Statistical Language Models for Intelligent XML Retrieval

Djoerd Hiemstra

University of Twente, Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands d.hiemstra@utwente.nl

1 Introduction

The XML standards that are currently emerging have a number of characteristics that can also be found in database management systems, like schemas (DTDs and XML schema) and query languages (XPath en XQuery). Following this line of reasoning, an XML database might resemble traditional database systems. However, XML is more than a language to mark up data; it is also a language to mark up textual documents. In this chapter we specifically address XML databases for the storage of 'document-centric' XML (as opposed to 'data-centric' XML [2]).

Document-centric XML is typically semi-structured, that is, it is characterised by less regular structure than data-centric XML. The documents might not strictly adhere to a DTD or schema, or possibly the DTD or schema might not have been specified at all. Furthermore, users will in general not be interested in retrieving data from document-centric XML: They will be interested in retrieving *information* from the database. That is, when searching for documents about "web information retrieval systems", it is not essential that the documents of interest actually contain the words "web", "information", "retrieval" and "systems" (i.e., they might be called "internet search engines").

An intelligent XML retrieval system combines 'traditional' data retrieval (as defined by the XPath and XQuery standards) with information retrieval. Essential for information retrieval is ranking documents by their probability, or degree, of relevance to a query. On a sufficiently large data set, a query for "web information retrieval systems" will retrieve many thousands of documents that contain any, or all, of the words in the query. As users are in general not willing to examine thousands of documents, it is important that the system ranks the retrieved set of documents in such a way that the most promising documents are ranked on top, i.e. are the first to be presented to the user.

Unlike the database and XML communities, which have developed some well-accepted standards, the information retrieval community does not have any comparable standard query language or retrieval model. If we look at some practical systems however, e.g. internet search engines like Google and AltaVista, or online search services as provided by e.g. Dialog and LexisNexis, it turns out that there is much overlap in the kind of functionality they provide.

```
1. IT magazines
2. +IT magazine* -MSDOS
3. "IT magazines"
4. IT NEAR magazines
5. (IT OR computer) (books OR magazines OR journals)
6. XML[0.9] IR[0.1] title:INEX site:utwente.nl
```

Fig. 1. Examples of complex information retrieval queries

Figure 1 gives some example queries from these systems. The first query is a simple "query by example": retrieve a ranked list of documents about IT magazines. The second query shows the use of a mandatory term operator '`+`', stating that the retrieved document *must* contain the word IT,[1] a wild card operator '`*`' stating that the document might match "magazine", but also "magazines" or "magazined" and the '`-`' operator stating that we do not prefer IT magazines about MSDOS. The third and fourth query searches for documents in which "IT" and "magazines" occur respectively adjacent or near to each other. The fifth query shows the use of the 'OR' operator, stating that the system might retrieve documents about "IT magazines", "computer magazines", "IT journals", "IT books", etc. The sixth and last query shows the use of structural information, very much like the kind of functionality that is provided by XPath; so "`title:INEX`" means that the title of the document should contain the word "INEX". The last query also shows additional term weighting, stating that the user finds "XML" much more important than "IR".

An intelligent XML retrieval system should support XPath and all of the examples above. For a more comprehensive overview of information retrieval requirements, we refer to Chapter **??** [3].

This chapter shows that statistical language models provide some interesting alternative ways of thinking about intelligent XML search. The rest of the chapter is organised as follows: Section 2 introduces the language modelling approach to information retrieval, and shows how language modelling

[1] Note that most retrieval systems do not distinguish upper case from lower case, and confuse the acronym "IT" with the very common word "it".

concepts like priors, mixtures and translation models, can be used to model intelligent retrieval from semi-structured data. Section 3 reports the experimental results of a preliminary prototype system based on language models. Finally, Section 4 concludes this paper.

2 Language modelling concepts

In this section, we introduce the language modelling approach to information retrieval. First we describe the language modelling approach that resembles traditional vector space information retrieval models that use so-called *tf.idf* weighting. In the sections that follow, we introduce a number of more advanced language modelling constructs like priors, mixtures and translation models, and show how these can be used to model intelligent retrieval from semi-structured data.

