A Flexible Model for Retrieval of SGML Documents

Sung Hyon Myaeng & Dong-Hyun Jang
Dept. Computer Science
Chungnam National University
TaeJon, Korea
irsun.chungnam.ac.kr/~shmyaeng

Mun-Seok Kim & Zong-Cheol Zhoo
Information Retrieval Lab
Systems Engineering Research Institute
Taejon, Korea
wwwkle.seri.re.kr/~zczhoo

Abstract In traditional information retrieval (IR) systems, a document as a whole is the target for a query. With increasing interests in structured documents like SGML documents, there is a growing need to build an IR system that can retrieve parts of documents, which satisfy not only content-based but also structure-based requirements. In this paper, we describe an inference-net-based approach to this problem. The model is capable of retrieving elements at any level in a principled way, satisfying certain containment constraints in a query. Moreover, while the model is general enough to reproduce the ranking strategy adopted by conventional document retrieval systems by making use of document and collection level statistics such as TF and IDF, its flexibility allows for incorporation of a variety of pragmatic and semantic information associated with document structures. We implemented the model and ran a series of experiments to show that, in addition to the added functionality, the use of the structural information embedded in SGML documents can improve the effectiveness of document retrieval, compared to the case where no such information is used. We also show that giving a pragmatic preference to a certain element type of the SGML documents can enhance retrieval effectiveness.

1 Introduction

In traditional information retrieval (IR) systems, a document as a whole is the target for a query. Since documents are the basic unit for retrieval regardless of their lengths and their internal structures, users are limited to a single access point and are not allowed to take advantage of the structural information (e.g. titles, chapters, sections, paragraphs) in expressing their information needs. Structural information may have a meaning only when users browse through individual documents. Even when structural information is used in conjunction with retrieval results to help users narrow down to smaller units of text ([1]), the granularity for statistical information is usually at the document or collection level. There are only a few exceptions: one is the recent research on passage retrieval, and the other is the effort to retrieve text units smaller than a document from structured documents ([2], [3]). In passage retrieval, a document is decomposed into a set passages, and its retrieval is based on the similarity of each passage to the query (see [4] for comparisons among different approaches).

This paper introduces a new approach to retrieval of structured documents, especially SGML (Structured Generalized Markup Language) documents[5]. With SGML, the logical structure of a document can be expressed by defining a coherent unit of text as an *element* that can be embedded in another element or include other smaller elements. In addition, SGML documents can contain the relationship between units of a document using a link and the *attributes* of individual elements. While our research scope covers all the three aspects of SGML documents, we focus on the first aspect in this paper. More specifically we are interested in:

- how a retrieval system can provide users with a variety of access points to documents in response to queries that express both the content and the structure requirements, and
- how the structural (sometimes discourse-level) information embedded in documents can be exploited to enhance effectiveness of document-level retrieval.

It should be noted that while the latter is related to the motivation of the recent research on passage retrieval, the former addresses the issue of providing new functionality to IR systems.

This paper introduces a new IR model that is derived from the inference net model developed for document-level retrieval [6,7]. The main thrust of our model is to represent all the SGML elements of various granularities in a network so that their structural and semantic interdependency can be expressed. This model allows the users to specify structural constraints in their queries. The basic retrieval scheme is to calculate the degree to which an element at any level supports the query by considering what elements are contained in it. This scheme makes it possible to systematically calculate the expected relevance of an element at any level, taking into account its relationship with other elements in the network.

Our model is distinguished from the "proximal nodes" model [8] that focuses on the set-oriented query language capable of expressing both content and structural requirements. Not only can our model both represent structured documents and process various query types, but also elements at any level can be in the model. Recent work on retrieval of structured document using a logic-based model [2] is also different from ours. The emphasis is on the logical framework within which various document objects (i.e. elements) are represented. The framework provides a specific method of calculating the relevance of composite objects with limited expressiveness of queries. Besides, there is no report on its implementation. In contrast, we have implemented our model that is flexible enough to accommodate a variety of queries, belief calculation methods, and some pragmatic aspects of element types and queries.

