Accurate Retrieval of XML Document Fragments using EXTIRP

Antoine Doucet* Lili Aunimo Miro Lehtonen Renaud Petit
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN–00014 University of Helsinki
Finland
[Antoine.Doucet|Lili.Aunimo|Miro.Lehtonen|Renaud.Petit]@cs.Helsinki.FI

ABSTRACT

EXTIRP[1], a novel XML retrieval system, aims at finding elements with exact coverage by first dividing XML documents into a set of minimal XML fragments and then ranking and combining them into retrieved document fragments. With respect to a query, a similarity measure is computed for each fragment by combining the scores of a vector space model with term-based features and a text phrase model. The similarity measures are propagated bottom-up from the smallest units to article-sized ancestor elements. The system also includes query expansion, with which the score calculation can be iterated.

1. INTRODUCTION

In this paper, we focus on the problem of finding an answer with optimal coverage of the topic, given an unstructured query (CO topics in INEX). That is, we want to find a trade-off between responding to a query with a 15 page article and a fragment that is not sufficient when deprived of its context. The architecture of the interactive part of the system is presented in Figure 1. As input, the system takes a CO topic, and as output, it gives a ranked list of document fragments. In Figures 2 and 3, the non-interactive part of the system is described. This non-interactive part is run offline when the system is taken into use or when the document collection changes. Figure 2 shows how the document collection is transformed into inverted indices consisting of document fragments of different granularities. Figure 3 illustrates how an inverted text phrase index is created for each of the different granularities. MFS stands for Maximal Frequent Sequence (see Section 3.3.1 for definition).

Previously, every single element of the document collection has been indexed, e.g., see [6, 7], but modeling and computing a *Retrieval Status Value* (RSV) for each element causes a clear problem with efficiency. We limit the set of indexed elements to those that can be retrieved on their own, and define the minimal unit of retrieval, such that none of its parts is big enough to be of interest by itself. An RSV is computed for each minimal unit using words as features in the vector space model and multiword expressions. The

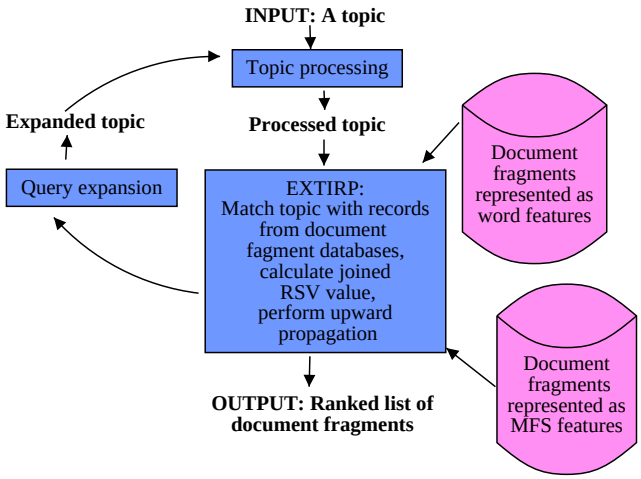

Figure 1: The system architecture of the interactive part of EXTIRP.

RSVs of the minimal units are finally propagated upwards to their ancestors. One or more query expansion steps can be iterated to form more extensive topic descriptions.

Section 2 describes the XML-related processing of the document collection. Our document and query models are presented in Section 3, followed by the corresponding techniques to evaluate similarities within these models in Section 4. We explain our query expansion technique in Section 5. The system description ends in Section 6, where we present the method used to propagate RSVs upwards. We finally describe our runs in Section 7 and conclude.

2. PREPARATORY PROCEDURES

Finding the most relevant text documents for each given topic is the basic problem to be solved in traditional IR. But, as the document collection is in XML format, we can identify two additional challenges that must be overcome before any traditional IR methods can be applied. First, the document collection consists of 125 XML documents which alone are too big to be retrieved on their own. Therefore, the collection is divided into smaller XML units which we shall call *XML fragments*. Second, the XML fragments contain all the XML markup that is present in the original XML format.

