Marie-Francine Moens, 'XML Retrieval Models for Legislation' in T. Gordon (ed.), *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference*. Amsterdam: IOS Press, 2004, pp. 1-10.

XML Retrieval Models for Legislation

Marie-Francine Moens
Interdisciplinary Centre for Law and Information Technology (ICRI)
Katholieke Universiteit Leuven
Tiensestraat 41 B-3000 Leuven,Belgium

Abstract. Legislation contains text-rich documents and is increasingly marked with XML tags. The XML markup can - among other uses - be exploited to more precisely answer free information queries. In this article we report on different XML retrieval models we explicitly designed for the retrieval of legislation and which are based on the vector space model and the probabilistic language model. In addition search data structures are designed for legislative databases that support these retrieval models. We show that the models provide more advanced access to the content of statutes.

1 Introduction

Legislation typically involves structured information including the division of a statute in for instance titles, chapters, sections and articles, and the typical metadata (e.g., indication of the date of enactment, the area of applicability and references to other statutes) that are assigned to the statute or its parts. Additionally, legislation contains large parts of unstructured information found in the natural language texts. The structured information is increasingly tagged with markup languages such as *XML* (*Extensible Markup Language*). The use of such a markup language makes it possible that documents can be easily interchanged between institutions and systems, and that the markups are interpretable across the use of different software.

From way back, *legal information retrieval* is an important information technology application [2], and it has an increasing significance. Legislative texts are currently accessible through specifically designed portal sites owned by governments or private institutions. The search engines that operate on the legal documents usually offer a full-text search (i.e., every word of the text including some metadata is indexed and can be searched). A full-text search is popular because it provides a flexible information access: The user can build any search query. When information is retrieved by using a full text search, the resulting answers of a search are ranked according to relevance to the query. The current search engines that operate on legislation allow for an extra selection of the content through filling out specific fields that represent specific structured content of the document (e.g., statute title, number of an article, etc.).

There is a recent trend in information retrieval to take into account the structured information of documents (e.g., as marked by XML) and especially the hierarchical logical document structure when generating the answer to a query and when computing the *relevance ranking*. This has several advantages. The use of the document structure allows generating a more precise answer to an information query. Instead of returning the complete document as the answer, a structural element or several elements are given. Such an approach meets the current need of users of legal information systems, who demand more precise answers to information queries [8]. Moreover, research has only recently started to exploit the relationships between structured elements in ranking functions. For instance, depending on where a

search term occurs in the hierarchy of document elements, its weight in a ranking function might be different.

The potential of integrating document structure into a retrieval model for interrogating legislation has not been examined yet. The research presented here studies how the structured information in legislation helps in formulating a precise answer to a user's search request. It also designs suitable and innovative relevance ranking functions that take into account the typical structural properties of legislative documents. Traditionally, search engines use auxiliary data structures that hold document information in order to speed up the search at query time. We designed also a data structure for legislative documents that supports the proposed relevance ranking functions. Our research is implemented in a proof-of-concept system and tested on Belgian legislation.[1]

The paper is organized as follows. First, we outline current XML retrieval models and explain why these models might be useful for the retrieval of legislation. The following sections respectively describe our proposed ranking functions and the auxiliary search data structures. Before the conclusion we outline some experiments.

2 XML Retrieval Models

Recently, there is a significant increase in the use of XML (Extensible Markup Language) to represent information. Often documents have markups that signal their structure or certain metadata that characterize content.

Two document modelling approaches are usually contrasted [10]. *Data-centric documents* have a regular and strict structure, and the content is usually not mixed with large stretches of unstructured information such as free text. This is the type of information usually stored in a relational or object-oriented database. *Document-centric documents* are characterized by a less regular structure, and they often contain considerably large text fragments apart from the structured content. The documents of this latter category might not strictly adhere to a DTD or XML schema, or possibly the DTD or schema might not have been specified at all. Furthermore users of the documents of this latter category will generally not be interested in retrieving data. Instead, they are interested in retrieving information from these documents that is relevant for an information need.

Legislative documents that are marked up with XML tags can be considered as an example of *document-centric* objects. Apart from some structured data, statutes contain for the largest part unstructured free text. Users can search for a specific statute or article in a legislative database, or they can formulate a free query by which they want to search for information that is present in the statutes.