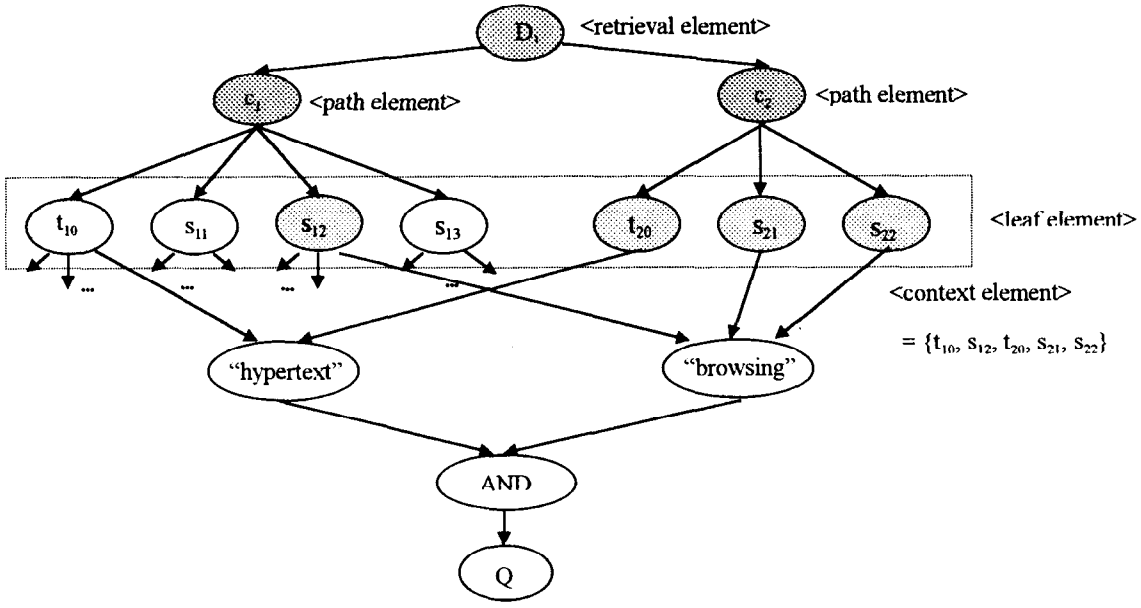

Figure 1. An inference net for a structured document and a query

There are several issues to be addressed in order to implement the model. First of all, since the original Bayesian inference net model requires expensive calculations of belief values, it is important to employ a more efficient calculation method that approximates the original. In fact, we take a stance that we can use a variety of belief calculation methods under the same representational framework as long as they do not violate basic probabilistic principles. We elaborate on this in the next section together with the description of the model.

Other implementation details are explained in Section 3, with an emphasis on the index structure and the belief calculation method that is applied at the time of indexing. As a way to make this model practical while complying with the inference net framework as faithfully as possible, we attempted to pre-calculate probability values at the time indices are created so that we can avoid calculating them at retrieval time.

In Section 4, we report on our experiments whose purpose was to demonstrate that the implemented model is capable of handling a variety of queries by exploiting additional information extracted from documents. In addition, we show how the structural information and pragmatic biases given to the text structure can enhance the quality of document retrieval.

2 A model for structured documents

From the outset of the research, we set up the following criteria with which we designed our model:

- provision of new retrieval functions for satisfying both content and structural information needs,
- improvement of retrieval effectiveness for *document* retrieval by using the structural information,
- flexibility in incorporating various interpretation and modulation of structural and semantic information embedded in structured documents, and
- possibility of developing a practical system with the model.

This section describes how these criteria can be addressed in our model. Other capabilities such as utilizing links and attributes and answering pure structural queries are left out as future works.

2.1 The Inference net model for IR

An inference net consists of nodes and directional links, representing random variables and relationships between the two linked nodes, respectively. It is often assumed that random variables take on values from {true, false}, allowing for a natural interpretation of a node as a proposition. A link from a parent to a child node represents a causal relationship, and a conditional probability is assigned to express the strength of the relationship. All dependencies are directly represented in the network, and no link between two nodes indicates that the nodes exist independently of each other and have no direct relationship between them. As long as there is a path between nodes, the belief that a node causes the other can be calculated by employing a mechanism that combines probabilities in a principled way.

When this framework is used for document retrieval as in [6], a node is created at least for each of document representations, index terms, query terms, and queries. The strength value on the link between a document and an index term, for example, can be interpreted as the degree of belief that the document supports for the term or vice versa, depending on which way the probabilities are propagated. Given all the probability values on the links as well as on the root nodes representing documents, the relevance of a document is determined by computing the degree to which a query is supported by a document. The readers are referred to [6] for details.

2.2 Representation of structured documents

In terms of topology, our model adopts the basic scheme of connecting two kinds of sub-nets, one for a set of documents and the other for a query. A unique aspect of our approach lies in its representation of structure information in documents. As in Fig. 1 that depicts a representation of a document instance and a query connected to it, a node and a link in the document

net represents an element and a hierarchical or containment relationship between two elements, respectively. This reflects the logical structure of SGML documents, expressing the fact that a child element (e.g. a section) is nested within a parent element (e.g. a chapter).

In the document sub-net in Fig. 1, the document consists of two chapters (c_1 and c_2), and the first chapter contains a title (t_{10}) and three sections (s_{11}, s_{12}, s_{13}). Although there are only three levels shown in this instance, the number of layers can vary depending on the document type. It should be obvious that only leaf nodes (titles and sections in this case) contain actual text, and that each leaf node has a number of term nodes that are omitted here.

The query sub-net connected to the document sub-net represents the following query containing both content and structural restrictions:

> "FIND a document that INCLUDES a chapter whose title CONTAINS the term *hypertext* AND whose section CONTAINS the term *browsing*."

