Element Retrieval Using a Passage Retrieval Approach

Weihua Huang
rory_huang@hotmail.com

Andrew Trotman
andrew@cs.otago.ac.nz

Richard O'Keefe
ok@cs.otago.ac.nz

Department of Computer Science
University of Otago
Dunedin, New Zealand

Abstract

Element and passage retrieval systems are able to extract and rank parts of documents and return them to the user rather than the whole document. Element retrieval is used to search XML documents and identify relevant XML elements, while passage retrieval is used to identify relevant passages. This paper reports a series of experiments on element retrieval, using a general passage retrieval algorithm. Firstly, an XML document is divided into overlapping or non-overlapping fixed size windows (passages), then the relevant passages which contain query terms are found. Given the position of a passage in the XML document, the smallest element which contains this passage is found. The experiments were conducted with the INEX 2005 ad hoc test collection and evaluation tool. Two passage extraction methods, three weight functions and various window sizes were tested. A comparison with element retrieval systems was also conducted. The experimental results show that a robust passage retrieval algorithm can yield an acceptable level of performance in XML element retrieval.

Keywords Element retrieval, passage retrieval, XML retrieval, INEX.

1 Introduction

Both element and passage retrieval are able to extract and rank small relevant parts of a long document, which addresses some shortcomings of traditional whole-document retrieval systems. Element retrieval is used to search XML documents and to identify relevant XML elements. Passage retrieval is used when there is no mark-up — these algorithms identify relevant passages of text.

Element retrieval relies on the structure of XML documents in which the content is organized into smaller, nested structural elements. Each of these elements in the document's hierarchy, along with the document itself (the root of the hierarchy), is a retrievable unit [4]. Due to the nested hierarchical structure of XML documents, a query of an XML retrieval system can be expressed as a combination of content and structural conditions.

Proceedings of the 11th Australasian Document Computing Symposium, Brisbane, Australia, December 11, 2006.

Passage retrieval is the task of identifying and extracting fragments from heterogeneous full-text documents. A retrieved passage can be one or more sections, paragraphs, sentences, or a fixed number of words.

Our element retrieval system employs a passage retrieval algorithm that divides an XML document into passages by sliding a fixed size window across the document. The main purpose of this paper is to investigate the behaviour of the passage retrieval technologies for element retrieval, and then compare this approach with other element retrieval systems. Specifically, we want to compare the performance of previous passage retrieval algorithms (that ignore document semantics) with element retrieval algorithms (that do not). We compare our implementation to those of consistent good performers at INEX.

2 System Overview

The experiments are conducted as follows. Each XML document is divided into fixed size overlapping or non-overlapping windows (passages). If a window contains at least one query term, then it is a relevant passage. Given the element paths of the starting word and finishing word of this relevant passage, their common element ancestor is the smallest XML element that fully contains the passage. This element is then given the retrieval status values (RSV) of the passage. Overlapping elements are removed and, for each query, the first 1500 most highly scored elements are output as an INEX submission file. The score of a run is computed using the INEX assessment and evaluation tools.

We envisage a two pass retrieval system. First relevant documents are identified, then from that pool, relevant fragments are identified. So, in order to compare passage and element retrieval *on the same documents*, only those documents already identified by a search engine were examined. These documents were those identified by the IBM submission to INEX 2005, to which we compare our results.

The experiments investigated three weighting functions [1]:

2.1 Term Frequency Model (FREQ)

In this approach, the weight of each window is calculated by summing the total occurrences of all terms, t_i of a query Q within the window, W. The window RSV can be computed as follows [1]:

$$P(anyQterm|W) = \sum_{t_i \in Q} p(t_i|W)$$

The FREQ weighting approach is used as a baseline.

