Classification and Intelligent Search on Information in XML

Norbert Fuhr
University of Dortmund, Germany

Gerhard Weikum
University of the Saarland, Germany

1 Introduction

XML will be the method of choice for representing all kinds of documents in product catalogs, digital libraries, scientific data repositories, and across the Web. This observation creates high expectations that XML will be a major catalyst in constructing the "Semantic Web". However, merely casting all documents into XML format does not necessarily make a document's semantics explicit and more amenable for effective information searching. Rather, to fully leverage XML on a global scale, significant progress is needed on the following issues:

1. providing an easy-to-use yet powerful and efficient search language that combines concepts from current XML pattern-matching languages (e.g., XPath, XQuery, etc.) with ontology-backed information-retrieval-style search result ranking,

2. extracting more semantics from existing document collections by constructing structural and ontological skeletons (e.g., in the form of DTDs or XML schemas) that describe the data at a higher semantic level and can also facilitate new forms of indexing for efficiency, and

3. classifying existing documents according to a given thematic or personalized, hierarchical ontology to make searching more effective (e.g., exploit relevance feedback) and efficient (e.g., limit the search focus).

CLASSIX, a joint project of the Universities of Dortmund and the Saarland in Germany, addresses these three issues. We describe our approaches for each of these topics in the remainder of this paper.

2 Query languages for information retrieval of XML documents

Looking at the broad variety of XML applications and systems that are currently under development, one can see that there are in fact two different views on XML:

- The *document-centric view* focuses on structured documents in the traditional sense (based on concepts from electronic publishing, especially SGML). Here XML is used for logical markup of texts both at the macro level (e.g. chapter, section, paragraph) and the micro level (e.g. MathML for mathematical formulas, CML for chemical formulas).

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

- The *data-centric view* uses XML for exchanging formatted data in a generic, serialized form between different applications (e.g. spreadsheets, database records). This is especially important for e-business applications (e.g. for exchanging orders, bills).

Unfortunately, most approaches for XML query languages (including the W3C working group on XML query languages with its proposal of XQuery) have concentrated on the data-centric view, thus providing little support for information retrieval of XML documents.

In contrast, our work focuses on the document-centric view. For information retrieval of XML documents, we take into account the intrinsic imprecision and vagueness of IR and the resulting need for ranked lists as search results (as opposed to result sets or bags in traditional database querying). For this purpose, we are developing the query language XIRQL (Xml IR Query Language), which extends the XPath part of the (proposed standard) query language XQuery by the following features:

- weighting and ranking,
- relevance-oriented search,
- data types with vague predicates,
- structural relativism.

Below, we describe each of these concepts in detail.

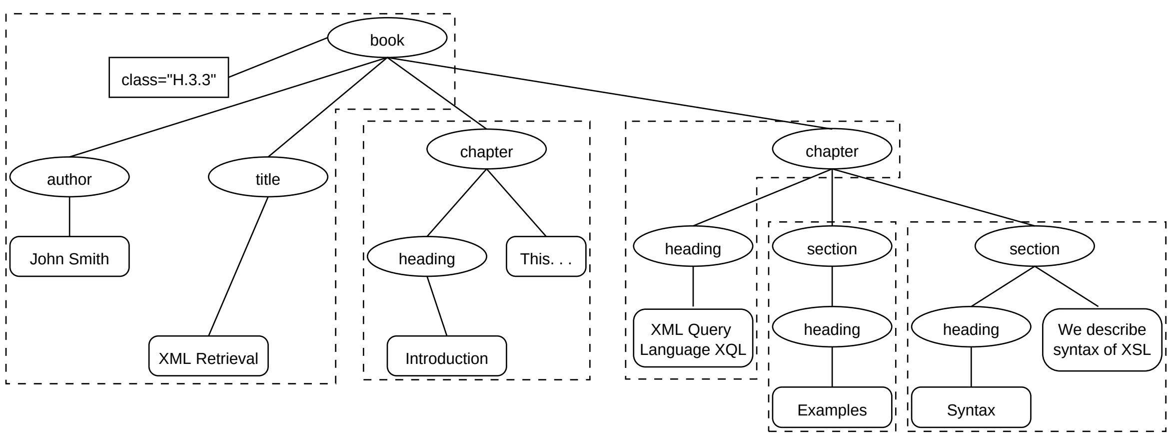

Figure 1: Example XML document tree

Weighting and ranking. IR research has shown that document term weighting as well as query term weighting are necessary tools for effective retrieval in textual documents. So query conditions referring to the text of elements should consider index term weights. Furthermore, query term weighting should also be possible, by introducing a weighted sum operator (e.g. $0.6 \cdot$ "XML" + $0.4 \cdot$ "retrieval"). These weights should be used for computing an overall retrieval score for the elements retrieved, thus resulting in a ranked list of elements.

