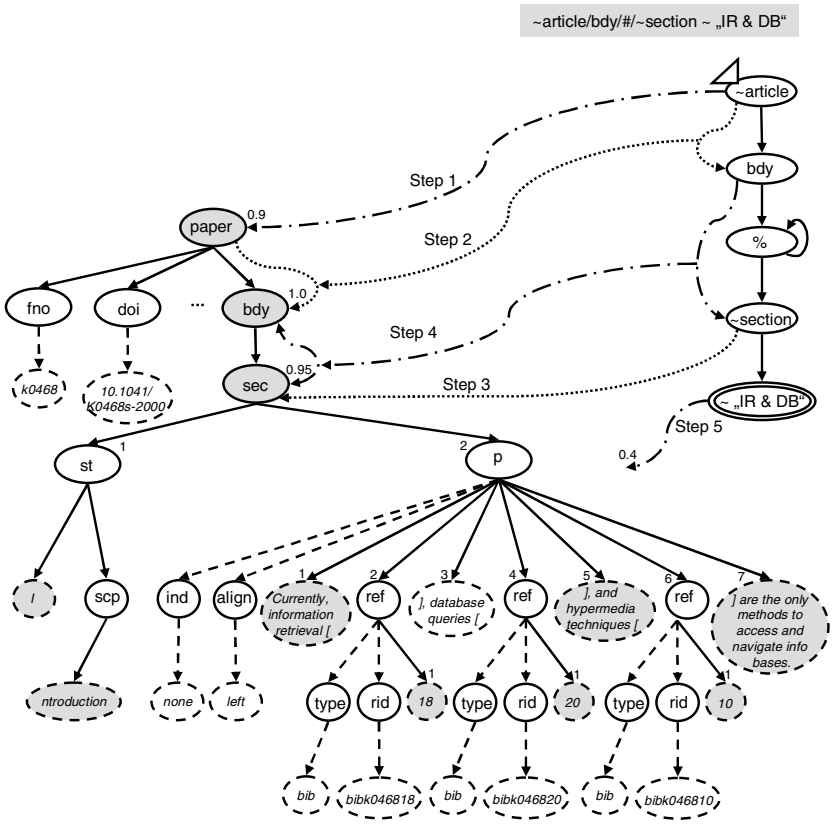

Figure 5. Evaluation of an NFSA in top-down manner.

to the inverse of the length of the path between the nodes: the longer the path is, the lower the relevance score.

Step 5: The final state of the NFSA contains a content-based semantic similarity search condition which must be satisfied by the content of a `sec`-element in the result set of Step 4. We first decompose the search condition that may consist of a conjunction of search terms into the atomic formulas (i.e., single terms). For each atomic formula we consult the ontology index for similar words and combine them in a disjunctive manner. We then use a text search engine to evaluate the relevance of each element's content which is expressed through an tf/ief-based relevance score. This score is combined with the ontology-based similarity score to the relevance score of the atomic formula. Finally, we multiply the relevance scores for each formula to get the relevance score for the similarity condition.

In our example, the shaded nodes in figure 5 form a relevant path for the given NFSA.

7. Experimental results with INEX

In this section, we present the results of our XXL search engine with the INEX benchmark; experimental results with several other data sets can be found in Theobald and Weikum (2002a). We start with a short summary of the INEX benchmark in Section 7.1. In Section 7.2, we present the results with keyword-only queries ("CO"-topics in INEX terminology), showing the overall quality of our approach as well as the additional benefit from applying ontology-backed query expansion. In Section 7.3, we discuss the results for queries with constraints on both content and structure ("CAS"-topics).

7.1. INEX overview

The INEX benchmark (Kazai et al. 2003) is an excellent testbed for information retrieval engines on XML data. It provides a large collection of XML documents with rich textual components, two sets of query topics (with and without structural constraints), and several metrics for evaluating the results.