For information retrieval from document-centric XML data, the research community has exhibited a large interest in XML retrieval models. In information retrieval a representation is made from each document, which at query time is matched with the representation of the query. The matching function often computes a ranking score that is used to rank the documents according to relevance to the query. A *retrieval model* (e.g., vector space model, probabilistic language model) is defined by the query representation, the document representation and the function that is used to match a document and a query. The retrieval model of an information retrieval system is often very different from the one of a database system. While the latter retrieval model relies on a *deterministic matching* of query data and object data in the database, the former incorporates an element of uncertainty, i.e., documents can be retrieved even if their content representation does not exactly match the one of the query. When retrieving data from a database and one of the query conditions is not fulfilled by a data

[1] The research is part of the E-Lex project, which studies the technical requirements of digital legislation and which is sponsored by FWO Vlaanderen (Grant Nr. G.0330.01).

object, the object will not be retrieved. Typical query languages such as Xpath and Xquery for retrieving information from XML documents have been designed. These languages are inspired by the SQL (Structured Query Language) language and exploit Boolean retrieval, i.e., a deterministic matching of query terms and markup information. Such an approach does not allow the ranking of documents according to the relevance to the query. Typical for an information retrieval model is the relevance ranking of the retrieval results that is the consequence of a *non-deterministic or probabilistic matching*.

Traditional information retrieval models completely ignore document structure or latent ontological information that is expressed by the element and attribute labels of XML documents, DTDs and XML schemas, missing a great opportunity for more effective search. So, it is of no surprise that the integration of structured and unstructured information in the design of retrieval models has recently received a large attention. So-called *XML retrieval models* preserve the non-deterministic matching of query and document, but exploit the document structure to more correctly formulate an answer to an information query [3]. XML retrieval models satisfy a general desire of improving the recall and precision of information retrieval.

As it will be shown below, XML retrieval models have a large potential for the *retrieval of legislation*. Current retrieval systems that access legislative databases offer the possibility of a full text search, i.e., every term can act as a search term, and return a ranked list of information answers. This answer list can be filtered by a deterministic fulfilment of extra conditions set by the structured information found in the statutes (e.g., the name of a statute, the domain of law). Legislative documents increasingly are marked-up with XML [1, 4, 6]. Thus, on one hand we can search the free texts of legislation and on the other hand the structured information, but no attempts have yet been made to incorporate the structured information in a ranking model in order to more precisely retrieve information for the legal problem at hand. An XML retrieval model that ranks legislative texts according to relevance to a query has not yet been designed.

3 The Potential of XML Retrieval Models for the Retrieval of Legislation

Legislative documents have a *hierarchical structure* in which sections with detailed content are nested in larger sections. Belgian legislation is, for instance, divided into titles, chapters, sections and articles (Figure 1). Articles can be further split up in paragraphs and subparts. Articles are grouped in higher order units when they treat related content. It is not uncommon to have lists as elements of articles and of an article's subdivisions. Statute parts on whatever level of detail might have own headings and descriptions. Moreover, specific *content elements* might be tagged (e.g., a definition, a reference). The aim of the XML retrieval models is to incorporate the knowledge on the document structure and on specific content elements in computing the relevance of an information unit. Apart from the use of specifically designed XML query languages (e.g., that can be used to specify that a certain query term has to occur in the heading of a chapter), the structured markup can be exploited in many other ways. Because the structure of an XML document is tagged, the document can be broken up in different units (of varying length when these units are nested) that can be returned as an answer to the information need. The user of a legislative retrieval system can retrieve a complete chapter, an article or even a list element of a paragraph depending on what is most relevant as an answer to the query.

Moreover, when evidence of being relevant to an information query can be found on different levels of granularity in the document especially in the broader or narrower segments of the document hierarchy, this evidence can be exploited in a ranking function.

How to incorporate structured information in the *ranking function* of a retrieval model that is especially designed for the retrieval of legislation is the main focus of this article.