The basic idea for assigning indexing weights to document terms is that the weight of a term depends on its context. So we split up a document into disjoint contexts which we call index nodes; based on the DTD, index nodes are specified by giving the names of those elements that form the roots of important and "semantically coherent" subtrees of XML documents. Figure 1 shows an example where index nodes are marked as dashed boxes. For each term in such a context, the indexing weight is computed by using standard weighting functions like e.g. tf·idf.

For retrieval, we use a probabilistic retrieval model where we treat all term occurrences within the same index node as a single probabilistic event, whereas occurrences of other terms in the same node or of the same term in other nodes are regarded as being independent events. When a query condition matches a term occurrence within an index node, the event representing the corresponding term-node pair (along with its weight) is returned as answer to this condition.

As an example, for the document in Figure 1, a query searching for the word 'syntax' occurring in a heading would match the rightmost heading element. Since the word 'syntax' also occurs in the body of the corresponding section, which is part of the same index node, these additional occurrences affect the weight of the term in the index node, and thus also the weight of the match.

By using the concept of event keys and event expressions from [9], we can compute result probabilities in a consistent way, even for complex combinations of query conditions. Due to the (assumed) independence of events, our approach always yields point probabilities, thus yielding a linear ranking of results.

Relevance-oriented search. The query language should also support traditional IR queries, where only the requested content is specified, but not the type of elements to be retrieved. In this case, the IR system should be able to retrieve the most relevant elements; following [4], we assume that this should be the most specific element(s) that satisfy the query. In the presence of weighted index terms, the tradeoff between these weights and the specifity of an answer has to be considered, e.g. by an appropriate weighting scheme.

For this purpose, we introduce the concept of augmentation. The index weights of the most specific index nodes are given directly. For retrieval of the higher-level objects, we have to combine the weights of the different text units contained. When propagating indexing weights to the higher-level objects, they are downweighted (multiplied by an augmentation weight), such that, in general, more specific results get higher retrieval weights.

In addition, since not all elements of a document may be reasonable answers for relevance-oriented queries, we restrict the set of possible answers to the roots of index nodes. For example, consider the relevance-oriented query 'syntax $\wedge$ example'. In the document shown in Figure 1, there is no single index node matching this query; however, the rightmost chapter satisfies all conditions, when we propagate the weights of the two query terms up to this level. In contrast, a query for 'XSL' would yield the highest weight for the last section, whereas the comprising chapter would be returned with a lower weight.

Data types and vague predicates. The standard IR approach for weighting supports vague searches on plain text only. XML allows for a fine grained markup of elements, and thus, there should be the possibility to use special search predicates for different types of elements. For example, for an element containing person names, similarity search for proper names should be offered; in technical documents, elements containing measurement values should be searchable by means of the comparison predicates $>$ and $<$ operating on floating point numbers. Thus, there should be the possibility of having elements of different data types, where each data type comes with a set of specific search predicates. In order to support the intrinsic vagueness of IR, most of these predicates should be vague (e.g. search for measurements that were taken at about 20 °C).

Whereas XML schema focuses on those properties of data types that can be checked at the syntactic level (i.e. structure and domain of possible values), we assume that the difference between many data types is visible at the semantic level only; thus, we characterize data types by their sets of vague predicates (such as phonetic similarity of names, English vs. French stemming). In principle, data types with vague predicates generalize text indexing methods for all kinds of data. Thus, the consideration regarding the probabilistic interpretation of weights apply here as well.

Structural relativism. Database-oriented XML query languages are closely tied to the structure of XML documents, but it is possible to use syntactically different XML variants to express the same meaning. For example, a particular information could be encoded as an XML attribute or as an XML element. As another example, a user may wish to search for a value of a specific datatype in a document (e.g. a person name), without bothering about the element names.