For each topic the results of all participants are collected into a result pool for this topic. Then the potentially relevant components from each pool are assessed by a human who assigns an *exhaustivity* value and a *specificity* value which are both in the range 0–3. Exhaustivity describes the extent to which the component discusses the topic of request, specificity describes the extent to which the component focusses on the topic of request.

To assess the quality of a set of search results a metric based on the traditional recall/precision metrics is applied. In order to apply this metric, the assessors' judgements have to be quantised onto a single relevance value. Two different quantisation functions have been used:

1. *Strict* quantisation is used to evaluate whether a given retrieval approach is capable of retrieving highly exhaustive and highly specific document components.

$$f_{\text{strict}}(ex, spec) = \begin{cases} 1 & ex = 3, spec = 3 \ (\textit{short}: 3/3) \\ 0 & \text{otherwise} \end{cases}$$

2. In order to credit document components according to their degree of relevance (generalised recall/precision), a *generalized* quantisation is used.

$$f_{\text{generalized}}(ex, spec) = \begin{cases} 1 & 3/3 \\ 0.75 & 2/3, 3/2, 3/1 \\ 0.5 & 1/3, 2/2, 2/1 \\ 0.25 & 1/1, 1/2 \\ 0 & 0/0 \end{cases}$$

Using one of these quantisation functions, each document component in a result set is assigned a single relevance value using the human–based relevance assessment.

7.2. CO-topics

To automatically transform a CO-topic into an XXL query we consider the keywords given for the query. As there is no way to automatically decide how to combine these keywords (conjunctively, disjunctively or mixed) in an optimal manner, we chose to combine them conjunctively. To include results that are semantically similar to the keywords, we add our similarity operator ∼. For CO-topic 98 with keywords `information exchange`, `XML` and `information integration`, this process yields the following XXL query:

```
SELECT *
FROM INDEX
WHERE article/# ~  "(information exchange)
                   & XML
                   & (information integration)"
```

At query evaluation, each keyword in the query is (conceptually) replaced by the disjunction of itself and all its related terms:

```
SELECT *
FROM INDEX
WHERE article/# ~
("information exchange" | "data exchange"
  | "heterogeneous data")
& ("XML" | "semistructured data")
& ("information integration" | "information sharing")
```

For the unexpanded query we obtain 7 results with an average precision of 0.0002 for the strict quantisation and with an average precision of 0.0043 for the generalized quantisation. For the expanded query we obtain 28 results with an average precision of 0.0002 for the strict quantisation and with an average precision of 0.0065 for the generalized quantisation. So even this straightforward query expansion helped to slightly increase result quality.

However, if we carefully look at the topic, it turns out that a recombination of the query keywords could return better results. (Note that such an optimization is not allowed for the "official" INEX runs as all queries had to be generated automatically.) Thus, we reformulate the unexpanded query:

```
("information exchange" | "information integration") & "XML"
```

The expanded query then has the following structure:

```
(("information exchange" | "data exchange" |
  "heterogeneous data") |
```

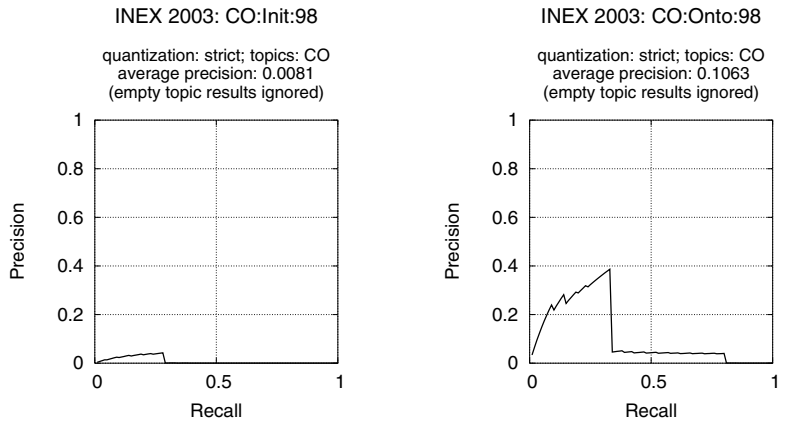

Figure 6. Average precision (strict) for the reformulated CO-query 98 without (left) and with (right) query expansion.

```
 ("information integration" | "information sharing")) &
("XML" | "semistructured data")
```