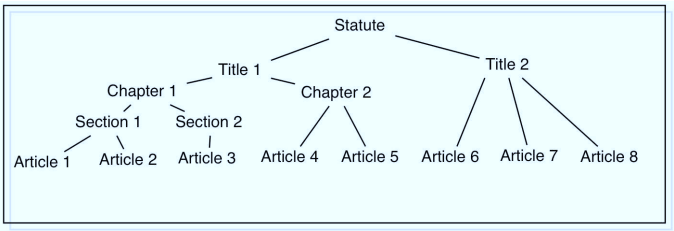

Figure 1: An example structure of a statute

The retrieval models that we design do not prevent the use of XML query languages, but we will not cover this aspect in this article (we refer here to [3] for more information). Because speed in answering an information question is important, information retrieval systems do not search text online, but search auxiliary data structures. In these data structures indexing representations are stored and searched at query time. In a traditional full text search these representations are composed of index terms and their address (usually file name or URL, and position in the file). XML-retrieval models want to incorporate information on the document structure for a better retrieval of the information, and appropriate representations and data structures are being researched. Which *search structures* can be developed for the retrieval of XML-tagged legislation is also a point of attention in this article.

4 Ranking Functions

The purpose of a retrieval or ranking model is to provide a score by which a retrieval unit can be ranked according to its estimated relevance to the query. This score is computed by the ranking function. A retrieval unit in the context of legislation is defined as any statute part that can form the answer to the information query (e.g., article, chapter). The smallest retrieval unit that is considered in our ranking models is the article. However, the proposed retrieval models can be adapted to smaller retrieval units such a paragraph, a subpart or even a list item. In an XML retrieval model, retrieval units are not necessarily mutually exclusive. One unit is often nested into a larger unit.

The research here focuses on queries composed of search terms. Because legislation deals with documents that are largely composed of natural language texts of which the semantics cannot always be unambiguously defined, and users of legislative databases formulate queries that do not unambiguously and exhaustively define their information need, the ranking function should incorporate these uncertainties.

In an XML retrieval model the content overlap of a retrieval unit with the query is usually measured in a classical way. Various algorithms that exploit term overlap between query and the retrieval unit can be used for this purpose. Because the XML documents are usually hierarchically structured, knowledge of the structure can be exploited in various ways in the ranking computations and terms might be weighted differently depending on the structural unit they occur in or depending on the depth in the tree of this structural unit. Besides using this macro-level structure of XML documents, on a micro-level certain specific information might be marked (e.g., a definition, reference to another statute or article). Very little research has been done to incorporate this information in a flexible non-deterministic ranking function. In this article we focus on novel ways of integrating knowledge on the macro-document structure of legislation into the ranking function.

For the ranking function we have designed and implemented *two retrieval models* and their variants. The first model regards a *vector space retrieval model* that is adapted to incorporate macro-level structural knowledge of XML-tagged legislation. The second model is

a *language model* that is a probabilistic model of several basic language model components that model the typical document structure of legislation. In the models we have to take into account that the structure of legislation might change from one statute to the other and even within the statute different structures of subtrees might apply (e.g., in Figure 1: Title 1 might be divided in chapters, section and articles, while Title 2 only contains articles).

For both models, the texts of the statutes are *preprocessed*. They are currently tokenized in words, stopwords (i.e., function words that do not bear on content) are removed, and all words are normalized to lower-case letters. A more sophisticated preprocessing of the texts might be employed in the future (e.g., the detection of concepts such as in [12]).

4.1 Vector Space Model

In the *vector space model*, retrieval units (here statute parts) and queries are represented as vectors in an $n-$dimensional vector space with the relevance of a document to a query computed as a distance measure (where$n$is the number of index terms in the text collection). The vector coefficients could take on numeric values indicating the weight or importance of the index terms. For simplicity, we use here a classical *tf* x *idf* weighting scheme for representing the query and retrieval unit texts. The *term frequency* (*tf*) is computed as the number of times a term occurs in the query or statute part, possibly augmented by the adapted term frequency in neighbouring statute parts (see below). The *inverse document frequency* (*idf*) is computed from a large reference corpus. The purpose of the *idf* metric, which is a number that is inversely proportional to the number of documents in which a term occurs, is to downweight common words. To compute the distance between a query and a retrieval unit vector, we use the classical cosine function, which also takes into account the length normalization of a text.