This is an example of a *mixed query* that consists of the *structure-based* part, signified by the INCLUDES clause, and the *content-based* part, signified by the CONTAINS clause. The structural requirements are shown in the network by the links connecting the terms and the leaf nodes and those among intermediate nodes. For example, if there were no connection from "hypertext" node to t_{10} under c_1, the chapter would not appear in this figure. In other words, the figure represents those parts of the document, which are relevant to the query.

In order to show how a variety of query types are handled, we make a distinction among document nodes. Nodes participating in belief calculations can be labeled as a context, retrieval, or path element. A *context element* is defined to be the unit of text where a term is sought after. While the context elements in the example are the title and section elements that are also leaf elements, a query may specify a non-leaf element as a context element as in "Find a document where a chapter CONTAINS the term *hypertext*."

A *retrieval element* is the type of an element the user wants to retrieve and can be a leaf or non-leaf element. In traditional IR, documents are the retrieval element by default even when passage retrieval is performed. In the example query, the retrieval element specified by the FIND clause also happens to be a document. Users, however, may want to find sections or titles satisfying a certain requirement. It should be noted that a node can be both a context and retrieval element at the same time as in the case of traditional IR. Between a context element and a retrieval element, there may be one or more *path elements* that are used to compute the final probability value between a retrieval element and a leaf element. Path elements are explicit or implicit depending on whether they are specified in a query. In the example query, the chapter node is an explicit path element because it is specified in the INCLUDES clause.

2.3 Element retrieval with sub-net instantiation

Given the representation of a structured document, there should be a mechanism by which the belief value of a retrieval element (more precisely, the degree of belief that a retrieval element supports a query) is computed in a principled way. Conceptually, a query consisting of terms and operators (e.g. Boolean or vector) *instantiates* a part of the net composed of the nodes and links participating in the computation. The overall process consists of four steps:

1. mark all the candidate retrieval elements and the leaf nodes containing at least one of the query terms as well as the path elements,
2. compute the degree of belief that each retrieval element supports the leaf node via path elements,
3. compute the degree of belief that a query term node is supported by a set of leaf nodes connected to itself, and
4. compute the degree of belief that the query as a whole is supported by each of the retrieval elements by combining the evidence from all of the query term nodes based on the query operator.

It should be noted that after the first step, the computation process is described for a single retrieval element. When the computations are done for all the candidate retrieval elements, their rank order must be determined.

We currently assume there is no query that is purely structure-based without any content-based requirements. Therefore the process always starts with a leaf node, which may or may not be a context element, and works its way up to a retrieval element. This assumption of bottom-up calculation is motivated by an implementational need described in the next section.

2.4 Bayesian computation of belief

Given the instantiation process, we still need to specify how to compute the degree of belief at each of the steps 2 through 4. The degree of belief B for a node is expressed in principle by a conditional probability. Let C be a child node and F a set of parent nodes linked to the child node. Then,

$$B(C) = \sum_F P(C|F)\, X\, P(F) \qquad (1)$$

where the sum is over the power set ($2^{|F|}$) of the parent elements that are random variables taking values on $\{0,1\}$. The probability function $P(C|F)$, expressed as a link matrix in [6], leads to various specifications for computing $B(C)$ in general and for ranking strategies in particular. The summation is weighted by the probability $P(F)$ that is either a priori probability associated with the node or a degree of belief calculated using the formula (1) recursively.

For the sake of simplicity, we use the net in Fig. 2 in explaining some variations of calculations. It shows a net with two section elements as the parent nodes of a term node T and as child nodes of the element C.

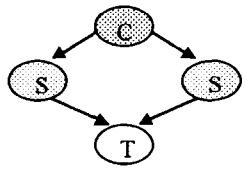

Figure 2. A simple network

The degree of belief for T given the network topology can be computed as follows:

$$\begin{aligned} B(T|C) = {} & P(T|\sim S_1, \sim S_2) * B(\sim S_1) * B(\sim S_2) \\ & + P(T \mid \sim S_1, S_2) * B(\sim S_1) * B(S_2) \\ & + P(T \mid S_1, \sim S_2) * B(S_1) * B(\sim S_2) \\ & + P(T \mid S_1, S_2) * B(S_1) * B(S_2) \end{aligned} \qquad (2)$$

The degree of belief for the section elements can be computed in a similar way:

$$B(S_i) = P(S_i \mid C) * P(C) + P(S_i \mid \sim C) * P(\sim C) \quad (3)$$

$$B(\sim Si) = P(\sim S_i \mid C) * P(C) + P(\sim S_i \mid \sim C) * P(\sim C)$$
$$= (1 - P(S_i \mid C)) * P(C) + (1 - P(S_i \mid \sim C)) * P(\sim C) \quad (4)$$

This computation yields an exponential time complexity with respect to the number of parent nodes, because it allows for an arbitrary link matrix or an arbitrary set of coefficients. It is only natural to consider different ways to calculate the degree of belief for an arbitrary node, which hopefully approximate the above formula.