For this purpose, we support query conditions that ignore the difference between elements and attributes of a specific name. Based on our notion of datatypes, we allow for searches covering all elements of a specific datatypes. We are investigating further possibilities of generalizations (e.g. by introducing hierarchies over elements, similar to `rdfs:subPropertyOf` in RDF schema).

Implementation. As a first prototype, we have implemented the retrieval engine named HyREX[1]. This system accepts XIRQL queries, translates them into an internal path algebra [7] [8] and processes these expressions based on inverted files; the current version performs efficient retrieval for document collections up to the gigabyte range.

3 Metadata

Metadata generally adds to the value of raw XML data by making it easier to interpret the data semantics and reason about relationships within the data. In the CLASSIX project we are pursuing a pragmatic approach toward constructing personal or domain-specific ontologies as a metadata backbone that can guide the user and an XML search engine in refining queries. In contrast to a pure logical representation of ontological relationships, we aim to quantify the similarity of related "concepts" as a basis for computing similarity scores between XML (sub-)documents and queries. This way we can capture, for example, that synonyms are closer to each other than hyper- and hyponyms.

Ontology construction. There is a trend toward organizing XML documents according to domain-specific ontologies with meaningful, carefully chosen element names. However, it cannot be expected that all documents for a given domain will eventually correspond to a standardized XML schema; rather we still expect substantial schematic variety, for three reasons: 1) not every author of a (high-quality) document is willing to adopt standardized terminology, 2) ontologies are evolving over time, and 3) there is some "natural" diversity of ontologies for the same or overlapping domains.

For these reasons, we treat ontogical metadata as an optional tool for searching XML data, rather than firmly relying on its existence and unambiguity. In the CLASSIX project, we are building application-specific ontology indexes as follows. When a crawler traverses XML documents from the Web or an intranet, all element names that are not yet in the application's ontology index are added as nodes to a graph. The edges between nodes represent semantic relationships; currently we distinguish only between synonym and (different kinds of) hypernym/hyponym edges. The position of a new element name in the index graph is determined by calling WordNet [6], a comprehensive thesaurus put together by cognitive scientists. We extract the concept description (i.e., the "word sense" in WordNet terminology), all synonyms, and all hypernyms and hyponyms for the given word. In addition to WordNet, we are also considering other sources of authoritative ontological information such as XML-based directories or lexicons for the domain of interest. When the crawler encounters a document collection whose entire data corresponds to the same DTD or XML schema, this step needs to be performed only once.

After an element name has been added to the index graph and its edges to and from already existing nodes have been constructed, the second step is to quantify the new relationships by computing a similarity weight between adjacent nodes. Here again, we are currently pursuing a fairly pragmatic approach, but plan to investigate better theoretical foundations as well. The current approach is based on extracting co-occurence frequencies from various sources like Google, Yahoo, and other Web search engines and directories. The similarity weight

[1] `http://ls6-www.informatik.uni-dortmund.de/ir/projects/hyrex/`

between two nodes increases in proportion to the frequency of finding two corresponding words (or, more precisely, synonym sets) in the same document.

Query refinement. The ontology index is exploited to expand or narrow the search scope of a user query, automatically or in combination with a user feedback step. This is important in two cases: when the original query is "overspecified" using overly specific search conditions that would lead to very few results or even no matches at all, or when the query is "underspecified" using overly broad or even ambiguous search terms and conditions. In the first case we would consider generating a refined query with additional search conditions based on hypernyms and their synonyms; in the second case we would add selected hyponyms (and possibly more related terms when useful for disambiguation). An additional option to consider would be the re-adjustment of term weights in the query.

As an example, consider a query with a similarity search condition $\approx$*region* $\sim\sim$ *"India"*. Here *region* is an element name that needs to be matched, with the additional condition that the element content contains the term *"India"*. The unary similarity operator $\sim$ denotes that the element name does not need to occur literally but should rather be matched "semantically", and the binary similarity operator $\sim\sim$ denotes an analogous approximation for finding the specified term in the element content. To allow for approximate matches, the original element name *region* could be expanded into the disjunction *region* | *country* | *continent* and the search for *"India"* should consider also related terms such as *"Asia"*, *"Bangladesh"*, *"Tamil Nadu"*, etc., provided that the similarity scores obtained from the ontology index are above some threshold. The ranking of the search results reflects the similarities between the terms in the query and the found documents. For example, a document with an element name *continent* and the word *"Asia"* in its content will be ranked lower than a document with an element named *region* and containing the term *"India"*.