2.2 Query Generation Model (GEN)

In this approach the window RSV is computed as the probability of generating a query[1]:

$$P(Q|W) = \prod_{t_i \in Q} p_{mix}(t_i|W)$$

where

$$p_{mix}(t_i|W) = \lambda \times p(t_i|W) + (1-\lambda) \times p(t_i|D)$$

The mixing parameter, λ, is to smooth the estimates. In this work, λ is set to 0.8. In [1] Harper and Lee investigate the best value for λ and best results were obtained in the range 0.8 through 0.999. λ = 0.8 is the value they suggest using. The word probabilities are calculated as follows:

$$p(t_i|W) = n_{iW}/n_W \quad p(t_i|D) = n_{iD}/n_D$$

where $n_{iW}(n_{iD})$ and $n_W(n_D)$ are the number of term occurrences of term i in the window (document), and total term occurrences in the window (document) respectively. The log of both sides of the first formula is then taken:

$$\log P(Q|W) = \sum_{t_i \in Q} \log p_{mix}(t_i|W)$$

2.3 Kullback-Leibler Model (KL)

Using this approach, the window RSV is calculated as follows [1]:

$$KL(W|Q) = \sum_{t_i \in Q} p(t_i|W) \log(p(t_i|W)/p(t_i|D))$$

where

$$p(t_i|W) = (n_{iW} + 0.5)/(n_W + 1.0)$$

$$p(t_i|D) = (n_{iD} + 0.5)/(n_D + 1.0)$$

3 Test Set and Task

The test collection was the INEX 2005 *ad hoc* test set, which contains a document collection, a set of queries, and relevance assessments.

The INEX 2005 document collection comprises 16,819 scientific articles published between 1995 and 2004. It contains more than 10 million XML elements and is about 764MB in size.

With respect to the element retrieval task, the CO.Focussed sub-task was tested, which is to find non-overlapping relevant elements [2].

For the query set, the INEX 2005 CO query set containing 40 topics was used. All the queries used in the project contain exactly the same query terms as the <title> part of the topic. There are five queries that contain two words and others contain three or more words. There is no single word query.

This system did not apply stemming or use stop words which is likely to affect the results substantially.

To evaluate the system, the INEX 2005 official evaluation tool XCGEval was used. The official system-oriented evaluation was based on the *ep/gr* measures, with mean average (*MAep*) and interpolated mean average (*iMAep*) being the overall performance indicators. The experimental results were generated using generalised quantisation [2].

4 Experiment Objectives

Four questions are investigated:

- Could current passage retrieval algorithms be used to retrieve XML elements? If so then how well?

- A document can be divided into overlapping or non-overlapping passages. Which method produces better results?

- How does window size affect the performance of a passage retrieval system? Is there an optimal window size for the overlapping and non-overlapping window?

- The retrieval status value (RSV) of each window is computed by using a weighting function. Three passage weighting functions are investigated. Which one is best?

5 Experiment Results

5.1 Overlap Experiments

Two different passage extraction methods were investigated: overlapping windows and non-overlapping windows. The non-overlapping method is to divide a document into a set of fixed size non-overlapping blocks, such as pages. The overlapping approach, however, divides the document up into fixed size possibly overlapping blocks. For example, a heavily overlapping window may start from the second word of the previous window. Intuitively, this approach may report too many redundant results. To avoid this problem, the window is slid until the first word of the window is in the query. The overlapping window approach used ensures a reported window always begin with a query word, but need not end with a query word.

The overlapping and non-overlapping window results are presented in Figures 1, 2 and 3 for the best window sizes we discovered. Figure 1 shows the results of window size 100 on the FREQ weighting function. The overlapping window approach performs better than the non-overlapping one (MAep 0.0158 vs 0.0119), especially at the low recall levels. Figure 2 shows window