2.5 A practical calculation method

In devising a method for belief calculations in our model, we attempted to enforce three criteria: computational tractability, consistency with the case of document retrieval (as opposed to element retrieval), and flexibility in exploiting semantics embedded in document structures and pragmatics users may want to specify in their queries. The first criterion is essential because strict adherence to the Bayesian method makes our system impractical. The second one ensures that if no structural information is used in addition to the usual TF, IDF, and length information, the bottom-up evaluation of a document similarity as described in Section 2.3 produce the same result as that of ignoring the document structure. In a sense, it makes the simplified calculation method sound. Finally, by allowing for the flexibility added to the basic framework, our model has a potential to take into account a variety of discourse level semantics and pragmatics for or against certain SGML elements of a document type.

The first criterion can be met by adopting a simplified formula that does not sacrifice the fundamental characteristics of the inference net. We opted for the following:

$$B(T|C) = P(T|S_1) * P(S_1) + P(T|S_2) * P(S_2)$$
$$= P(T|S_1) * P(S_1|C) * P(C) + P(T|S_2) * P(S_2|C) * P(C) \quad (5)$$

which considers only positive events. This is somewhat in line with Dempster-Shafer theory [9] where interactions among propositions (hypotheses) are not handled by listing all combinations of conditional probabilities, but by manipulating sets of propositions. The difference, though, is that we only deal with singleton sets of propositions (i.e. S_i's in (5)). The forumula can be easily generalized for cases with more than two parents.

The second criterion ensures that the similarity of a document to a given query be the same regardless of whether the document is considered as a structured document or as a flat text without any structural elements. Since one of our goals is to enhance retrieval effectiveness by exploiting the semantics implicit in structured documents, this restriction makes sense only when no additional information other than usual statistical parameters such as TF, IDF, and lengths of elements is used in evaluations. In other words, we want to make sure that identical terms in a document have the equal contributions toward the document similarity regardless of their positions, as long as no additional information about individual elements participates in belief calculations.

Fig. 3 depicts the situation where two identical terms occur in two different elements. Since the distance between S_{n1} and D is greater than that between S_{12} *and* D, there are more multiplication operations involved with S_{n1} than with S_2 in computing $B(T|D)$ as in:

$$B(T|D) = P(T|S_{n,1}) * P(S_{n,1}|S_{n-1,1}) * \ldots * P(S_{2,1}|S_{1,1}) * P(S_{1,1}|D)$$
$$* P(D) + P(T|S_{1,2}) * P(S_{1,2}|S_0) * P(D) \quad (6)$$

where $S_{i,j}$ is for j-th element at the i-th level. We contend that in this situation, the influence from the two terms occurring in the two elements must be the same, provided that no information other than the usual statistical parameters and the topology comes into play.

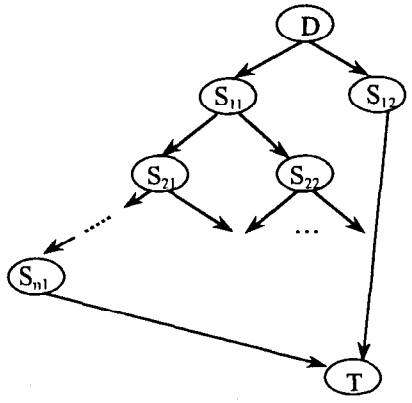

Figure 3. A term occurring at different levels

A natural way of enforcing the restriction is to make conditional probabilities proportional to the length of the child element and ensure the sum of the conditional probabilities on the links branching out of a parent node to be equal to 1. In other words, we must assign probability values in such a way that:

$$\sum_j P(S_{i,j}|S_{i-1,k}) = 1 \quad (7)$$

where $|j|$ is the number of children for $S_{i-1,k}$ for any k, and

$$P(S_{i,j}|S_{i-1,k}) \propto |S_{i,j}| \quad (8)$$

where $|S_{i,j}|$ for a leaf node is the number of index terms occurring in the element. If $S_{i,j}$ is a non-leaf node, $|S_{i,j}|$ is the sum of the sizes of its children.

In Fig. 3, for instance, $P(S_{1,1}|D)$ would be significantly larger than $P(S_{1,2}|D)$ if the sizes of the text associated with leaf elements are about the same. While the distance (i.e. the number conditional probabilities to be multiplied) to the root node makes the influence of T in $S_{1,1}$ weak, the same term T in $S_{1,2}$ has a small impact on D as well, because of the small value of the conditional probability $P(S_{1,2}|D)$.

2.6 Belief calculation with biases

The restrictions described so far provide our model with computational tractability and consistency of the belief calculation mechanism. Although it serves the purpose of calculating the similarity of any element at any level (hence retrieval of elements), it would have no impact on improving effectiveness of document retrieval. In order to take advantage of the effect of passage retrieval where the similarities of passages are used to retrieve documents, we should be able to discriminate among different elements depending on their contents and types. [10] has demonstrated by experiments that certain elements are more useful in retrieving structured

documents than others.