*This author is supported by the Academy of Finland (project 50959; DoReMi - Document Management, Information Retrieval and Text Mining)
[1]EXacT coverage IR based on static Passage clusters

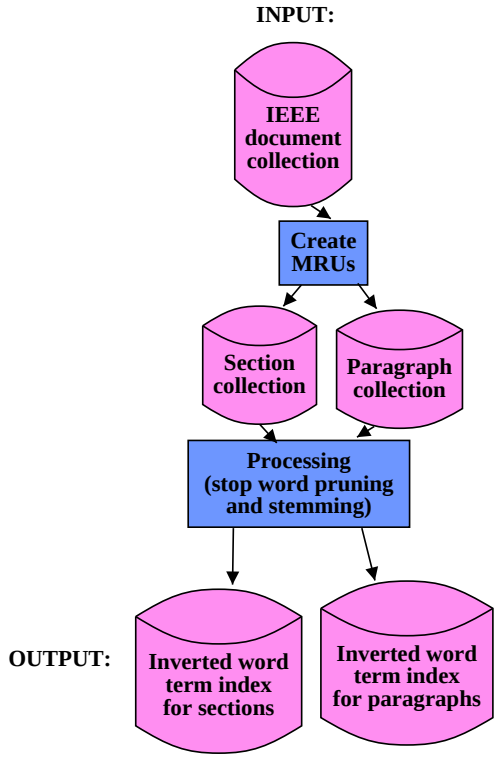

Figure 2: This module transforms the IEEE document collection into word term indices.

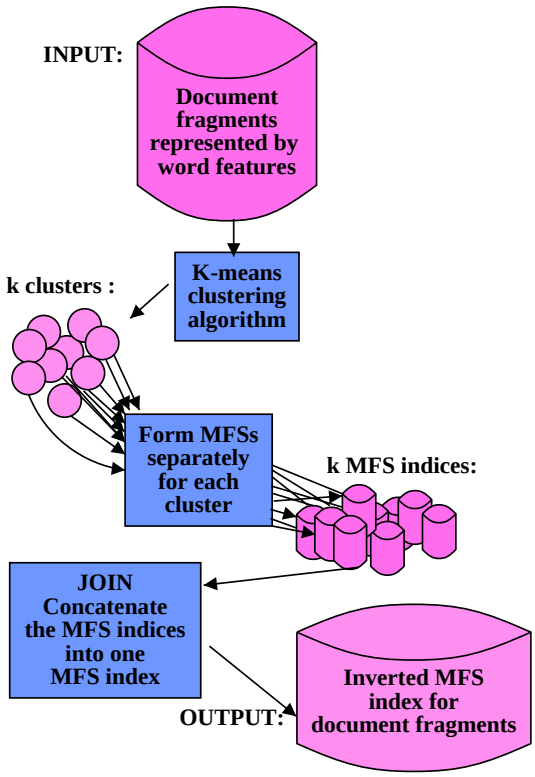

Figure 3: This module forms MFS indices. It is run separately for each of the different levels of granularitry.

Our goal is to convert the XML fragments into a text-only format where all XML markup has been removed without losing any of the information that is implicitly or explicitly coded in the XML structure of the original documents.

2.1 Division of the collection

The division of the collection was performed at two different levels of granularity called section-level and paragraph-level divisions. The levels for these two separate divisions were defined manually by looking into both the XML DTD and the XML documents. For example, the division into section-sized fragments concerned the following XML elements: `sec, fm, bm, dialog, vt`. In the document tree, all of these elements are close descendants of the `article` element, and none of them have text node children. In a similar fashion, the paragraph-sized elements taken into account in the paragraph-level division are `p, p1, p2, ip1, ip2, ip3, bq`. These elements have text node children, and also, most of the text content of the collection is covered by choosing these elements. A similar approach with a different set of element names was chosen by Ben-Aharon et al. [3].

By carefully defining the set of similar elements for each level, we intend to approximate an unsupervised division into fragments that is based on structural features only. Moreover, concentrating on structural similarity and discarding the information about element names will set us free from any particular document type or DTD. One might argue that contextual information is neglected by ignoring information specific to the document type. We believe, however, that identical content should be valued equally whether its parent element is called `sec` (section) or `bm` (back matter). Automating the division still remains part of our future work.