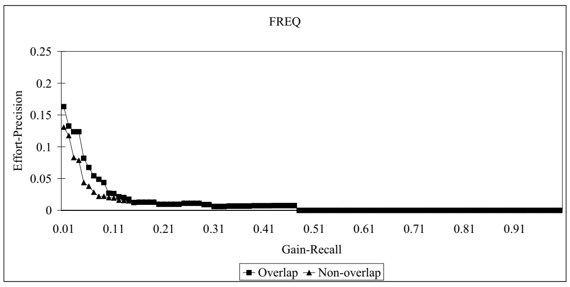

Figure 1: FREQ model, window size = 100

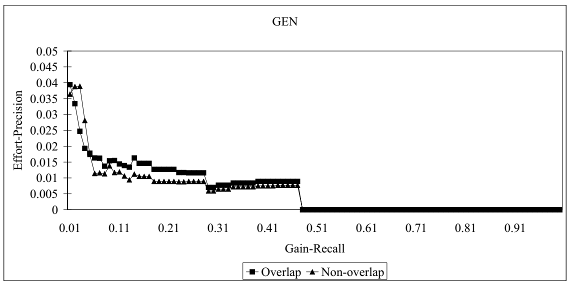

Figure 2: GEN model, window size = 250

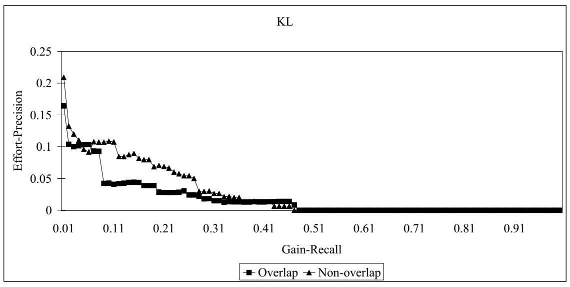

Figure 3: KL model, window size = 225

size 250 on the GEN model, in which the overlapping window approach also performs better than the non-overlapping one (0.0077 vs 0.0066).

Figure 3 shows window size 225 on the KL model. Surprisingly, on this model the overlapping window approach performs much worse than the non-overlapping one (0.0197 vs 0.0316).

5.2 Window Size Experiments

The second experiment investigated the effect of the window size on three weighting functions. The size of of the sliding window from 75 words up to 250 words increasing by 25 words each time.

This window size changing method was tested on three weight functions: the query generation model (GEN), the Kullback-Leibler model (KL), and the term frequency model (FREQ). The passage extracting method was overlapping windows.

The window size experimental results are presented in Tables 1, 2 and 3, where the **bold figures** are the best values achieved for each weighting function.

For the GEN model, smaller window sizes such as 75, 100 yield slightly better results than the bigger ones (125 – 200).

Table 2 and 3 show the performances of KL and FREQ model using overlapping passages. KL produces better results when the window size is large. On the other hand, FREQ model yields better results with small sized windows (75, 100).

A special window size test for KL model using non-overlapping passages was conducted. The results are shown in table 4.

When the run of 250 words size window finished, the experimental results showed that the KL model tends to favour large window sizes. To validate this conjecture, another six runs on window sizes from 275 to 400 were carried out. The results shows that KL produces better results when the window size is large.

window size	MAep	iMAep
75	**0.0077**	**0.0062**
100	**0.0077**	0.0061
125	0.0075	0.0058
150	0.0075	0.0057
175	0.0076	0.0059
200	0.0075	0.0060
225	**0.0077**	0.0061
250	**0.0077**	0.0060

Table 1: GEN_Model using overlapping passages

window size	MAep	iMAep
75	0.0149	0.0138
100	0.0169	0.0159
125	0.0178	0.0167
150	0.0183	0.0171
175	0.0186	0.0173
200	0.0194	0.0181
225	**0.0197**	**0.0182**
250	0.0182	0.0163

Table 2: KL_Model using overlapping passages

window size	MAep	iMAep
75	0.0145	0.0117
100	**0.0158**	**0.0124**
125	0.0138	0.0103
150	0.0138	0.0098
175	0.0138	0.0103
200	0.0130	0.0098
225	0.0131	0.0095
250	0.0129	0.0093

Table 3: FREQ_Model using overlapping passages

5.3 Weighting Function Experiments

In this section three weighting functions are compared. For comparing the performances of different weighting

window size	MAep	iMAep
75	0.0151	0.0137
100	0.0221	0.0204
125	0.0279	0.0259
150	0.0296	0.0267
175	0.0322	0.0294
200	0.0339	0.0303
225	0.0316	0.0277
250	0.0383	0.0337
275	0.0396	0.0344
300	0.0403	0.0349
325	0.0419	0.0361
350	0.0439	0.0365
375	0.0446	0.0371
400	**0.0454**	**0.0376**