2.1 The basic model

The idea behind the language modelling approach to information retrieval [4, 5] is to assign to each XML element X the probability $P(X|q_1, \cdots, q_n)$, i.e., the probability that the element X is relevant, given the query $Q = q_1, \cdots, q_n$. The system uses the probabilities to rank the elements by the descending order of the probabilities. Using Bayes' rule we can rewrite this as follows.

$$P(X|q_1, q_2, \cdots, q_n) \;=\; \frac{P(q_1, q_2, \cdots, q_n|X)P(X)}{P(q_1, q_2, \cdots, q_n)} \tag{1}$$

Note that the denominator on the right hand side does not depend on the XML element X. It might therefore be ignored when a ranking is needed. The prior $P(X)$ however, should only be ignored if we assume a uniform prior, that is, if we assume that all elements are equally likely to be relevant in absence of a query. Some non-content information, e.g. the number of accesses by other users to an XML element, or e.g. the length of an XML element, might be used to determine $P(X)$.

Let's turn our attention to $P(q_1, q_2, \cdots, q_n|X)$. The use of probability theory might here be justified by modelling the process of generating a query Q given an XML element as a random process. If we assume that the current page in this book is an XML element in the data, we might imagine picking a word at random from the page by pointing at the page with closed eyes. Such a process would define a probability $P(q|X)$ for each term q, which would be defined by the number of times a word occurs on this page, divided by the total number of words on the page. Similar generative probabilistic models have been used successfully in speech recognition systems [6], for which they are called "language models".

The mechanism above suggests that terms that do not occur in an XML element are assigned zero probability. However the fact that a term is never observed does not mean that this term is never entered in a query for which the XML element is relevant. This problem – i.e., events which are not observed in the data might still be reasonable in a new setting – is called the *sparse data problem* in the world of language models [7]. In general, zero probabilities should be avoided. A standard solution to the sparse data problem is to interpolate the model $P(q|X)$ with a background model $P(q)$ which assigns a non-zero probability to each query term. If we additionally assume that query terms are independent given X, then:

$$P(q_1, q_2, \cdots, q_n|X) \;=\; \prod_{i=1}^{n} \Big((1-\lambda) P(q_i) + \lambda P(q_i|X) \Big) \tag{2}$$

Equation 2 defines our basic language model if we assume that each term is generated independently from previous terms given relevant XML element. Here, λ is an unknown mixture parameter, which might be set using e.g. relevance feedback of the user. The probability $P(q_i)$ is the probability of the word q_i in 'general query English'. Ideally, we would like to train $P(q_i)$ on a large corpus of queries. In practice however, we will use the document collection to define these probabilities as the number of times the word occurs in the database, divided by the size of the database, measured in the total number of word occurrences. It can be shown by some simple rewriting that Equation 2 can be implemented as a vector space weighting algorithm, where $\lambda P(q_i|X)$ resides on the '*tf*-position' and 1 / $(1-\lambda)P(q_i)$ resides on the '*idf*-position'. The following 'vector-space-like' formula assigns zero weight to words not occurring in a XML element, but ranks the elements in exactly the same order as the probability measure of Equation 2 [4]:

$$P(q_1, q_2, \cdots, q_n|X) \;\propto\; \sum_{i=1}^{n} \log\Big(1 + \frac{\lambda P(q_i|X)}{(1-\lambda) P(q_i)} \Big) \tag{3}$$

Why would we prefer the use of language models over the use of e.g. a vector space model with some *tf.idf* weighting algorithm as e.g. described by [8]? The reason is the following: our generative query language model gives a nice intuitive explanation of *tf.idf* weighting algorithms by means of calculating the probability of picking at random, one at a time, the query terms from an XML element. We might extend this by any other generating process to model complex information retrieval queries in a theoretically sound way that is not provided by a vector space approach.