Intra-document links create connections between related XML elements. For example, footnotes are linked to the paragraphs that have a reference to the footnote element. Other examples include figure and table captions, biographical and bibliographical information, and other out-of-line content. Fragmentation of the collection separates linked elements unless both ends of the link belong to the same fragment. To avoid this, we have included some of the referred content that would increase the informational value of the fragment. Again, finding the intra-document links is possible without knowing the document type by a careful analysis of attribute values.

After the division, we have a collection of XML fragments. Each fragment is considered independent of the others, although information about the origin of the fragment is still included. The fragments can be combined later to make results with wider coverage, but dividing them further is hardly sensible as the size of a fragment is already supposed to be sufficiently small. In our system, these XML fragments constitute what are defined as *Minimal Retrieval Units* (MRU).

2.2 Structural conversion

The XML structure of an MRU is not ideal for linguistic processing. Although XML is a textual format and the tag names often are words, the semantics of the markup is different from that of the actual text content and thus

should be treated differently. Our goal of a text-only format is achieved by simply removing all markup; however, this would lead to the loss of all the information carried by the structure. To avoid this unnecessary information loss, we suggest that the structure be analysed and utilised to the greatest extent possible before being removed.

Unlike Ben-Aharon et al., we set a goal that the structural analysis must not be specific to any document type. As a consequence, no particular element type has a special way of being processed, and also, elements of the same type are processed differently under different structural circumstances. Only the structural properties of an element should determine the way it is handled.

The starting point of the analysis is the highest level of text nodes in the tree representation of the XML fragment. Any text node at a lower level is seen to stand out, and it is usually formatted differently in a printable version of the document. For example, all the text with added emphasis is marked with inline-level tags which often imply changes in the current typeface. Although not all inline-level elements denote a change in the typeface, we have found heuristics with which we can automatically determine whether added emphasis or other inline-level content is in question. After detecting the emphasised content, we can remove the tags and preserve the emphasis by giving the content more weight in the index than the unemphasised content, e.g. by repetition.

3. DOCUMENT AND QUERY MODELS

In our approach, we represent the MRUs by word features of the vector space model, and by multiword expressions accounting for the sequential aspect of text. An RSV is computed for each of those two representations. These values are later combined to form a single RSV per MRU, that will later be propagated to parent nodes as described in Section 6.

3.1 Preprocessing

The first step of the modeling phase is to cleanse the data. We do this by skipping a set of words that are considered least informative, the *stopwords*. We also discarded all words of small size (less than three characters).

In addition, we reduced each word to its stem using the Porter algorithm [10]. For example, the words "models", "modelling" and "modeled" were all stemmed to "model". This technique for reducing words to their stems allows further reduction of the number of term features.

This feature selection phase brings more computational comfort for the next steps since it greatly reduces the size of the document collection representation in the vector space model (the *dimension* of the vector space).

3.2 Modeling document fragments

The set of the remaining word stems W is used to represent the MRUs of the document collection within the *vector space model*. Each minimal retrieval fragment is represented by a $\|W\|$-dimensional vector filled in with a weight standing for the importance of each word w.r.t. that fragment. To calculate this weight, we used a normalized *tfidf* variation following the "tfc" term-weighting components as detailed by Salton et al. [13], that is:

$$tfidf_w = \frac{tf_w \cdot \log \frac{N}{n_w}}{\sqrt{\sum_{w_i \in W} \left(tf_{w_i} \cdot \log \frac{N}{n_{w_i}} \right)^2}},$$

where tf_w is the term frequency of the word w. N is the total number of MRUs in the document collection and n the number of MRUs in which w occurs.

3.3 Extracting Maximal Frequent Sequences

3.3.1 Definition and technique

Maximal Frequent Sequences (MFS) are sequences of words that are frequent in the document collection and, moreover, that are not contained in any other longer frequent sequence. Given a frequency threshold σ, a sequence is considered to be frequent if it appears in at least σ documents.

Ahonen-Myka presents an algorithm combining bottom-up and greedy methods in [1], that permits to extract maximal sequences without considering all their frequent subsequences. This is a necessity, since maximal frequent sequences may be rather long.