The *relevance ranking* retrieval model proposed by [9] computes the relevance with each retrievable component (e.g., chapter, section, article). A retrievable component is represented as a set of nodes in a tree. Each retrieval unit is represented as a vector of terms reflecting its text. When computing the relevance of a parent retrieval unit (e.g., section) the vector of the parent node is augmented with the weights of terms of children (Figure 2). Weights from children terms are promoted upwards in the tree but at the same time downweighted by a priori defined factor (e.g., 0.6). The weights then are combined using probabilistic rules using the "." operator for the logical conjunction and the "+" operator for the logical disjunction (e.g., for the term *theft* in section 1 of Figure 2: 0.5 + 0.6 *0.8 – 0.5 * 0.6 * 0.8 where the last term refers to the probability of the common occurrence of *theft* in section 1 and article 2). The philosophy behind the downweighting is the following. If the vectors of the retrieval units of varying sizes are compared to the query, a large statute component has more chances to be relevant (even with the cosine length normalization), while the user of the retrieval system wants a shortest possible, precise answer to the query.

Note that in this approach leave nodes (in our case articles) are treated similarly irrespective of whether the retrieval unit has parent units. In legislation, the parent units contain text that is common for the children and the more closer the parent, the more connected the text is with content of the child text. This gives the opportunity to design novel XML retrieval models for the retrieval of legislation. In a simple baseline approach, we can just concatenate the terms that are exclusive for a parent (e.g., de title of a chapter) to the text of the children and construct the vector accordingly. In a second more sophisticated approach we can demote parent term weights downwards the tree and combine them with the term weights of a child retrieval unit, but the parent term weight is downweighted by a priori defined factor and using the same probabilistic rules as above. In a first instance we consider the texts that are exclusive for the parent node (e.g., titles) (in Figure 3 (1) the weight of *theft* is computed as

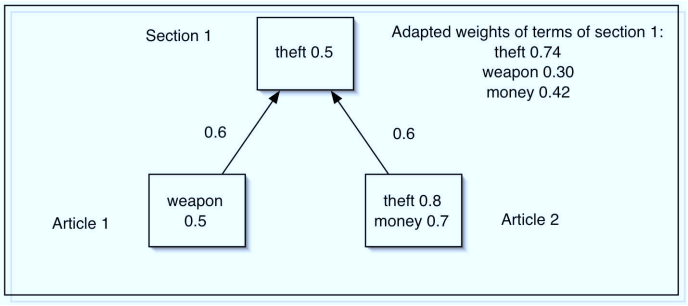

Figure 2: Example of the influence of child term weights on parent term weights.

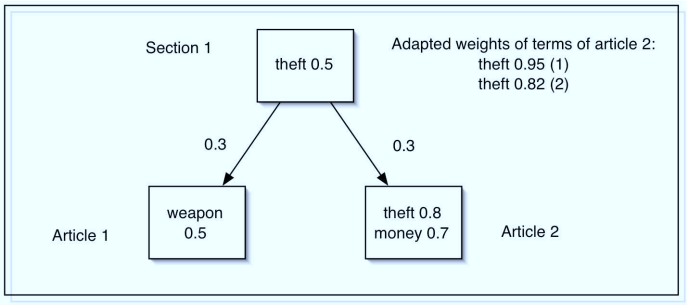

Figure 3: Example of the influence of parent term weights on child term weights.

0.8 + 0.5 * 0.3). However, legislative texts naturally group articles into sections or broader components when they treat the same topic, and contain very related content. In information retrieval it is common to expand terms with related terms (often learned from related documents) as a means to improve the recall of the retrieval. So, we experiment with a model that considers all terms of the parents of a retrieval unit to adapt the weights of the retrieval unit vector (i.e., like for instance expanding the terms of an article vector with related terms from the whole parent section) (in Figure 3 (2) the weight of *theft* is computed as 0.8 + 0.5 * 0.3 – 0.8* 0.5 * 0.3).