Implementation. The XXL prototype system (fle<u>x</u>ible <u>X</u>ML search <u>l</u>anguage), an XML crawler and search engine, includes basic support for ontology-based similarity search [15, 16]. The system constructs an ontology index from XML element names seen during a crawl by looking up WordNet entries as sketched above, and expands queries with synonyms and hyper-/hyponyms. The ranking of search results reflects the corresponding proximity measures. Preliminary experiments indicate that this approach can significantly improve the precision and recall of searching XML document collections [15].

4 Automatic Classification

XML documents can be searched more effectively and efficiently when they are organized in a more explicit way. For this purpose, we are investigating how we can automatically classify newly crawled documents into a hierarchical taxonomy. In other words, we not only build ontologies at the metadata level, but want to populate the ontological concepts or categories of interest with actual data. For simplicity, we so far concentrate on trees of topics that reflect the user's or user community's interest profile. Typically, such an application-specific taxonomy would be an order of magnitude smaller than say the complete directory structure of a Yahoo-like service. Each node in the tree would be a priori populated with a number of prototypical documents derived from user bookmarks, surf trails, and other user profiling information. These documents serve as training data for the classifier.

Rather than viewing crawling and classification as two separate stages of a data organization engine, we interleave these two steps following a paradigm known as *focused crawling* [3]. A focused crawl starts from the training documents of the taxonomy's categories as seeds and then traverses a, hopefully small, fraction of the Web with focus on these topics of interest. The classification is invoked for each visited document, which is either added to one or more categories when the classification test is positive or discarded in the negative case.

The accuracy of the classification step depends on three key aspects:

- the mathematical model and algorithm that are used for classification,
- the feature set upon which the classifier makes its decisions, and
- the quality of the training data that is initially categorized by human users and from which the classifier derives parameters for its decision model.

These three issues are discussed next.

Supervised learning for hierarchical taxonomies. The most natural way of classifying a new document is a top-down procedure starting from the root of the taxonomy [5]. For each category that is considered a classifier is invoked and returns a yes-or-no decision and possibly a confidence measure of the decision. When the documents fits with more than one category, either the one with the highest confidence measure is chosen or the document is classified into multiple categories. Then the classification proceeds with the children of the categories to which the document has been added.

This procedure requires a binary classifier for each category, a test that decides whether a document belongs to the category or not. To this end we are using a supervised learning method known as support vector machines (SVM) [2]. Unlike simpler classifiers such as Naive Bayes, this technique takes all correlations in the underlying feature space (i.e., the term vector space of the documents) into account, by learning an optimal hyperplane that separates positive from negative training documents in the feature space (possibly with some outliers on the "wrong" side of the hyperplane). The decision procedure is a very efficient scalar product computation, testing on which side of the hyperplane a document lies, and the distance from the hpyerplane can be interpreted as a confidence measure for the classification of a document.

An SVM classifier needs both positive and negative training data for each topic. So far we treat all positive training data from a given category's siblings in the taxonomy (i.e., the competing categories given that a document has already been classified into the parent category) as negative training data for the given category. This seemingly natural approach can be expected to work well for a "closed-world" document collection such as the popular Reuters collection in the TREC benchmark, where each document that will ever be passed to the classifier is known to belong to at least one of the categories. With Web data or documents from a highly diverse intranet, however, this assumption is not valid. We are currently investigating to what extent additional, explicit negative training data should be incorporated and how this can be done without imposing too much of a burden on the human user.

Feature selection. For efficiency as well as accuracy typically only a subset of the documents' terms are used as components of the feature space on which the classifiers are based [10, 17]. A good feature discriminates competing topics (i.e., siblings in the taxonomy tree) from each other. Therefore, feature selection has to be topic-specific; it needs to be performed for each topic in the tree individually. As an example, consider a taxonomy with topics mathematics, agriculture, and arts, where mathematics has subcategories algebra and stochastics. Obviously, the term "theorem" is very characteristic for math documents and thus an excellent discriminator between mathematics, agriculture, and arts. However, it is of no use at all to discriminate algebra versus stochastics. A terms such as "field", on the other hand, is a good indicator for the topic algebra when the only competing topic is stochastics; however, it is useless for a classifier that test mathematics versus agriculture.