Nevertheless, when we tried to extract the maximal frequent sequences from the collection of MRUs obtained as described in Section 2, their number and the total number of word features in the collection did pose a clear computational problem and did not actually permit to obtain any result.

To bypass this complexity problem, we partitioned the collection of MRUs into several disjoint subcollections, small enough so that we could efficiently extract the set of maximal frequent sequences of each subcollection. Joining all the MFS sets, we obtained an *approximate* of the maximal frequent sequence set for the full collection. This process is shown in Figure 3.

We conjecture that more consistent subcollections permit to obtain a better approximation. This is due to the fact that MFSs are formed from similar text fragments. Followingly, we formed the subcollection by clustering similar documents together using the common k-means algorithm (see for example [17, 5]).

3.3.2 Main Strengths of the MFSs

The method efficiently extracts all the maximal frequent word sequences from the collection. From the definitions above, a sequence is said to be maximal if and only if no other frequent sequence contains that sequence.

Furthermore, a *gap* between words is allowed: in a sentence, the words do not have to appear continuously. A parameter g tells how many other words two words in a sequence can have between them. This parameter g usually gets values between 1 and 3.

For instance, if g = 2, a phrase "president Bush" will be found in both of the following text fragments:
`...President of the United States Bush...`
`...President George W. Bush...`
Note: Articles and prepositions were pruned away.

This allowance of gaps between words of a sequence is probably the strongest specificity of the method, compared to the other existing methods for extracting text phrase descriptors. This greatly increases the quality of the phrases, since the variety of natural language is taken into account.

Another strength of the technique is the ability to extract maximal frequent sequences of any length. This permits to obtain a very compact description of documents. For example, by restricting the length of phrases to 8, a maximal frequent sequence of 25 words would have to be represented by thousands of phrases of size 8, even though they would represent the same knowledge !

3.4 Modeling queries

To build our queries, we only considered words found in the <title> and <keywords> elements. For consistency, we applied the same preprocessing to them as to MRUs.

Vector space model. A novelty in INEX 2003 was the possibility to precede keywords with various operators. A keyword preceded by "-" meant that this word was not desired, whereas a keyword preceded by "+" indicated that this word was especially important. We attached different weights to keywords preceded by such operators:

- no prefix operator: the normal case, weight 1
- +: especially important, weight 1.5
- -: especially not desired, negative weight -1

In practice, things were not that simple, since the same word could occur within two phrases with contradictory operators (e.g., "language" in topic 111 occurs in -"programming language" and in +"human language"). In such rare cases, we ignored the word (weight: 0).

Keyphrases. All the phrases occurring in the <title> and <keywords> elements are stored in the (possibly empty) set of keyphrases representing the topic. For example, topic 117 (see Figure 4) will be represented by the 4 phrases: "Patricia Tries", "text search", "string search algorithm", "string pattern matching".