For instance, we might might calculate the probability of complex processes like the following: What is the probability of sampling eiter "Smith" or "Jones" from the `author` element, and sampling "software" and "engineering" from either the `body` element or from the `title` element? Probability theory will provide us with a sound way of coming up with these probabilities, whereas a vector space approach provides us with little clues on how to

combine the scores of words on different XML elements, or how to distinguish between "Smith" or "Jones", and "Smit" *and* "Jones".

2.2 Mixture models and augmentation weights

Equation 2 shows a model consisting of a *mixture* of two components: the element component $P(q_i|X)$ and a general component $P(q_i)$. In this formula, λ is an unknown mixture parameter. The two-component mixture effectively models the following generation process: What is the probability of sampling the words q_1, q_2, etc. at random from either the XML element X, or from the XML database in total? We might easily extent this to mixtures with an arbitrary number of components, for instance to model the fact that we would prefer XML elements X whose descendent title or abstract (or both) contain the query terms, over elements of which the descendent title or abstract do not contain the query terms. A three-component mixture like this might be described by the following generation process: What is the probability of sampling the words q_1, q_2, etc. from either the XML element X, or from the descendent title element, or from the descendent abstract element? The corresponding probability measure would be:

$$P(q_1, q_2, \cdots, q_n|X) = \prod_{i=1}^{n} \Big(\alpha P(q_i|X) + \beta P(q_i|X, \texttt{title}) + \gamma P(q_i|X, \texttt{abstract}) \Big)$$

Instead of one unknown mixture parameter, we now have to set the value of two unknown mixture parameters: α and β (where $\gamma = 1 - \alpha - \beta$). $P(q_i|X, \texttt{title})$ would simply be defined by the number of occurrences of q_i in the descendent title of X divided by the total number of words in the descendent title of X, and $P(q_i|X, \texttt{title})$ would be defined similarly for the descendent abstract.

In other words, the mixture expresses something similar to the logical OR: if a word q should match either XML element X or a related XML element Y, then the probability is calculated by a mixture. Note that we cannot simply add the probabilities without the mixture parameters, because the two events are not disjoint, that is, a word might match both X and Y.

The unknown mixture parameters play a role that is similar to the augmentation weights described in Chapter **??** and **??** of this book [9, 10]. Both are essentially unknown parameters that determine the importance of XML elements relatively to some related XML elements. The main difference between the augmentation weights and the mixture parameters of the language models, is that the augmentation weights are propagated upwards from a leaf node to its parent, whereas the language models might combine XML elements in an ad-hoc way. Interestingly, as said above, a two-component mixture of an element and the document root, behaves like a vector space approach with *tf.idf* weights.

2.3 Statistical translation and the use of ontologies

Another interesting advanced language modelling construct is the combination of a language model with a statistical translation model [11, 12, 13]. Such a combined model describes a two-stage sampling process: What is the probability of sampling at random a word q from the XML element X, from which we in turn – given that q defines an entry in a probabilistic translation dictionary – sample at random the possible translation c_i? Such a model might be applied to cross-language information retrieval. In cross-language retrieval, the user poses a query in one language, e.g. Dutch, to search for documents in another language, e.g. English documents. For instance, the user enters the Dutch word "college", which has as its possible translations "lecture", "course", "reading" or "class", each possibly with a different probability of the Dutch word given the English word. The system now ranks the elements using the following probability measure:

$$P(c_1, c_2, \cdots, c_n|X) \;=\; \prod_{i=1}^{n} \sum_{q} \Big(P(c_i|q) P(q|X) \Big) \tag{4}$$

In this formula, q sums over all possible words, or alternatively over all words for which $P(c_i|q)$ is non-zero. Given the example above, the sum would include $P(c|\texttt{lecture})\, P(\texttt{lecture}|X)$, $P(c|\texttt{course})\, P(\texttt{course}|X)$, etc. Superficially, this looks very similar to the mixture model. Like the mixtures, the translation models also express something similar to the logical OR: if an element should match either the word "lecture", or the word "course", then we can add the probabilities weighted by the translation probabilities. Note however, that the translation probabilities do not necessarily sum up to one, because they are conditioned on different qs. Adding the probabilities is allowed because the qs are disjoint, i.e. the occurrence of one word can never be "lecture" *and* "course". This is like adding the probabilities of tossing a 5 or a 6 with a fair die, it is impossible to throw a 5 *and* a 6 with only one toss, so we can add the probabilities: $1/6 + 1/6 = 1/3$.