To compute the best features for a topic-specific classifier we are using information-theoretic measures, specifically, the cross-entropy between feature frequencies and topics (a special case of the Kullback-Leibler divergence, which measures the differences between multivariate probability distributions, the joint distribution of features and classes versus independent distributions in our setting) [12]. As feature candidates we are studying individual terms as well as term pairs that occur in the same XML element, XML element names, and also pairs of parent-child element names. A widely open issue in this context is to what extent additional, external sources can provide good features; for example, the neighbors of a hyperlinked document, the DTD of an XML document, or a thesaurus for a given category may yield features that do not occur in the training documents themselves.

Online training. An inherent difficulty in automatic classification is the scarceness of good training data. Motivated by some initial work on using automatically classified documents as additional training data [13], we are pursuing an approach with continuous online training. For this purpose, a set of the most characteristic documents of a topic, coined *archetypes*, are determined in two, complementary ways. First, a link analysis procedure, using a variant of Kleinberg's HITS algorithm [11, 1], is initiated. This procedure yields a ranking of *authorities* for each topic, documents that contain high quality information relevant to the topic. The second source of topic-specific archetypes builds on the confidence of the classifier's yes-or-no decision for a given node of the taxonomy tree. Among the automatically classified documents of a topic those documents whose yes decision had the highest confidence measures are selected as archetypes. Archetypes and the original training documents are then fed into the classifier for online re-training.

Implementation. The above considerations have been implemented in a prototype focused crawler coined BINGO! (for bookmark-induced gathering of information) [14]. BINGO! is completely implemented in Java and uses Oracle8i as an underlying storage engine. We are currently working on a first round of large-scale, long-running experiments to evaluate the viability and benefits of our approach.

5 Conclusion

This paper has discussed several key issues in exploiting the potential role of XML as a catalyst toward a "Semantic Web". We are investigating each of these issues, aiming to understand how they can contribute to better organization and more effective search of data. In addition, we are studying the interplay of these various aspects, striving for synergies among IR and DB querying concepts, ontological metadata, and machine learning techniques for automatic classification.

References

[1] K. Bharat, M. Henzinger: Improved Algorithms for Topic Distillation in a Hyperlinked Environment, ACM SIGIR Conference, 1998.

[2] C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery Vol.2 No.2, 1998.

[3] S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, WWW Conference, 1999.

[4] Y. Chiaramella, P.Mulhem, F.Fourel: A Model for Multimedia Information Retrieval, FERMI ESPRIT BRA 8134, University of Glasgow, 1996.

[5] S. Dumais, H. Chen: Hierarchical Classification of Web Content, ACM SIGIR Conference, 2000.

[6] C. Fellbaum (Editor): WordNet: An Electronic Lexical Database, MIT Press, 1998.

[7] N.Fuhr, K.Großjohann: XIRQL: A Query Language for Information Retrieval in XML Documents, Proceedings ACM-SIGIR Conference. pp.172-180, 2001.

[8] N.Fuhr, K.Großjohann: XIRQL: An XML Query Language Based on Information Retrieval Concepts. (Submitted for publication) http://ls6-www.informatik.uni-dortmund.de/ir/publications/2002/Fuhr_Grossjohann:02.html

[9] N. Fuhr, T. Rölleke: A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems, ACM Transactions on Information Systems, Vol.14 No.1, 1997.

[10] D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, International Conference on Machine Learning (ICML), 1997.

[11] J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM Vol.46 No.5, 1999.

[12] C.D. Manning, H. Schuetze: Foundations of Statistical Natural Language Processing, MIT Press, 1999.

[13] K. Nigam, A. McCallum, S. Thrun, T. Mitchell: Text Classification from Labeled and Unlabeled Documents Using EM, Machine Intelligence Vol.39 No.2/3, 2000.

[14] S. Sizov, S. Siersdorfer, M. Theobald, G. Weikum: The BINGO! Focused Crawler: From Bookmarks to Archetypes, Demo Paper, International Conference on Data Engineering (ICDE), San Jose, 2002.

[15] A. Theobald, G. Weikum: Adding Relevance to XML, 3rd International Workshop on the Web and Databases (WebDB), Dallas, 2000, LNCS 1997, Springer, 2001.

[16] A. Theobald, G. Weikum: The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking, 8th International Conference on Extending Database Technology (EDBT), Prague, 2002.

[17] Y. Yang, J.O. Pedersen: A Comparative Study on Feature Selection in Text Categorization, International Conference on Machine Learning (ICML), 1997.