4. EVALUATING DOCUMENTS

Once document fragments and queries are represented within our two models, a way to estimate the relevance of a fragment w.r.t. a query remains to be found. As mentioned earlier, we compute separate RSVs for the word features vector space model and the MFS model. In a second step, we aggregate these two RSVs into one single relevance score for each document fragment w.r.t. a query.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="117" query_type="CO"
  ct_no="98">
<title>Patricia Tries </title>
<description>Find documents/elements that describe
Patricia tries and their use.</description>
<narrative>To be relevant, a document/element
must deal with the use of Patricia Tries for text
search. Description of the standard algorithm,
optimisied implementation and use in Information
retrieval applications are all relevant.
</narrative>
<keywords>Patricia tries, tries, text search,
string search algorithm, string pattern matching
</keywords>
</inex_topic>
```

Figure 4: Topic 117.

4.1 Word features RSV

The vector space model offers a very convenient framework for computing similarities between MRUs and queries. Indeed, there exists a number of techniques to compare two vectors. Eucclidean distance, Jaccard and cosine similarity being the most frequently used in IR. We have used cosine similarity because of its computational efficiency. By normalizing the vectors, $cosine(\overrightarrow{d_1}, \overrightarrow{d_2})$ indeed simplifies to vector product $(d_1 \cdot d_2)$.

4.2 MFS RSV

To compute a RSV using MFSs, the first step is to create an MFS index for the MRU collection. Once a set of MFSs has been extracted and each document fragment has been attached to its corresponding phrases, as detailed in Section 3.3, it only remains to define the procedure to match a phrase describing a MRU to a keyphrase and compute a corresponding RSV for each MRU.

Note that from here onwards, *keyphrase* denotes a phrase found in a query, and *maximal sequence* denotes a phrase extracted from a document fragment.

To compare keyphrases and MFSs, our approach consists of decomposing keyphrases of a query into pairs. Each of these pairs is bound to a score representing its *quantity of relevance*. Informally speaking, the quantity of relevance of a word pair tells how much it makes a document relevant to include an occurrence of this pair. This value depends on the specificity of the pair (expressed in terms of inverted document frequency) and modifiers, among which an *adjacency coefficient*, reducing the quantity of relevance given to a pair formed by two words that are not adjacent.

4.2.1 Definitions:

Let D be a collection of N document fragments and $A_1 \dots A_m$ a keyphrase of size m. Let A_i and A_j be 2 words of $A_1 \dots A_m$ occurring in this order, and $n_{A_iA_j}$ be the number of MRUs of the collection in which A_iA_j was found. We define the quantity of relevance of the pair A_iA_j as:

$$Q_{rel}(A_iA_j) = idf(A_iA_j) \cdot adj(A_iA_j),$$

where $idf(A_iA_j, D)$ represents the specificity of A_iA_j in the collection D:

$$idf(A_iA_j) = \log\left(\frac{N}{n_{A_iA_j}}\right),$$

and when decomposing the keyphrase $A_1 \ldots A_m$ into pairs, $adj(A_iA_j)$ is a score modifier to penalize word pairs A_iA_j formed from non adjacent words. d(A_i,A_j) indicates the number of words appearing between the two words A_i and A_j (d(A_i,A_j) = 0 means that A_i and A_j are adjacent):

$$adj(A_iA_j) = \begin{cases} 1, & & \text{if d}(A_i,A_j) = 0 \\ \alpha_1, & 0 \leq \alpha_1 \leq 1, & \text{if d}(A_i,A_j) = 1 \\ \alpha_2, & 0 \leq \alpha_2 \leq \alpha_1 & \text{if d}(A_i,A_j) = 2 \\ \ldots & & \\ \alpha_{m-2}, & 0 \leq \alpha_{m-2} \leq \alpha_{m-3}, & \text{if d}(A_i,A_j) = m-2 \end{cases}$$

Followingly, the larger the distance between the two words, the lowest quantity of relevance is attributed to the corresponding pair. In our runs, we will actually ignore distances higher than 1 (i.e., $(k > 1) \Rightarrow (\alpha_k = 0)$).

4.2.2 Example:

For example, ignoring distances above 1, a keyphrase ABCD is decomposed into 5 tuples (pair, adjacency coefficient):

(AB, 1), (BC, 1), (CD, 1), (AC, α_1), (BD, α_1)

Let us compare this keyphrase to the documents d_1, d_2, d_3, d_4 and d_5, described respectively by the frequent sequences AB, AC, AFB, ABC and ACB. The corresponding quantities of relevance brought by the keyphrase ABCD are shown in table 1.

Assuming equal idf values, we observe that the quantities of relevance form a meaningful order. The longest matches rank first, and matches of equal size are untied by adjacency. Moreover, non adjacent matches (AC and ABC) are not ignored as in many other phrases representations [9].

4.3 Aggregated RSV

In practice, some queries do not contain any keyphrase, and some documents do not contain any MFS. However, there can of course be correct answers to these queries, and those documents can be relevant answers to some queries. Also, all document fragments containing the same matching phrases get the same MFS RSV. Therefore, it is necessary to find a way to separate them. The word-based cosine similarity measure is very appropriate for that.