Translation models might play a role in using ontologies for 'semantic' search of XML data as described in Chapter **??** by Schenkel, Theobald and Weikum [14]. They introduce a new operator to express semantic similarity search conditions. As in cross-language retrieval, ontology-based search will retrieve an element that matches words that are related, according to the ontology, to the word in the query. If we follow the approach by Schenkel et al., the ontology might define $P(c_i|q)$ in 4 as the probability of a concept c_i, given a word q.

2.4 Element priors

Maybe the easiest language modelling concept to experiment with is the XML element prior $P(X)$. The prior $P(X)$ defines the probability that the user

likes the element X if we do not have any further information (i.e., no query). An example of the usefulness of prior knowledge is the PageRank [15] algorithm that analyses the hyperlink structure of the world wide web to come up with pages to which many documents link. Such pages might be highly recommended by the system: If we do not have a clue what the user is looking for, an educated guess would be to prefer a page with a high pagerank over e.g. the personal home page of the author of this chapter. Experiments show that document priors can provide over 100 % improvement in retrieval performance for a web entry page search task [16]. The usefulness of some simple priors for XML search in investigated in section refsec:results.

2.5 A note on global word statistics

As said above, ideally, the probability $P(q_i)$ is defined as the probability of the word q_i in 'general query English'. In practice it is estimated on any sufficiently large collection of documents, e.g. quite conveniently, the XML document collection we are currently searching.

Note that this viewpoint is quite different from most other approaches to XML retrieval. The approach presented in Chapter **??** by Grabs and Schek [10] makes a successful effort in reconstructing the global frequencies of the part of the database that is the scope of the query, while still keeping the size of the database reasonably small. The language modelling approach suggests that it is not necessary to reconstruct the total number of times a word occurs in a certain XML element type (or to reconstruct the total number of XML elements of a certain type that contain the word, that is, the so-called 'document frequency' of the word). The model suggests that $P(q)$ is the probability of a word in "general query English": It is the same for all queries, whatever the scope of the query. Furthermore, to avoid the sparse data problem, it should be estimated on as much data as possible, and not – in case of a selective query – on a relatively small part of the database.

Van Zwol [17] compared the effect of fixed global frequencies vs. on-the-fly (fragmented) computation of global frequencies, and concluded that the exact definition of global word statistics has no measurable influence on the performance of an intelligent XML retrieval system. The advantage of fixed global frequencies over on-the-fly computation of global frequencies is that the former approach allows for simpler and more efficient query plans.

2.6 Discussion

This section presented some interesting new ways of thinking about intelligent XML retrieval. Whether these approaches perform well in practice, has to be determined by experiments on benchmark test collections as e.g. provided by INEX. Preliminary experiments are described in the next chapter.

However, experience with language models on other tasks look promising. Recent experiments that use translation models for cross-language retrieval

[12], document priors for web search [16], and mixture models for video retrieval [18] have shown that language models provide top performance on these tasks. Other systems that use language models for intelligent XML retrieval are described by Ogilvie and Callan [19], and by List and De Vries [20].

3 Preliminary evaluation on INEX

In this section we describe a preliminary prototype system for intelligent XML retrieval, based on the language modelling approach described above. The system is evaluated using the INEX testbed. We briefly describe the system, the tasks and evaluation procedure, the experimental setup and research questions, and finally the experimental results.

3.1 A first prototype

The preliminary prototype should in principle support 'all of XPath and all of IR'. In order to support XPath, the system should contain a complete representation of the XML data. The system should be able to reproduce any part of the data as the result of the query. For XPath we refer to [21].