Figure 6 shows the precision-recall curves with the strict quantization for CO-topic 98 for the reformulated, but unexpanded query (left) and the expanded query (right), and figure 7 shows the precision-recall curves with the generalized quantization for this topic with reformulation, but without expansion (left) and with query expansion (right). For both metrics, the expanded queries have a much higher average precision, and they deliver much more relevant results than the unexpanded queries. However, especially for the generalized

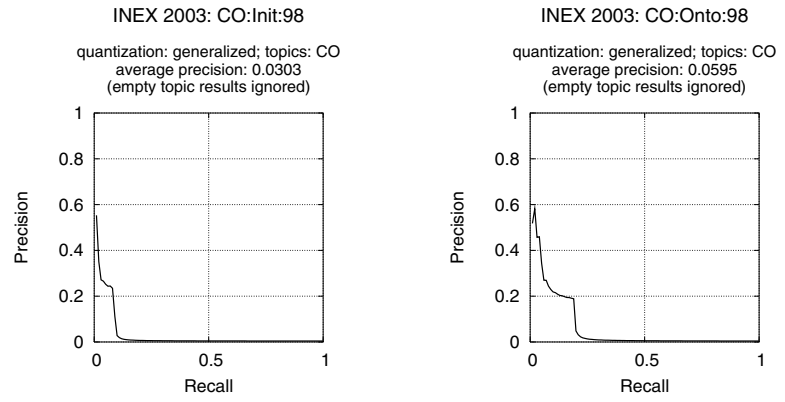

Figure 7. Average precision (generalized) for the reformulated CO-query 98 without (left) and with (right) query expansion.

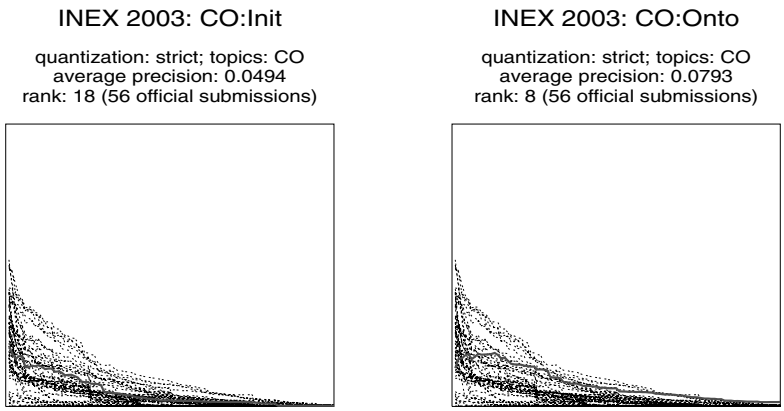

Figure 8. Precision-recall curves (strict) for all 36 CO-topics without (left) and with (right) ontological query expansion.

metrics, recall is still quite low; we attribute this to the fact that many elements have been assessed as (to some extent) relevant that don't contain all or even any of the ontologically expanded keywords. As XXL uses conjunctive keyword conditions, it considers all those elements as non-relevant, leaving them out of the result list.

For the complete set of all 36 CO-topics,[1] the results clearly indicate that ontological query expansion indeed does increase query results. Figure 8 show the precision-recall curves with strict quantisation for our search Engine (drawn in bold) in comparison to the others that participated in INEX, figure 9 shows them with generalized quantization. Note that the absolute average precisions are quite low for all systems; this is mostly due to the