4.2 Language Model

The second model regards a *language modelling approach*. In recent years statistical language modelling has become a major retrieval modelling approach [7]. In such an approach a document is viewed as a model and a query as a string of text randomly sampled from this model. Most of the approaches rank the documents in the collection by the probability that the query $Q$is generated given a document D: $P(Q|D)$, i.e., the probability that the query Q would be observed during repeated random sampling from the model of document D. In the language model the query is seen as a set of query terms that are assumed to be conditionally independent given the document, and thus the query probability can be represented as a product of the individual term probabilities:

$$P(q_{1,}q_{2,..}, q_m|D) - \prod_{i=1}^{m} (P(qi|D)) \quad (1)$$

Where $q_i$is the i^{th} query term in the query, and $P(q_i|D)$ is specified by the document language model. Computing the probability that a query term appears in document D(1) might yield a zero probability. So, a more sophisticated document model is usually chosen

<statute>^{0}<title 1>^{1}KB^{2} <chapter 1>^{3} algemene^{4} bepalingen^{5} <section 1>^{6} diefstal^{7} <article 1>^{8} voertuigen^{9} </article 1>^{10} alarm^{11} <article 2>^{12} voertuig^{13} </article 2>^{14}</section 1>^{15}<section 2>^{16} douane^{17} <article 3>^{18} douane^{19} controle^{20} </article 3>^{21}</section 2>^{22}</chapter 1>^{23}<chapter 2>^{24} voorzieningen^{25} <article 4>^{26} wet^{27} </article 4>^{28}<article 5>^{29} besluit^{30} </article 5>^{31} </chapter 2>^{32}</title 1>^{33}<title 2>^{34} diefstal^{35} onderdelen^{36} <article 7>^{37} radio^{38} </article 7>^{39}<article 8>^{40} banden^{41} </article 8>^{42}<article 9>^{43} auto^{44} onderdeel^{45} </article 9>^{46}</title 2>^{47}</statute>^{48}

Figure 4: The XML example statute of figure 1 shown as a sequence of tokens. Only a few preprocessed text words are included. The numbers indicate the token positions in the statute.

that allows for a *smoothing of the probabilities*. Usually the collection probabilities of a term are used to smooth the document probabilities yielding the following model:

$$P(q_1,q_{2,..},q_m|D) = \prod_{i=1}^{m} (\lambda(P(qi\,|D) + (1-\lambda)P(qi\,|C)) \quad (2)$$

where $C$is the collection of documents. According to [10] $P(q_i|C)$ should be estimated from a text collection as large as possible. The *interpolation weight* λ is set empirically or learned from a training corpus (possibly with the Expectation-Maximization algorithm). Our language model for ranking is inspired on this approach, but incorporates the macro-level structure of legislation. Here also, it is possible that a term of the query does not occur in the XML unit to be ranked. However, the fact that a term is never observed does not mean that the term is never entered in a query for which the XML retrieval unit is relevant. In order to avoid zero probabilities in the model, one uses the structured format when building a document language model, and interpolates the model *P(q|X)* with other background models based on the structure of the document as it was suggested by [10]. We expand the classical two-component mixture used in information retrieval (2) by a multi-component mixture model. That allows us, for instance, to model the fact that we would prefer an XML unit X (here e.g., article) whose parents (e.g., title, chapter, section) contain query terms, over units of which the parent sections or chapters do not contain these query terms. For the retrieval of legislation, this is especially relevant, because we know for sure that if parent units are present, they explicitly group related articles about a same subject. For instance, we build a language model that takes into account the texts of the section, chapter and title to which an article belongs. A five component mixture like this is described by the following generation process.

$$P(q_1, q_{2,..}, q_m \quad |X) =$$

$$\prod_{i=1}^{m} (\lambda P(qi\,|X) + \alpha P(qi\,|S) + \beta P(qi\,|Ch) + \gamma P(qi\,|T) + (1-\lambda-\alpha-\beta-\gamma)P(qi\,|C) \quad (3)$$

where Tstands for title, *Ch* for chapter and $S$for section to which the article belongs, and $\lambda + \alpha + \beta + \gamma$ = 1. $P(q_i|S)$, $P(q_i|Ch)$ and $P(q_i|T)$ are simply defined by the number of occurrences of $q_i$in the respective statute part. Depending on the type of retrieval unit that should be ranked different interpolation weights can be chosen. For instance, if X is the article to be ranked, λ can be chosen quite high with α, β and γ having decreasing values. $P(q_i|C)$ is equal for all queries and acts just as a smoothing factor in a probability estimation making that the final probability estimate is not zero when a query term does not occur in the article, section, chapter or title. With such a mixture model, we can model the interactions between the various parts in the structure of the statute. We can here use different interpolation weights for the models of parent sections in statutes that model different influences on the ranking of the article. In another model, we consider the text of the parent that is common for all its

children, where we would set the influence by the interpolation weights for parent sections quite high. Or, we use the complete parent section to compute the probability that a query term occurs. By this approach we consider a parent unit as a cluster of related documents the terms of which are used to expand the terms of the article that is actually ranked (cf. [11]). This last variant of the language modelling approach is used as a recall enhancing means.