For our first prototype we implemented the XML relational storage scheme proposed in Chapter **??** by Grust and Van Keulen [22]. They suggest to assign two identifiers (id) to each instance node: one id is assigned in pre-order, and the other in post-order. The pre and post order assignment of XML element ids provides elegant support for processing XPath queries, forming an alternative to explicit parent-child relations which are often used to store highly structured data in relational tables [23, 24, 25].[2]

Note that pre and post order assignment can be done almost trivially in XML by keeping track of the order of respectively the opening and closing tags. Since we are going to build a textual index for content-based retrieval, we assign an id (or position) to each word in the XML text content as well. The word positions are used in a term position index to evaluate phrasal queries and proximity queries. Interestingly, if we number the XML data as a linearised string of tokens (including the content words), we obey the pre/post order id assignment, but we also allow the use of theory and practice of region algebras (see Chapter **??** [26]). For a more detailed description of the storage scheme, we refer to [27].

3.2 The INEX evaluation

INEX is the Initiative for the Evaluation of XML Retrieval. The initiative provides a large testbed, consisting of XML documents, retrieval tasks, and

[2] Actually, Grust et al. [22] store the id of the parent as well. Similarly, Schmidt et al. [25] add a field to keep track of the order of XML elements; here we emphasise different view points.

relevance judgements on the data. INEX identifies two tasks: the content-only task, and the content-and-structure task.

The content-only task provides 30 queries like the following example: `//*[. =~ "computational biology"]` ("XPath & IR" for: any element about "computational biology"). In this task, the system needs to identify the most appropriate XML element for retrieval. The task resembles users that want to search XML data without knowing the schema or DTD.

The content-and-structure task provides 30 queries like the following: `//article[author =~ "Smith|Jones" and bdy =~ "software engineering"]` ("XPath & IR" for: retrieve articles written by either Smith or Jones about software engineering). This task resembles users or applications that *do* know the schema or DTD, and want to search some particular XML elements while formulating restrictions on some other elements.

For each query in both tasks, quality assessments are available. XML elements are assessed based on *relevance* and *coverage*. Relevance is judged on a four-point scale from 0 (irrelevant) to 3 (highly relevant). Coverage is judged by the following four categories: N (no coverage), E (exact coverage), L (the XML element is too large), and S (the XML element is too small).

In order to apply traditional evaluation metrics like precision and recall, the values for relevance and coverage must be quantised to a single quality value. INEX suggests the use of two quantisation functions: Strict and liberal quantisation. The strict quantisation function evaluates whether a given retrieval method is capable of retrieving highly relevant XML elements: it assigns 1 to elements that have a relevance value 3, and exact coverage. The liberal quantisation function assigns 1 to elements that have a relevance value of 2 and exact coverage, or, a relevance value of 3 and either exact, too small, or too big coverage. An extensive overview of INEX is given in Chapter **??** of this volume [28].

3.3 Experimental setup and research questions

We evaluate a system that only has limited functionality. First of all, we assume that $\lambda = 1$ in Equation 2, so we do not have to store the global word statistics. The system supports queries with a content restriction on only one XML element, so the example content-and-structure query in the previous section is not supported: Either the restriction on the `author` tag, or the restriction on the `bdy` tag has to be dropped. The system supports conjunction and disjunction operators, which are evaluated as defined by Equation 4 where the translation probabilities were set to 1. All queries were manually formulated from the topic statements.

The experiments are designed to answer the following research question: Can we use the prior probability $P(X)$ (see Equation 1) to improve the retrieval quality of the system? We present three experiments using the system described in this paper, for which only the prior probabilities $P(X)$ differ. The baseline experiment uses a uniform prior $P(X) = c$, where c is some constant

value, so each XML element will have the same a priori probability of being retrieved. A second experiment uses a length prior $P(X) = \textit{number of tokens in the XML element}$, where a token is either a word or a tag. This means that the system will prefer bigger elements, i.e. elements higher up the XML tree, over smaller elements. A third experiment uses a prior that is somewhere in between the two extremes. The prior is defined by $P(X) = 100 + \textit{number of tokens in the XML element}$. Of course, the priors should be properly scaled, but the exact scaling does not matter for the purpose of ranking. We hypothesise that the system using the length prior will outperform the baseline system.