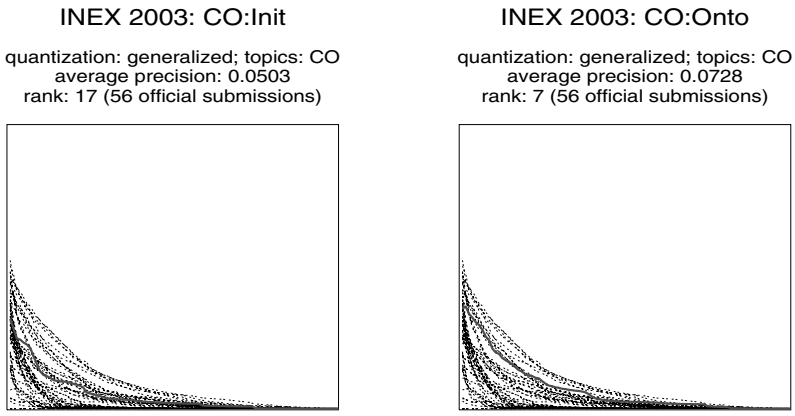

Figure 9. Precision-recall curves (generalized) for all 36 CO-topics without (left) and with (right) ontological query expansion.

difficulty to select the right result granularity (depending on the topic, an `article` element may not be a good result even though it contains all keywords, because its content is too general; analogously, the content of a `section` element may be too narrow even though containing all keywords. Choosing the right granularity for the result is currently an active research problem in XML retrieval.

Our search engine took about 15 minutes to generate the results for all 36 CO-topics; this time was dominated by the queries to OracleText and the generation of the results in the INEX format.

The results clearly show that ontology-based query expansion for keyword-based XML retrieval provides much better average precision and better recall.

7.3. CAS-topics

CAS-topics are represented by an expression in an XPath-like query language. We map this expression in a straightforward way to a corresponding XXL expression, adding semantic similarity conditions to all element names and keywords that appear in the XPath expression. For CAS-topic 63, the query

```
//article[about(., "digital library")]
      //p[about(., "authorization & access control & security")]
```

is mapped to the following XXL query:

```
SELECT $A
FROM INDEX
WHERE article AS $A
```

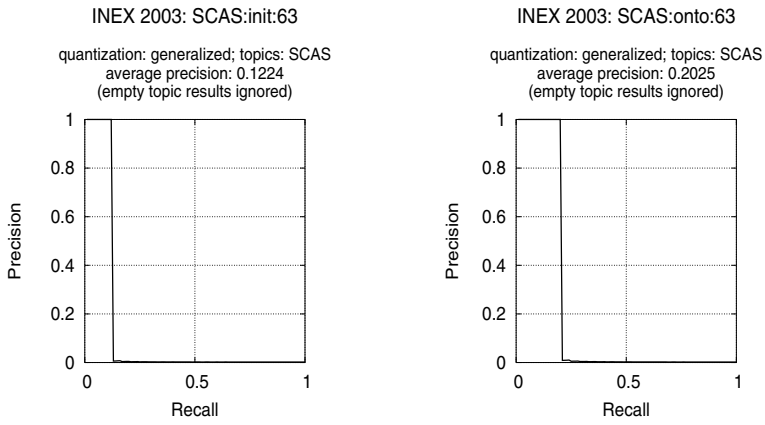

Figure 10. Precision-recall curves with generalized quantization for CAS-topic 63 without (left) and with (right) ontological query expansion.

```
AND $A ~ "digital library"
AND $A/#/p ~ "authorization & (access control) &
              security"
```

As an example for the ontology-based query evaluation on CAS-topics, figure 10 shows the generalized results for CAS-topic 63 with the original query (left) and with the expanded query (right). The strict evaluation results in an average precision of 1.0 in both cases.

This experiment shows that the XXL search engine is able to evaluate conditions on XML structure as well as conditions on XML contents. In addition, the ontology–based query expansion for the content condition provides much better average precision and better recall. Due to the regular structure of the INEX data, XXL could not make much use of its structural similarity features. We expect that INEX' upcoming heterogeneous data track will allow us to demonstrate the strength of XXL in that field.