The content and type of elements can be utilized by reflecting them on the process of estimating conditional probabilities with the formula (5), which starts from the leaf element and all the way up to the retrieval element. A conditional probability in this context can be interpreted as the representativeness of the child node for the parent node or as the degree to which the parent node supports the child element. The type of an element can serve as a good indicator for the representativeness of the element. For example, we can assume that a title element is in general more central to the topic of the document (hence more representative) than a figure caption element and reflect it on probability calculations.

While there can be a number of different ways of estimating the conditional probabilities based on the contents of participating elements, which require further research, we consider two intuitive methods. One is to use the well-known vector similarity measure between the parent element and each of its children elements. Assuming that element vectors are constructed with the usual TF and IDF values, a child element with a large number of unique terms whose IDF values are high would have a high value for representativeness. That means, a term in an element with important terms makes a stronger contribution to the similarity of the retrieval element than a term in an element with many weak terms.

Given a set of children elements S_i and their parent C, represented as $\boldsymbol{S_i} = \langle s_{i1}, s_{i2}, ..., s_{in} \rangle$ and $\boldsymbol{C} = \langle c_1, c_2, ..., c_n \rangle$, respectively, where n is the number of index terms and s_{ij} and c_i are calculated with IDF and TF, the probability can be estimated as follows:

$$P(S_i|C) = \lambda_i * (\boldsymbol{S_i} \bullet \boldsymbol{C}) \tag{9}$$

where λ_i is a weight of the particular element type, representing its overall importance relative to other types of elements sharing the same parent. This computation incorporates both the content and the type of elements. The value of λ_i can be fixed a priori or changed with the history of system operations. As in the equation (7), the parameter should be set in such a way that the sum of all the probabilities conditioned by the same parent is 1:

$$\sum \lambda_i * (\boldsymbol{Si} \bullet \boldsymbol{C}) = 1 \tag{10}$$

The other method is based on our notion of coherency of an element, which is assumed to be the degree to which the content of the element is similar to those of sibling elements. If the content of an element is a complete digression from the rest of the elements under the same parent, it is determined to have low coherency relative to its siblings and hence low representativeness. On the other hand, if the average similarity between the element and the rest of the elements under the same parent is high, it is deemed to have high coherency. More specifically, coherency of S_i with respect to its parent C can be computed as follows:

$$Coherency(S_i) = \sum_j \boldsymbol{S_i} \bullet \boldsymbol{S_j} \quad where\ j \neq i \tag{11}$$

It should be noted that $|j|$ is the number of children for S_i's parent.

3 Implementation Issues

There are three main issues we explore in this section. The first is how the probabilities are estimated in indexing, and the second is how the indexing results are organized and stored for retrieval tasks. Finally we describe how elements at any level can be efficiently retrieved using the index structure. In the current implementation, it is assumed that content-based retrieval can be done before structural constrains are satisfied in processing queries that contain both content and structure specifications..

3.1 Indexing

The probabilities to be estimated directly from documents are basically $P(T|S_i)$, $P(S_i|C)$, *and* $P(C)$ *as* in formula (5). As indicated in the previous section, we make use of more or less conventional statistical techniques using TF and IDF values and the vector similarity measure in estimating various probabilities. The probability $P(T|\ S_i)$ of observing a term T given an element S_i (or the degree of belief that S_i supports T) can be estimated with:

$$P(T|S_i) \cong IDF_T * TF_{i,Si} \tag{12}$$

Here IDF_T and $TF_{i,Si}$ are normalized as in [TC91]. The probability $P(S_i|C)$ of observing S_i given C (or the degree of belief that C supports S_i) can be calculated as a similarity between the two vectors as follows:

$$P(S_i|C) \cong \boldsymbol{S_i} \bullet \boldsymbol{C} \tag{13}$$

$P(C)$ is the probability of observing C assuming that C is the root node (i.e., document) in the inference net. One feasible way of estimating this is to use the history of the system. The more often a document (element) has been retrieved and judged to be relevant, the higher the value. In our current implementation, however, it is set to 1 as is the case in other work [6].

It should be noted that all the probability values can be calculated off-line, i.e., at the time of indexing. The only calculations to be done at the retrieval time are the multiplication operations of probability values from the leaf nodes all the way up to the retrieval node. As explained in the next sub-section, all the necessary information for the multiplication operations can be found in a single inverted index.

3.2 Index structure

Since an IR system that supports structured documents must exploit additional information embedded in the structure, it is natural that additional space is required for an index. In order to minimize the space as well as time complexity, however, we used the following design decisions.

- Since processing a structure-based query is likely to be expensive, we process the content-based retrieval first to result in a smaller search space and then filter out those that are not satisfied with attribute and/or structure constraints.
- In order to process pure structure-based and attribute-based queries, there should be separate index structures.
- Considering a huge number of composite elements in which other elements are nested, indexing should be done only with the leaf elements so that the storage overhead is minimized.