Another possible response would have been to further decompose the pairs into single words and form fragment vectors accordingly. However, this would not be satisfying, because the least frequent words are missed by the algorithm for MFS extraction. An even more important category of missed words is that of frequent words that do not frequently cooccur with other words. The loss would be considerable.

This is the reason to compute another RSV using a basic word-features vector space model. To combine both RSVs to one single score, we must first make them comparable by mapping them onto a common interval. To do so, we used *Max Norm*, as presented in [14], which permits to bring all positives scores within the range [0,1]:

$$New\ Score = \frac{Old\ Score}{Max\ Score}$$

Following this normalization, we aggregate both RSVs using a linear interpolation factor λ representing the relative weight of scores obtained with each technique (similarly as in [8]).

$$Aggregated\ Score = \lambda \cdot RSV_{Word\ Features} + (1-\lambda) \cdot RSV_{MFS}$$

The evidence of experiments with the INEX 2002 collection showed good results when weighting the single word RSV with the number of distinct word terms in the query (let a be that number), and the MFS RSV with the number of distinct word terms found in keyphrases of the query (let b be that number). Thus:

$$\lambda = \frac{a}{a+b}$$

For example, in Figure 4 showing topic 117, there are 11 distinct word terms and 7 distinct word terms occurring in keyphrases. Thus, for this topic, we have $\lambda = \frac{11}{11+7}$.

5. QUERY EXPANSION

Query expansion (QE) was used in two of the three runs that we submitted to INEX 2003. Both of these runs performed better than the one with no expansion at all. However, as the two official runs using QE also contained some other parameters that differed from those used in the run without QE (See Section 7 for a detailed description of the parameters.), these runs cannot be used to assess the performance of QE. We did a separate experiment to assess the performance of QE alone, and it showed that the average precision was increased by 11,5 % (from 0.0357 to 0.0398) when using the strict measure and by 44 % (from 0.0207 to 0.0298) when using the generalized measure. In the rest of this chapter we will first describe some background concepts of QE. In Section 5.2, we will describe our QE method, and in 5.3, we will describe further work in developing the method.

5.1 Background

It is generally agreed that modern variants of query expansion improve the results of a query engine [2]. However, there are many different ways in which QE can be performed. Some methods are based on relevance feedback, which can be blind or which can involve feedback from the user. In both cases, the QE approach is local because it is based

Document	MFS	Corresponding pairs	Matches	Quantity of relevance
d_1	AB	AB	AB	idf(AB)
d_2	ACD	AC CD AD	AC CD	idf(CD) + α_1·idf(AC)
d_3	AFB	AF FB AB	AB	idf(AB)
d_4	ABC	AB BC AC	AB BC AC	idf(AB) + idf(BC) + α_1·idf(AC)
d_5	ACB	AC CB AB	AC AB	idf(AB) + α_1·idf(AC)

Table 1: Quantity of relevance stemming from various MFSs w.r.t. a keyphrase query ABCD

on the retrieved set of documents. A global QE approach uses the the information derived from the whole document collection. Modern global QE methods usually use an automatically constructed thesaurus [11, 4]. Other methods are based on manually crafted thesauri, such as WordNet, but experimental studies have shown that if the expansion terms from such theasuri are selected automatically, QE can even degrade the performance of the system [16].

5.2 The Process

Our QE process can be considered a form of blind relevance feedback that has been inspired by the standard Rocchio way [12] of calculating the modified query vectors. However, it is different from the traditional relevance feedback framework in that it takes into account only positive terms and no negative terms and in that it does not take into account all of the terms in the fragments, but only the best ones. This limits in practice the number of expansion terms per QE iteration between zero and ten. However, experimental studies have shown that even a few carefully selected QE terms can considerably improve the performance of a system [15].

Here is the outline of the process:

1. Run EXTIRP. The output from EXTIRP is a set of ranked lists of document fragments. There is one list per topic and the fragments are ranked according to their RSVs with regard to the topic.
2. Take the ten topmost items of each list.
3. Calculate the similarity threshold value.
4. For each topic do:
   (a) Take those fragments whose RSV is greater than the similarity threshold value. Make a list of words occurring in these fragments followed by their frequency count, and sort by frequency.