8. Conclusions and future work

Querying large collections of highly heterogeneous, richly structured XML documents is an important task where existing XML query languages like XQuery and keyword-based search engines are failing. The XXL Search Engine presented in this article is an important step towards solving this problem. Combining keyword-based search with structural conditions and semantic similarity, its query language XXL is more expressive than all existing query languages for XML. The results obtained with the INEX benchmark clearly indicate that exploiting semantic similarity generally increases the quality of search results.

Our ongoing and future work includes generalizing query semantics, exploiting user feedback, and further optimizing query performance. More specifically, we plan to generalize the semantics of the path wildcard operator # to include more general notions of connectivity like those discussed by Amer-Yahia et al. (2002). To further extend the result quality, we plan to add a relevance feedback step to incrementally increase the quality over time, including a user-specific personal ontology. Finally, we plan further studies on query optimization heuristics. This includes finding good global and local evaluation ordering of subqueries and elementary search conditions based on selectivity estimations as well as algorithms to quickly return the best results for the query (without having to compute all results first).

Note

1. Note that these results were submitted after the official INEX deadline, so the figures do not show the official rank of our search engine in the INEX benchmark.

References

Aboulnaga A, et al. (2001) Estimating the selectivity of XML path expressions for Internet scale applications. In: VLDB 2001, pp. 591–600.

Agirre E and Rigau G (1996) Word sense disambiguation using conceptual density. In: 16th Int. Conf. on Computational Linguistics 1996, pp. 16–22.

Amer-Yahia S, Botev C and Shanmugasundaram J (2004) TeXQuery: A full-text search extension to XQuery. In: WWW 2004. Online Proceedings, available from http://www2004.org/.

Amer-Yahia S, et al. (2002) Tree pattern relaxation. In: Jensen CS et al., Eds., EDBT 2002, pp. 496–513.

Baeza-Yates RA and Riberto-Neto B, Eds. (1999) Modern Information Retrieval. Addison Wesley.

Blanken H, Grabs T, Schek H-J, Schenkel R and Weikum G., Eds. (2003) Intelligent Search on XML Data, vol. 2818 of LNCS.

Boag S, et al. (2002) XQuery 1.0: An XML query language. W3c recommendation, World Wide Web Consortium. http://www.w3.org/TR/xquery.

Brin S and Page L (1998) The anatomy of a large-scale hypertextual web search engine. Computer Networks, 30(1–7):107–117.

Bruno N, Koudas N and Srivastava D (2002) Holistic twig joins: Optimal XML pattern matching. In: SIGMOD 2002, pp. 310–321.

Budanitsky A and Hirst G (2001) Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In: Workshop on WordNet and Other Lexical Resources. Online Proceedings, available from http://engr.smu.edu/ rada/mwnw/.

Chen Z, et al. (2001) Counting twig matches in a tree. In: ICDE 2001, pp. 395–404.

Chinenyanga TT and Kushmerick N (2001) Expressive and efficient ranked querying of XML data. In: WebDB 2001, pp. 1–6.

Chinenyanga TT and Kushmerick N (2002) An expressive and efficient language for XML information retrieval. Journal of the Americal Society for Information Science & Technology, 53(6):438–453.

Choi B, et al. (2003) On the optimality of holistic algorithms for twig queries. In: DEXA 2003, pp. 28–37.

Chung C-W, et al. (2002) APEX: An adaptive path index for XML data. In: SIGMOD 2002, pp. 121–132.

Cohen E, et al. (2002) Reachability and distance queries via 2-hop labels. In: 13th ACM-SIAM Symposium on Discrete algorithms (SODA 2002), pp. 937–946.

Cohen S, et al. (2003) XSEarch: A semantic search engine for XML. In: VLDB 2003, pp. 45–56.

Cohen WW (1999) Recognizing structure in Web Pages using similarity queries. In: 16th National Conference of Artificial Intelligence (AAAI)/11th Conference on Innovative Applications of Artificial Intelligence (IAAI), pp. 59–66.