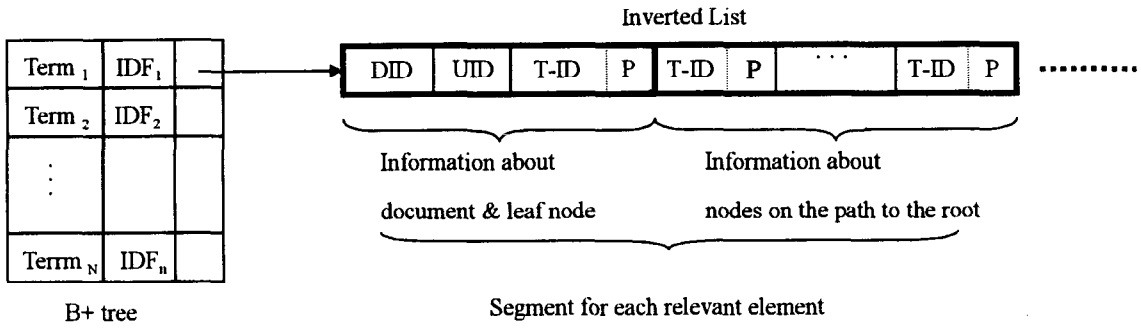

Figure 4. Structure of Inverted Lists

- Since conditional probabilities between parents and children are used several times to calculate the similarity value of an arbitrary element to a given query, these probabilities should be accessible efficiently.

Considering the design decisions and our retrieval model where the belief value is calculated in a bottom-up fashion from the leaf node toward the retrieval element, we designed and implemented an inverted index structure shown in Fig. 4. The main thrust of this structure is that an inverted index list consists of not only the list of leaf elements containing the path, but also the path information associated with each of the leaf. The path information specifies what elements exist between the leaf element and the root element, together with a set of the conditional probabilities between each pair of a parent and a child along the path.

As shown in Fig. 4, an inverted list consists of one or more *segments*, each of which corresponds to a complete path starting from a leaf node to the root node. A segment is divided into two parts. The first part contains a document ID, a unique element ID, an element type (TID), and a probability value. The pair of a DID (document ID) and a UID uniquely identifies an element in the entire collection even if there are more than one document types, hence more than one DID's. The second part contains a list of <element type, probability> pairs, each of which specifies the type of an element along the path and the associated probability that the parent node supports the child node.

It should be noted that the TID's are stored for both leaf nodes and the path nodes in order to deal the specifications as to the context, explicit path, and retrieval elements in queries. That is, not all the leaf nodes containing a query term can lead to the final set of retrieval elements. The types of elements in the path are checked against the context and/or explicit path element type in the query so that the leaf nodes not satisfying the structural constraints can be eliminated at an early stage. On the other hand, the element ID for each pair in the second part is not stored because it can be calculated dynamically by using the leaf element ID as in [11]. The basic idea is that when a unique ID is assigned to each element (node), a complete tree with a branching factor k is assumed. Since unique IDs are assigned to virtual nodes as well as the actual nodes, it is not hard to compute the ID for the parent of a node.

At first glance, the inverted index structure seems to take up an enormous amount of storage because the number elements in a document can be large. When there are t index terms and e elements per document on average, however, the total number of element ID's in the entire index (i.e. the sum of the number of elements in all the inverted lists) is not as many as e times the number of documents. This is because each element contains a significantly less number of terms than a document does, and hence an element ID does not occur in as many inverted index as a document. In other words, the term-element matrix is sparser than the term-document sparse although the number of columns in the term-element matrix is e times the number of columns in the term-document matrix.

3.3 Retrieval process

As described in 2.3., the actual retrieval process is performed in a bottom-up fashion, mainly due to the use of an inverted index that drives the retrieval process to be triggered by a term in a query as in other conventional systems. The other reason for the bottom-up query evaluation is because of the way documents are represented in an inference net: no elements other than leaf elements contain actual text, and the retrieval process must start with where the text containing a query term is.

Our design of the inverted index structure as described in the previous sub-section makes it possible to minimize the time required for the bottom-up retrieval process. Since a large portion of belief calculations is done off-line, retrieval decisions can be made fairly efficiently. In addition, an inverted list contains all the necessary information to determine whether traversing the path toward the root node will lead to a success. By scanning the inverted list associated with a query term, the retrieval module can easily find out whether there is a segment with the type of element specified as the retrieval element in the query. If a segment does not have such element in the path, it is not a candidate for retrieval although the leaf element contains the query term. If a segment contains an element of the specified type, however, the belief calculation begins.

Our system currently implements the p-norm model [12] for query processing so that it can process both the Boolean and vector space query model. Once the evidence that a retrieval element supports a query term is calculated, it can be combined with the evidence calculated for another query term. The result can be combined with a still new query term, and so on, until all of the query terms are taken into account.