   (b) Take the ten topmost words and expand the topic with them.
   (c) Multiply the weights of the old terms by two and give the new terms a weight of 1.
5. Run EXTIRP with the expanded topics.

We will now describe each of the steps in the process in more detail. In steps 1 and 5 EXTIRP is run with the same parameters and the RSV is calculated according to these. This means that the only things that change from the first iteration to the second are the keywords in the topic and the threshold value for similarity.

In step 3, the similarity threshold for a given topic is determined in the following way: Read the topmost RSV of the matches for each topic and maintain a list of the six smallest values. The threshold value is the highest one among the six smallest values. This way of determining the similarity threshold value implies that there are always at least six topics that are not expanded. The topics vary a lot and it is thus necessary to treat them differently from each other. The number six was determined by training the QE method on the topics and assesments of the year 2002. This step of determining the similarity threshold value is crucial to the success of the QE step, because running EXTIRP with different parameters results in radically different RSVs.

In step 4 (a), a list of words occurring in the fragments is created. This list is pruned from stopwords, and the remaining words are stemmed with the Porter stemmer[2][10]. A standard list for English language as well as a collection-specific list is used as a stopword list. The collection-specific list is created simply by gathering the most frequent terms in the collection.

In step 4 (c), the weights of the old terms are multiplied by two and the new terms are given a weight of 1. The possible weights of the old terms are: -1, 1 and 1.5. This means that the term weights in the expanded topic vectors can have the following values: -2, 1, 2 and 3. The topic vectors are normalized so that their length is one when they are processed by EXTIRP.

5.3 Improvement and further work

The above QE method can be developed further in many ways. We plan to treat different topics in more individual ways, run the method through more iterations and perform QE on negative query terms as well. For example, EXTIRP can be run separately for each topic instead of running it for all topics at the same time. This would mean a loop in step 4. In this loop, EXTIRP would be run for each topic until the RSVs of the resulting fragments reach a stable level. In this way, the number of iterations performed per topic would vary. The topics that perform well in the beginning would receive less attention than those which do not perform well in the beginning but that have a potential for improvement.

Expansion of negative query terms can be performed in a similar way as expansion of positive query terms. In negative

[2]The program was obtained from http://www.tartarus.org/ martin/PorterStemmer/

1. *Initialisation*:
   - $\forall n \in N$, score(n)=0
   - $\forall m \in M$, score(m)=RSV(m)
2. *Iterate*:
   - $\forall m \in M$: $\forall n_m \in N$ such that n_m is an ancestor of m, $score(n_m) = score(n_m) + score(m)$
3. *Final step*:
   - $\forall n \in N$, score(n)=$\frac{score(n)}{(size(n))^{UPF}}$

Figure 5: Greedy upward propagation algorithm.

expansion, we will run EXTIRP with the negative terms and expand the topics with those terms that are most common in the matches of this negative query.

6. UPWARD PROPAGATION OF MRU'S

The result of the previous steps is the assignment of an RSV to each MRU of the document collection. In this section, we propose a technique for assigning an RSV to each of their ancestors.

Its principle is to propagate upwards the relevance value of each MRU, weighting it upon the size of the corresponding element. We define the size of an element to be the sum of the sizes of all its MRU descendants. In turn, the size of an MRU is the number of characters it contains.

Let A be an XML document, N the set of elements of A, and M the set of MRUs of A. We compute the score of each element $n \in A$ as shown in Figure 5. UPF (*Upward Propagation Factor*) is a parameter that modulates the importance of the size of the elements. High UPF values give more penalty to big elements, and cause smaller ones to be promoted. On the other hand, if UPF=0, for any given article, the best score will always be given to the full article.

Because we assume that users go through answers in increasing rank order, we decided to avoid to propose them a document fragment they had already seen. Therefore, as a postprocessing, we decided to prune every element having an ancestor with a higher rank. This implies for instance, that if UPF=0, the set of answers will only contain full articles.

7. OUR RUNS

Our three official runs are described below. More details and the corresponding results are presented in Table 2.

- **UHel-Run1.**
  - Number of clusters: 200
  - MFS frequency threshold: $\sigma = 7$
- **UHel-Run2.**