Cormen TH, Leiserson CE and Rivest RL (2001) Introduction to Algorithms. 2nd edition, MIT Press.

Deutsch A, Fernandez MF, Florescu D, Levy AY and Suciu D (1998) XML-QL. In: QL '98, The Query Languages Workshop, W3C Workshop. available from http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/.

Fellbaum C, ed. (1998) WordNet: An Electronic Lexical Database. MIT Press.

Fuhr N and Großjohann K (2001) 'XIRQL: A query language for information retrieval in XML documents. In: SIGIR 2001, pp. 172–180.

Graupmann J, Biwer M, Zimmer C, Zimmer P, Bender M, Theobald M and Weikum G (2004) COMPASS: A concept-based Web search engine for HTML, XML, and deep Web data (demo). In: VLDB 2004.

Grust T (2002)Accelerating XPath location steps. In: SIGMOD 2002, pp. 109–120.

Grust T and van Keulen M (2003) Tree awareness for relational DBMS kernels: staircase join. In: Blanken H, Grabs T, Schek H-J, Schenkel R and Weikum G., Eds. Intelligent Search on XML Data, vol. 2818 of LNCS, pp. 231–245.

Guha S, et al. (2002) Approximate XML joins. In: SIGMOD 2002, pp. 278–298.

Guo L, et al. (2003) XRANK: ranked keyword search over XML documents. In: SIGMOD 2003, pp. 16–27.

Hayashi Y, et al. (2000) Searching text-rich XML documents with relevance ranking. In: ACM SIGIR 2000 Workshop on XML and Information Retrieval. Online Proceedings, available from http://www.haifa.il.ibm.com/sigir00-xml/.

Hirst G and St-Onge D (1998) Lexical chains as representations of context for the detection and correction of malapropisms. In: Fellbaum C, Ed. WordNet: An Electronic Lexical Database. MIT Press, pp. 305–332.

Horrocks I (2002) DAML+OIL: A reason-able Web ontology language. In: EDBT 2002, pp. 2–13.

Jamasz M and Szpankowicz S (2001) Roget's thesaurus: A lexical resource to treasure. In: Proceedings of the NAACL Workshop "WordNet and Other Lexical Resources," Pittsburg, pp. 186–188.

Jamasz M and Szpankowicz S (2003) Roget's thesaurus and semantic similarity. Technical Report TR-2003-01, University of Ottawa, Canada.

Jiang H, et al. (2003) Holistic twig joins on indexed XML documents. In: VLDB 2003, pp. 273–284.

Jiang JJ and Conrath DW (1997) Semantic similarity based on corpus statistics and lexical taxonomy. In: 10th Int. Conf. on Research on Computational Linguistics (ROCLING 1997), Taipeh, Taiwan, pp. 19–33.

Kaushik R, et al. (2002) Covering indexes for branching path queries. In: SIGMOD 2002, pp. 133–144.

Kaushik R, et al. (2004) On the integration of structure indexes and inverted lists. In: SIGMOD 2004, pp. 779–790.

Kazai G, et al. (2003) The INEX evaluation initiative. In: Blanken H, Grabs T, Schek H-J, Schenkel R and Weikum G., Eds. Intelligent Search on XML Data, vol. 2818 of LNCS, pp. 279–293.

Kleinberg J (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5):604–632.

Leacock C and Chodrow M (1998) Combining local context and WordNet similarity for word sense disambiguation. In: Fellbaum C, Ed. WordNet: An Electronic Lexical Database. MIT Press, pp. 265–283.

Lenat DB and Guha RV (1990) Building Large Knowledge Based Systems. Addison Wesley.

Lesk M (1969) Word-word association in document retrieval systems'. American Documentation, 20(1):27–38.

Lewis WD (2002) Measuring conceptual distance using WordNet: The design of a metric for measuring semantic similarity. The University of Arizona Working Papers in Linguistics, 12.

Lin D (1998) An information-theoretic definition of similarity'. In: 15th Int. Conf. on Machine Learning (ICML 1998), pp. 296–304.