4 Experiments

Since there was no suitable test collection available, which consists of SGML documents, a set of queries with structural constraints, and associated relevance judgments, it was not possible to test how effectively a variety of element-level

queries (as opposed to document-level queries) can be handled. As a result, the goal in our experiments is limited to demonstrate the following three aspects.

The first was to see if the implemented system could process arbitrary queries that specify not only a retrieval element at any level in SGML documents but also some containment constraints (i.e., explicit path and context elements) in themselves. The purpose was to validate the model in a qualitative sense and see if the retrieval results are reasonable. Since it is a test for new functionality supported by the proposed model and the system, we won't discuss it further in this section. The second was to show that our use of the structural information embedded in the SGML documents can actually improve the effectiveness of document retrieval, in comparison with the case where no such information is used. This is an indirect way of validating the model for its passage retrieval function. Finally, we were interested in finding out whether the idea of giving a bias (i.e. semantic and pragmatic preference) to an element type actually improves retrieval effectiveness. This is also a way of indirectly testing the hypothesis that certain element types have their inherent values in satisfying queries.

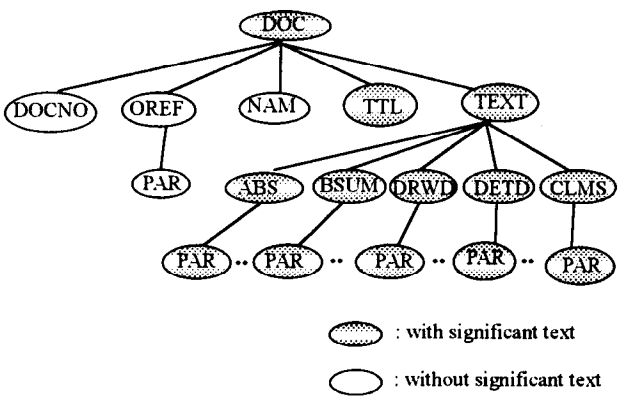

Figure 5. A typical structure of a patent document

We used the patent data and a subset of topics 51 through 150 in the TREC collection. The main reason for our choice was because the patent documents are fairly long and, more importantly, contain nested element type tags. Most patent document trees have at least four levels. Fig. 5 shows the structure of a typical patent document. We used 32 queries because other queries in the collection have no relevant documents in the patent collection. By selecting 234 relevant documents for the queries and 766 irrelevant ones randomly selected from the rest of the original patent collection, we built a collection of 1,000 documents (about 40MB).

The baseline was the flat document case where all the tags were removed. We employed the typical TF*IDF weighting scheme, IDF weights on query terms, and the vector cosign similarity measure for ranking. In order to test the value of using structural information embedded in the documents, the baseline was compared with the "pure" model case where the belief calculation model described in sub-section 2.5 was employed without any weighting biases against individual element types. That is, the conditional probabilities were computed using (12) and (13) in sub-section 3.1 with the constraints imposed by (7) in sub-section 2.5.

Table 1 shows the comparison between a baseline that ignores all the structural information (i.e. flat documents) and the test case where evidence from leaf nodes are combined in a bottom-up fashion. The overall results indirectly demonstrates the feasibility of using our model for structured documents and directly shows that the use of structures in SGML documents indeed improves the retrieval effectiveness, especially precision, confirming previous results that the use of passages is generally helpful. The major difference is that instead of using artifacts such as the use of best scores, we used the notion of the representativeness of elements and our belief calculation method in the inference net representation of SGML documents.

Recall Level	Baseline (Flat Documents)	Use of Structured Documents	% Charge
0.00	0.2826	0.4027	42
0.10	0.2603	0.3864	48
0.20	0.2383	0.3295	38
0.30	0.2204	0.2710	23
0.40	0.1897	0.2304	21
0.50	0.1766	0.2156	22
0.60	0.1345	0.1433	6
0.70	0.1291	0.1224	-5
0.80	0.1117	0.0956	-14
0.90	0.0808	0.0872	8
1.00	0.0550	0.0795	45
Average	0.1563	0.2030	30

Table 1. Comparison between a case with flat documents and a case with structured documents

For the third aspect of our experimental goal, we first investigated on whether different element types in the patent document make different contributions to retrieval of whole documents. Each of the non-trivial element types (i.e. those containing some content-bearing terms) was used as the only indexable region and tested for their effectiveness in document retrieval. The experimental results are shown in Table 2. It should be clear that when the <TEXT> element type and the <BSUM> element type, a child of <TEXT>, were used alone, respectively, the retrieval results outperform the case where all the element types were used.