3.4 Evaluation results

This section presents the evaluation results of three retrieval approaches (no prior, 'half' prior, and length prior) on two query sets (content-only, and content-and-structure), following two evaluation methods (strict and liberal). We will report for each combination the precision at respectively 5, 10, 15, 20, 30 and 100 documents retrieved.

Strict evaluation

Table 1 shows the results of the three experiments on the content-only queries following the strict evaluation. The precision values are averages over 22 queries. The results show an impressive improvement of the length prior on all cut-off values. Apparently, if the elements that need to be retrieved are not specified in the query, users prefer larger elements over smaller elements.

Table 1. Results of content-only (CO) runs with strict evaluation

precision	no prior	'half' prior	length prior
at 5	0.0455	0.0455	0.1909
at 10	0.0364	0.0455	0.1591
at 15	0.0303	0.0424	0.1394
at 20	0.0341	0.0364	0.1318
at 30	0.0364	0.0424	0.1318
at 100	0.0373	0.0559	0.1000

Table 2 shows the results of the three experiments on the content-and-structure queries following the strict evaluation. The precision values are averages over 28 queries. The baseline system performs much better on the content-and-structure queries than on the content-only queries. Surprisingly, the length prior again leads to substantial improvement on all cut-off values in the ranked list.

Table 2. Results of content-and-structure (CAS) runs with strict evaluation

precision	no prior	'half' prior	length prior
at 5	0.1929	0.2357	0.2857
at 10	0.1964	0.2321	0.2857
at 15	0.1976	0.2333	0.2714
at 20	0.1929	0.2232	0.2589
at 30	0.1786	0.2060	0.2607
at 100	0.0954	0.1107	0.1471

Liberal evaluation

Table 3 shows the results of the three experiments on the content-only queries using the liberal quantisation function defined above for evaluation. The precision values are averages over 23 queries. Again, the results show a significant improvement of the length prior on all cut-off values.

Table 3. Results of content-only (CO) runs with liberal evaluation

precision	no prior	'half' prior	length prior
at 5	0.1130	0.1391	0.4261
at 10	0.0957	0.1304	0.3609
at 15	0.0957	0.1333	0.3304
at 20	0.1000	0.1152	0.3000
at 30	0.1087	0.1232	0.2812
at 100	0.0896	0.1222	0.2065

Table 4. Results of content-and-structure (CAS) runs with liberal evaluation

precision	no prior	'half' prior	length prior
at 5	0.2429	0.2929	0.4000
at 10	0.2286	0.2823	0.3750
at 15	0.2262	0.2881	0.3738
at 20	0.2268	0.2821	0.3607
at 30	0.2179	0.2583	0.3595
at 100	0.1279	0.1571	0.2054

Table 4 shows the results of the three experiments on the content-and-structure queries following the liberal evaluation. The precision values are averages over 28 queries. The length prior again shows better performance

on all cut-off values. Note that the content-only task and the content-and-structure task show practically equal performance if the liberal evaluation procedure is followed.

4 Conclusion

In this paper we described in some detail the ideas behind the language modelling approach to information retrieval, and suggested several advanced language modelling concepts to model intelligent XML retrieval. We presented an preliminary implementation of a system that supports XPath and complex information retrieval queries based on language models. From the experiments we conclude that it is beneficial to assign a higher prior probability of relevance to bigger fragments of XML data than to smaller XML fragments, that is, to users, more information seems to be better information.

Whether the advanced modelling contructs presented in Section 2 will in fact result in good retrieval performance will be evaluated in the CIRQUID project (Complex Information Retrieval Queries in a Database). In this project, which is run in cooperation with CWI Amsterdam, we will develop a logical data model that allows us to define complex queries using advanced language modelling primitives.

Acknowledgements

The research presented in this chapter was funded in part by the Netherlands Organisation for Scientific Research (NWO project number 612.061.210).