Although various interactions between statute sections are modelled with the language modelling approach, it is less flexible than the above vector space model. First, we have to know in advance the possible sections that a statute is composed of. The above vector space model can be dynamically adapted to any kind of hierarchical structure of the statute. Secondly, the influence of children when a parent retrieval unit (e.g., chapter) is ranked, is very hard to model a priori, because of the varying amount of children (e.g., articles and sections of the chapter).

A language model does not need a *tf* x *idf* (or any other) term weighting scheme. The model itself uses the notions of the term frequency and of the collection frequency on which the *idf* metric is based.

Another advantage of the language modelling approach is that we can easily integrate ontological knowledge [5] in the statute model that offers the possibility to translate a term of the text into a synonym or hypernym term and to incorporate into the model the probabilities of these translations. We did not yet exploit this direction, but refer to [10] for more details.

5 Data Search Structures

In any retrieval system of a respectable size, the documents are not searched on-line, but additional *data search structures* that contain indexed representations are searched at query time. These indices provide search speed at the expense of additional overhead for storage. The problem is to define an index structure that models the legislative documents, that provides efficient searching with a restricted storage. In traditional retrieval models, the indexing terms and their addresses (usually document id and term position) are stored in a so-called inverted file. XML documents contain many overlapping retrieval units. Using the traditional data structure model would mean storing many redundant term positions when covering all possible retrieval units.

For storing the legislative indices, we currently use a *text region approach* similar to the one described in [13]. The XML tagged statute (example see figure 4) is viewed as a sequence of tokens where tokens are opening and closing tags (indicating the XML regions) as well as the pre-processed text content. Currently, we extract several indexing tables for each statute. A word index W stores all the index terms and their position. The node-index N stores the set of all XML regions and their statute delimiters in terms of position of the opening and closing tag. We added here also the file pointer positions in the statute for easy retrieval of the full text of the region. Both tables allow computing the term weight in a text region or estimating the probability of a term in the region. A relationship index Rcontains all the binary relationships between a parent and its immediate child where a parent and child are identified by their position. This last index allows dynamically adapting the term weights by taking into account the broader context of parent text regions or the narrower context of child regions. As in [13] an attribute index Acan be added that stores regions and their attribute values and a table that stores all unique paths linked to the binary relations. The latter is useful for efficient processing of structured queries.

6 Experiments

We performed a limited number of *experiments* with statutes in the domain of Belgian criminal law written in Dutch. Inverse document frequency weights and the collection frequencies needed for the probability estimates were computed from a general corpus of Dutch texts. The documents include Royal Decrees, Decrees of the Flemish government, regular laws and some international treaties. Unfortunately we do not have a model test corpus, queries and model answers for testing the recall and precision of the retrieval of legislation. Currently, we are testing the value of the indexing tables for the efficiency of ranking all possible answers for an information query, which might lead to the storage of extra indexing information that avoids computations at the time of querying.

7 Conclusions and Future Work

We now have richer document descriptions and formats for legislative documents that are described with description languages such as XML. Information retrieval engines should be able to cope with the complexity of the new legislative standards so as to fully exploit the potential of these representations and to provide new functionalities for information access. For example, users may need to access some specific document part, navigate through complex structured collections, queries may address both metadata and textual content and users want more to the point answers for their information queries. It is especially this last task that we have focused upon in this paper. We have designed and implemented several XML retrieval models (based on the vector space model and the language model) that exploit the XML marked structure of legislative text to find the answers to an information query. The models allow statute components to be retrieved at different levels of granularity and to integrate the content of parent and child components in the ranking function. If we have an information question that should be answered by the retrieval of relevant legislation, it is logical that the relevant information units can take different sizes and forms. For very specific queries, a single statute paragraph may contain the right answer, whereas more general questions could be answering best by returning a whole chapter. Some of the models that we presented also allow that a group of related statute components (e.g., the articles in a section or chapter) contribute to the overall relevance of a single component (e.g., an article). In the near future we plan to evaluate the models more carefully both with regard to retrieval quality and efficiency.