  - Number of clusters: 100
  - MFS frequency threshold: $\sigma = 7$
- **UHel-Run3.**
  - Number of clusters: 100
  - MFS frequency threshold: $\sigma = 7$

The results of our first run are based on the paragraph-level division. Section-sized and bigger results are composites of the paragraph-sized fragments. Combining the paragraphs relies heavily on the upward propagation method described in Section 6. Due to their small size, paragraph-level fragments could benefit from Query Expansion more than bigger fragments, which partly explains the low evaluation scores of our first run. Also, small elements are more sensitive to changes in the fragment combination process.

The minimal result granularity of the second and the third run is a section. The section-level fragment count is substantially smaller than the corresponding paragraph count, which makes it slightly easier to find the best fragments for each query.

8. CONCLUSIONS

We came up with a new technique for exploiting the logical structure of XML documents so as to give more focused answers to information retrieval queries. We developed a system with the new ideas implemented, and the runs were submitted to INEX 2003. After preliminary observation, we notice that EXTIRP performs best at the very beginning of the top 1,500 answers where recall is relatively low. Considering the answers ranking between 1 and 50, our best runs are among the top of all submissions for CO topics.

There is a number of improvements to be achieved. First, we plan to reuse the clusterings formed prior to the extraction of maximal frequent sequences, aiming at query optimization. The idea is that by comparing queries to centroids of MRU clusters, we will be able to efficiently skip large quantities of MRUs, without having to compute similarity measures for each minimal unit individually.

Another concern is the fact that the current upward propagation formula is exponential in nature, meaning a small variation in the UPF factor can cause a switch from a set of answers with a large majority of minimal units to a set of answers with a large majority of full articles. Part of our future work is to explore the various ways to smooth this effect.

9. REFERENCES

[1] H. Ahonen-Myka. Finding All Frequent Maximal Sequences in Text. In *Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis, Ljubljana, Slovenia*, pages 11–17. J. Stefan Institute, eds. D. Mladenic and M. Grobelnik, 1999.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, ACM Press, New York, 1999.

[3] Y. Ben-Aharon, S. Cohen, Y. Grumbach, Y. Kanza, J. Mamou, Y. Sagiv, B. Sznajder, and E. Twito.

Runs	MRU Granularity	UPF	QE	strict	generalized
UHel-Run1	paragraph	2	no	0.0061 (51st)	0.0105 (46th)
UHel-Run2	section	2	yes	0.0323 (31st)	0.0222 (39th)
UHel-Run3	section	5	yes	0.0449 (20th)	0.0235 (38th)

Table 2: Results and ranks of our official runs (out of 56).

Searching in an XML Corpus Using Content and Structure. In *Proceedings of the Second Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, Schloss Dagsuhl, Germany, 2003.

[4] C. J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In *Proceedings of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 77–88, Copenhagen, Denmark, 1992.

[5] A. Doucet and H. Ahonen-Myka. Naive clustering of a large XML document collection. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 81–87, Schloss Dagsuhl, Germany, 2002.

[6] K. Hatano, H. Kinutani, M. Yoshikawa, and S. Uemura. Information Retrieval System for XML Documents. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002)*, pages 758–767, 2002.

[7] D. Hiemstra. A Database Approach to Content-based XML Retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 111–118, Schloss Dagsuhl, Germany, 2002.

[8] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. The Importance of Morphological Normalization for XML Retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 41–48, Schloss Dagsuhl, Germany, 2002.

[9] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200–214, 1987.

[10] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] Y. Qiu and H. Frei. Concept based query expansion. In *Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, PA, USA, 1993.

[12] J. J. Rocchio. *Relevance feedback in information retrieval. In Salton, G., editor, The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall Inc., 1971.

[13] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523, 1988.

[14] C. C. Vogt and G. W. Cottrell. Predicting the performance of linearly combined IR systems. In *Research and Development in Information Retrieval*, pages 190–196, 1998.

[15] E. Voorhees. Relevance feedback revisited. In *Proceedings of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–10, Copenhagen, Denmark, 1992.

[16] E. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, Dublin, Ireland, 1994.

[17] P. Willett. Recent trends in hierarchic document clustering: a critical review. *In Information Processing and Management*, 24(5):577–597, 1988.