Manning CD and Schuetze H (1999) Foundations of Statistical Natural Language Processing. The MIT Press.

McHale ML (1998) A comparison of WordNet and Roget's taxonomy for measuring semantic similarity. In: Workshop on Usage of WordNet in Natural Language Processing Systems (COLING-ACL 1998). Online Proceedings, available from http://xxx.lanl.gov/abs/cmp-lg/9809003.

Oracle Corp. (2004) Oracle 9i text. http://otn.oracle.com/products/text/.

Polyzotis N and Garofalakis MN (2002a) Statistical synopses for graph-structured XML databases. In: SIGMOD 2002, pp. 358–369.

Polyzotis N and Garofalakis MN (2002b) Structure and value synopses for XML data graphs. In: VLDB 2002, pp. 466–477.

Qiu Y and Frei H-P (1993) Concept-based query expansion. In: SIGIR 1993, pp. 160–169.

Qiu Y and Frei H-P (1995) Improving the retrieval effectiveness by a similarity thesaurus. Technical Report 225, Swiss Federate Institute of Technology, Zürich, Switzerland.

Qun C, et al. (2003) D(k)-index: An adaptive structural summary for graph-structured data. In: SIGMOD 2003, pp. 134–144.

Rada R, et al. (1989) Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics, 19(1):17–30.

Resnik P (1995) Using information content to evaluate semantic similarity in a taxonomy. In: 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 95), Vol. 1. pp. 448–453.

Resnik P (1999) Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research, 11:95–130.

Richardson R, et al. (1994) Using WordNet as a knowledge base for measuring semantic similarity between words. In: Proceedings of the AICS Conference.

Rubenstein H and Goodenough JB (1965) Contextual correlates of synonymy. Communications of the ACM, 8(10):627–633.

Russel S and Norvig P (1995) Artificial Intelligence—A Modern Approach. Prentice Hall.

Schenkel R (2004) FliX: A flexible framework for indexing complex XML document collections. In: 1st Int. Workshop on Database Technologies for Handling XML Information on the Web.

Schenkel R, Theobald A and Weikum G (2003) Ontology-enabled XML search. In: Blanken H, Grabs T, Schek H-J, Schenkel R and Weikum G., Eds. Intelligent Search on XML Data, vol. 2818 of LNCS, pp. 119–131.

Schenkel R, Theobald A and Weikum G (2004) HOPI: An efficient connection index for complex XML document collections. In: EDBT 2004, pp. 237–255.

Schlieder T (2002) Schema-driven evaluation of approximate tree-pattern queries. In: EDBT 2002, pp. 514–532.

Schlieder T and Meuss H (2000) Result ranking for structured queries against XML documents. In: DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries, available from http://www.ercim.org/publication/ws-proceedings/DelNoe01.

Staab S, et al. (2000) Semantic community web portals. Computer Networks, 33(1–6):473–491.
Sussna M (1993) Word sense disambiguation for free-text indexing using a massive semantic network. In: 2nd Int. Conf. on Information and Knowledge Management (CIKM 1993), pp. 67–74.
Theobald A and Weikum G (2000) Adding relevance to XML. In: 3rd Int. Workshop WebDB 2000, pp. 105–124.
Theobald A and Weikum G (2002a) The index-based XXL search engine for querying XML data with relevance ranking. In: EDBT 2002, pp. 477–495.
Theobald A and Weikum G (2002b) The XXL search engine: Ranked retrieval of XML data using indexes and ontologies. In: SIGMOD 2002, p. 615.
Wu Y, Patel JM and Jagadish H (2002) Estimating answer sizes for XML queries. In: EDBT 2002, pp. 590–608.
Wu Z and Palmer M (1994) Verb semantics and lexical selection. In: 32nd. Annual Meeting of the Association for Computational Linguistics 1994, pp. 133 –138.
Zezula P, et al. (2003) Tree signatures for XML querying and navigation. In: 1st Int. XML Database Symposium, pp. 149–163.