A lot of interesting work can be done with regard to the design and implementation of an answer interface that exploits and perhaps visualizes the structure of the statutes and their retrieved components. In traditional document retrieval, the retrievable items (i.e., documents) are considered to be independent of each other. This means that the system only needs to visualize the ordering imposed by the ranking (or very similar documents can be grouped in a cluster). When showing highly relevant statute components in the answer interface the intrastatute (e.g., structural links, links to definitions) and interstatute relationships (e.g., references to other statutes) might be visualized. Individual components might be described by headings or important concepts found in the texts.

Further work with regard to the proposed retrieval models can be done by integrating structured query conditions in the ranking computations. We also plan to automatically extract and tag certain information in the statutes and to incorporate this micro-level information in the models. Another track of research is using the structured information in models for automatic categorization of legislative texts with ontological concepts.

References

[1] Biagioli, C., Francesconi, E., Spinosa, P. & Taddei, M. (2003). NIREditor, A XML specific environment for legislative drafting (Norme in Rete project). In *Proceedings of the JURIX 2003 International Workshop on the Development of Standards for Describing Legal Documents.* http://www.lri.jur.uva.nl/standards2003/

[2] Bing, J. (1984). *Handbook of Legal Information Retrieval* . Amsterdam: North Holland.

[3] Blanken, H.M, Grabs, T., Schek, H.-J., Schenkel, R. & Weikum, G. (Eds.) (2003). *Intelligent Search on XML Data, Applications, Languages, Models, Implementations and Benchmarks* (*Lecture Notes in Computer Science, 2818*). New York: Springer.

[4] Boer, A., Hoekstra, R. & Winkels, R. (2002). MetaLex: Legislation in XML. In T. Bench-Capon, A. Daskalopulu & R. Winkels (Eds), *Proceedings of the Fifteenth Annual International Conference on Legal Knowledge and Information Systems* (*Frontiers in Artificial Intelligence*) (pp. 73-82). Amsterdam: IOS.

[5] Boer, A., van Engers, T. & Winkels, R. (2003). Using ontologies for comparing and harmonizing legislation. In *Proceedings of the 9 th International Conference on Artificial Intelligence and Law* (pp. 60-69). New York: ACM.

[6] Bolioli, A., Dini, L., Mercatali, P. & Romano, F. (2002). For the automated mark-up of Italian legislative texts in XML. In T. Bench-Capon, A. Daskalopulu & R. Winkels (Eds), *Proceedings of the Fifteenth Annual International Conference on Legal Knowledge and Information Systems* (*Frontiers in Artificial Intelligence*) (pp. 21-30). Amsterdam: IOS.

[7] Croft, W.B & Lafferty, J. (2003). *Language Modeling for Information Retrieval.* Dordrecht: Kluwer Academic Publishers.

[8] Daniels, J.J. & Rissland (1997). Finding legally relevant passages in case opinions. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law* (pp. 39-46). New York: ACM.

[9] Fuhr, N. Groβjohann & S. Kriewel (2003). A query language and user interface for XML information retrieval. In H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel & G. Weikum (Eds.), *Intelligent SEARCH on XML Data* (pp. 59-75). Berlin: Springer.

[10] Hiemstra, D. (2003). Statistical language models for intelligent XML retrieval. In H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel & G. Weikum (Eds.), *Intelligent SEARCH on XML Data* (pp. 107-118). Berlin: Springer.

[11] Liu, X. & Croft, W.B (2004). Cluster-based retrieval using language models. In K. Järvelin, J. Allen, P. Bruza & M. Sanderson (Eds.), *Proceedings of the Twenty-seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 186-193). ACM: New York.

[12] Moens, M.-F. & Angheluta, R. (2003). Concept extraction from legal cases: The use of a statistic of coincidence. In *Proceedings of the Eight International Conference on Information Retrieval* (pp. 142-146). ACM: New York.

[13] Ramirez, G. & de Vries A.P. (2004). Combining indexing schemes to accelarate querying XML on content and structure. In *Proceedings of the First Twente Data Management Workshop On XML Databases & Information Retrieval* (pp. 44-51). University of Twente.