References

1. Blanken, H., Schek, H., Weikum, G., Grabs, T., Schenkel, R., eds.: Intelligent XML Retrieval. Lecture Notes in Computer Science (LNCS). Springer-Verlag (2003)
2. Bourret, R.: XML and databases. Technical Report (2003) http://www.rpbourret.com/xml/XMLAndDatabases.htm
3. Rys, M.: Full-Text Search with XQuery: A status report. In [1] (in this volume)
4. Hiemstra, D., Kraaij, W.: Twenty-One at TREC-7: Ad-hoc and cross-language track. In: Proceedings of the seventh Text Retrieval Conference TREC-7, NIST Special Publication 500-242 (1998) 227–238
5. Miller, D., Leek, T., Schwartz, R.: A hidden Markov model information retrieval system. In: Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99) (1999) 214–221
6. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. In Waibel, A., Lee, K., eds.: Readings in speech recognition. Morgan Kaufmann (1990) 267–296

7. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
8. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management **24** (1988) 513–523
9. Fuhr, N., Großjohann, K., Kriewel, S.: A query language and user interface for XML information retrieval. In [1] (in this volume)
10. Grabs, T., Schek, H.: Flexible information retrieval on XML documents. In [1] (in this volume)
11. Berger, A., Lafferty, J.: Information retrieval as statistical translation. In: Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99). (1999) 222–229
12. Hiemstra, D., Jong, F. de: Disambiguation strategies for cross-language information retrieval. In: Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries (ECDL). (1999) 274–293
13. Xu, J., Weischedel, R., Nguyen, C.: Evaluating a probabilistic model for cross-lingual information retrieval. In: Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR'01). (2001) 105–110
14. Schenkel, R., Theobald, A., Weikum, G.: Ontology-enabled XML search. In [1] (in this volume)
15. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30** (1998) 107–117
16. Kraaij, W., Westerveld, T., Hiemstra, D.: The importance of prior probabilities for entry page search. In: Proceedings of the 25th ACM Conference on Research and Development in Information Retrieval (SIGIR'02). (2002) (in this volume).
17. Zwol, R. van: Modelling and searching web-based document collections. PhD thesis, Centre for Telematics and Information Technology, University of Twente (2002)
18. Westerveld, T., Vries, A. de, van Ballegooij, A., Jong, F. de, Hiemstra, D.: A probabilistic multimedia retrieval model and its evaluation. Eurasip Journal on Applied Signal Processing 2003(2) (2003) 186–198
19. Ogilvie, P., Callan, J.: Language models and structured document retrieval. In: Proceedings of the first Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), ERCIM Workshop Proceedings (2003)
20. List, J., Vries, A. de: CWI at INEX 2002. In: Proceedings of the first Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), ERCIM Workshop Proceedings (2003)
21. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., Simeon, J.: XML Path language (XPath) 2.0. Technical report, World Wide Web Consortium (2002) http://www.w3.org/TR/xpath20/
22. Grust, T., Keulen, M. van: Tree awareness for relational DBMS kernels: Staircase join. In [1] (in this volume)
23. Florescu, D., Kossmann, D.: A performance evaluation of alternative mapping schemes for storing xml data in a relational database. In: Proceedings of the VLDB'99. (2001) 105–110
24. Keulen, M. van, Vonk, J., Vries, A. de, Flokstra, J., Blok, H.: Moa: extensibility and efficiency in querying nested data. Technical Report 02-19, Centre for Telematics and Information Technology (2002)
25. Schmidt, A.R., Kersten, M.L., Windhouwer, M.A., Waas, F.: Efficient Relational Storage and Retrieval of XML Documents. In: The World Wide Web and

Databases - Selected Papers of WebDB 2000. Volume 1997 of Lecture Notes in Computer Science (LNCS)., Springer-Verlag (2000) 137–150
26. Vries, A. de, List, J., Blok, H.: The multi-model DBMS architecture and XML information retrieval. In [1] (in this volume)
27. Hiemstra, D.: A database approach to content-based XML retrieval. In: Proceedings of the first Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), ERCIM Workshop Proceedings (2003)
28. Kazai, G., Gövert, N., Lalmas, M., Fuhr, N.: The INEX evaluation initiative. In [1] (in this volume)