Element Type Name	11-point average precision
All element types used	0.2030 (0)
<TTL>	0.1311 (-35.4)
<TEXT>	0.2123 (+4.6)
<BSUM>	0.2419 (+19.2)
<ABST>	0.1633 (-19.6)
<CLMS>	0.1407 (-31.7)
<DETD>	0.1512 (-25.5)
<DRWD>	0.0879 (-56.7)

Table 2. Retrieval results with individual element types

Having observed element types vary in their inherent values for retrieval, we tested the idea of assigning weights on element types as described in sub-section 2.6. Since the immediate children of the <TEXT> element are most interesting in their individual contributions to retrieval effectiveness as in Table 2, we ran the system with different biases given to <BSUM>, <ABST>, <CLMS>, <DETD>, and <DRWD>. Table 3 shows some prominent parts of the

experimental results. The first row is the pure model case without any biases at all, whereas the other rows shows the cases with different weights given to the element types. It is clear that the strategy of giving biases to individual element types is worth further investigation.

<BSUM>	<ABST>	<CLMS>	<DETD>	<DRWD>	11-pt. Avg. P
-	-	-	-	-	0.2030 (0 %)
0.6	0.1	0.1	0.1	0.1	0.2458 (+21 %)
0.7	0.1	0.1	0.1	-	0.2503 (+23 %)
0.8	0.1	-	0.1	-	0.2571 (+27 %)
0.9	0.1	-	-	-	0.2558 (+26 %)
0.9	-	-	0.1	-	0.2453 (+20 %)

Table 3. Results with biases given to element types

5 Conclusions

We have defined and implemented an inference-net-based model for retrieval of structured documents, which can handle queries based on both the content and the structure of SGML documents. The main thrust of the model lies in its capability of retrieving elements at any level in a principled way, satisfying certain containment constraints in a query. Moreover, while the model is general enough to reproduce the ranking strategy adopted by conventional document retrieval systems by making use of document and collection level statistics such as IDF and TF, its flexibility allows for incorporation of a variety of pragmatic and semantic information associated with document structures.

We implemented the model and described some of the details to show how the complex-looking model can be realized as a practical system. In our current implementation, a large portion of probability calculations are done off-line, and the results are stored an inverted index so that retrieval can be done efficiently. With the system, we conducted some experiments to demonstrate:

- that the model can be used to effectively process queries that specify not only a retrieval element at any level in SGML documents but also some containment constraints in themselves,
- that our use of the structural information embedded in the SGML documents can improve the effectiveness of document retrieval, compared to the case where no such information is used, and
- that careful assignment of biases (weights) on different element types actually improve the retrieval effectiveness because different element types have their own pragmatic values for retrieval.

There are a number of issues to be addressed in the near future. Most notably, we are currently in the process of extending the current model to handle more complicated queries including such constraints as attribute values, pure structures, and hypertext links. We also plan to run more extensive experiments to figure out better ways to estimate various probabilities and to assign weights to element types, exploiting semantic and pragmatic information obtainable from various sources. We also plan on comparing the usual passage retrieval against the case of using SGML tags for passage identification. Additional experiments must be done for the purpose of understanding what influences different belief calculation methods will have on retrieval effectiveness.

References

[1] Egan, D. E. et al. "Formative design-evaluation of SuperBook", *ACM Transactions on Information Systems*, 7 (1), January, 1989, pp 30-57.

[2] Lalmas, M., "Dempster-Shafer's theory evidence applied to structured documents: modeling uncertainty," *Proc. of ACM SIGIR'97*, Philadelphia, pp 110-118, 1997.

[3] Volz, M., Aberer, K., & Bohm, K., "Applying a flexible OODBMS-IRS-coupling to structured document handling," *Proc. of 12th International Conference on Data Engineering*, New Orleans, 1996.

[4] Kaszkiel, M & Zobel, Justin, "Passage retrieval revisited", *Proc. of the 20th ACM-SIGIR'97*, Philadelphia, 1997.

[5] Van Herwijnen, E., *Practical SGML*, 2nd ed., Kluwer Academic Publishers, Boston, 1994.

[6] Turtle, H. & Croft, B. W., "Evaluation of an inference network-based retrieval model," *ACM Transactions on Information Systems*, 9 (3), pp 187-222, 1991.

[7] Ribeiro, B. & Muntz, R., "A belief network model for IR," *Proc. of ACM SIGIR'96*, Zurich, pp 253-260, 1996.

[8] Navaro, G. & Baeza-Yates, R., "A language for queries on structure and contents of textual databases," *Proc. of ACM SIGIR'95*, Seattle, pp93-101, 1995.

[9] Shafer, A., *A Mathematical Theory of Evidence, Princeton University Press*, 1976.

[10] Wilkinson, R., "Effective retrieval of structured documents," *Proc. of ACM SIGIR'94*, pp 311-317, Dublin City, 1994.

[11] Lee, Y., Yoo, S., Yoo, K., & Berra, B., "Index structure for structured documents," *Proc. of Digital Libraries '96*, Bethesda, pp 91-99, 1996.

[12] Salton, G., Fox, E,, & Wu, H., "Extended Boolean information retrieval," *Communications of ACM*, 26 (12), pp1022-1036, 1983.