Table 4: KL_Model using non-overlapping passages

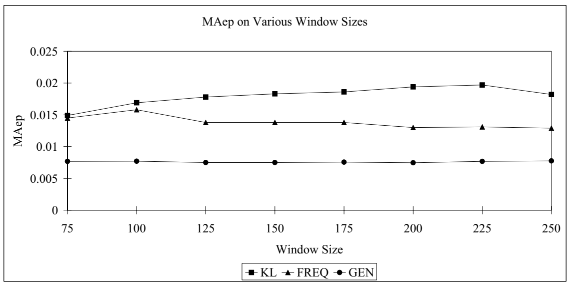

Figure 4: MAep on various window sizes

functions, the window size was fixed at various settings for the three weighting functions. The results are shown in figure 4.

Figure 4 shows that, for all the tested window sizes, the performance of the Kullback-Leibler model exceeds that of the other two weighting functions. Interestingly, it seems that FREQ performs better than GEN.

5.4 Comparison with Element Retrieval

The last experiment was to compare this system with element retrieval systems. Because the new system employed passage retrieval algorithms and passage weighting functions to retrieve XML elements, which has not been researched before (as far as we are aware), it is interesting to know how well the system performs by comparison to element retrieval.

To conduct this comparison, two sets of INEX 2005 submission files were chosen: IBM Haifa research lab (from which we took the relevant document list) which was ranked 4^{th} at INEX 2005 and University of Amsterdam which ranked 28^{th} (of 44).

The submission file for our system used non-overlapping passages. The window size was 300 words. The weight function was KL. We are aware that our results are overfitted and include them only to show that such an approach could be effective. No significance tests were conducted for the same reason.

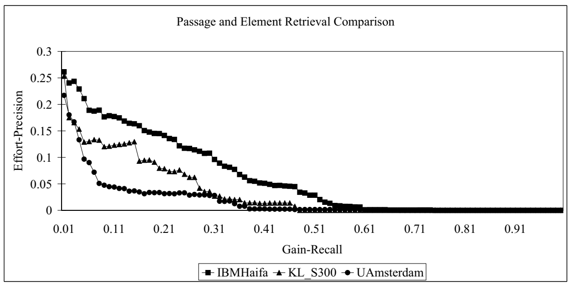

Figure 5: Comparison with element retrieval

Figure 5 shows that the run of IBM Haifa exceeds that of the other two systems. Our system performs better than that of the University of Amsterdam except at low recall levels.

6 Conclusion

This paper describes a comparison between passage and element retrieval approaches. The experiments were conducted with the INEX 2005 test collection and official evaluation tool. Two passage extracting approaches, three window weighting functions and various window sizes were tested. Major findings are:

- Given a robust passage retrieval algorithm and an XML parser, it is possible to retrieve elements efficiently.

- Compared with its non-overlapping counterpart, the overlapping window approach yields better results on the query generation model (GEN) and term frequency model (FREQ), but not on the Kullback-Leibler model (KL).

- There is a notable difference of performance when using different window sizes. The Kullback-Leibler (KL) favours larger windows.

- Among the three weight functions, the KL model outperforms the others. The query generation model (GEN) produced worse results than expected.

- Prior passage retrieval systems tuned for the document collection perform well compared to element systems tailored to the collection. It is reasonable to investigate tailoring a passage system to the collection and perhaps to include structural semantics in the algorithm.

References

[1] D. J. Harper and D. Lee (2004), On the Effectiveness of Relevance Profiling, *In Proceedings of the 9th ADCS*, pp. 10-16.

[2] G. Kazai and M. Lalmas (2005), INEX 2005 Evaluation Metrics, *In INEX 2005 Pre-proceedings*, pp. 401-406.

[3] Y. Mass and M. Mandelbrod (2004), Component Ranking and Automatic Query Refinement for XML Retrieval, *In Proceedings of INEX 2004*, pp. 73-84.

[4] B. Sigurbjörnsson, A. Trotman, S. Geva, M. Lalmas, B. Larsen and S. Malik (2005), INEX 2005 Guidelines for Topic Development, *In INEX 2005 Pre-proceedings*, pp. 375-384.