From Passages into Elements in XML Retrieval

Kelly Y. Itakura
David R. Cheriton School of Computer Science,
University of Waterloo
200 Univ. Ave. W.
Waterloo, ON, Canada
yitakura@cs.uwaterloo.ca

Charles L. A. Clarke
David R. Cheriton School of Computer Science,
University of Waterloo
200 Univ. Ave. W.
Waterloo, ON, Canada
claclark@plg2.uwaterloo.ca

ABSTRACT

Trotman and Geva [8] suggest that XML retrieval must move from element-based to passage-based because human assessors see passages when judging relevance. Since the current XML retrieval evaluation involves returning XML elements, they suggest ways to convert passage retrieval results into XML elements. In this paper, we implemented one of their algorithms and argue that the algorithm returns a lot of excessive text. We also implemented an element-based XML retrieval algorithm and analyze why it works better, linking its behavior to the other algorithm of Trotman and Geva. We finally compare the results of these two implemented algorithms to a gold standard obtained by passage retrieval to compare the excess and the lack of text in the result sets. We conclude the paper by suggesting a better way to represent results of XML retrieval.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - retrieval models.

General Terms

Algorithms, Experimentation, Theory

Keywords

XML retrieval, passage retrieval

1. INTRODUCTION

INEX [1] is an evaluation forum for XML retrieval. In the adhoc track, a *topic* contains a user's information need, including structural information as well as a set of query terms called a *title*. A title is what we normally type into a search engine; phrases are contained within double quotes, terms that must be in the returned elements are headed by the plus sign, terms that must not be in the returned elements are headed by the minus sign. The content-oriented (CO) task requires processing only titles. In the focused task, the result must be a set of *single* elements that are "the most exhaustive and specific" [6]. In this paper, we address adhoc, content-oriented, focused task.

Trotman and Geva [8] mentions that human assessors see passages when making a relevance judgement on results of XML retrieval. To transition into passage based XML retrieval, they propose how to convert elements into passages and passages into elements. In particular, they suggest a couple of ways to convert passages into XML elements for the focused retrieval task. This process involves finding an appropriate size of an XML element to return that is not redundant but contains necessary passages. The first algorithm of Trotman and Geva [8] to convert a passage into an XML element, takes the smallest XML element that contains the passages. The second algorithm takes the largest XML element that is contained in the passages. We call the first algorithm TG+ retrieval and the second algorithm TG− retrieval. Trotman and Geva cast doubts on specificity of the TG+ algorithm and exhaustivity of the TG− algorithm. That is, even though TG+ retrieval returns elements that contains relevant text, it may contain too much irrelevant text as well. On the other hand, TG− retrieval may return elements without much irrelevant text, but it may miss too much relevant text. In this paper, we implemented the TG+ algorithm to show that the returned results contain a lot of irrelevant text. We also implemented an algorithm analogous to TG− retrieval, and compared the exhaustivity and specificity of these two algorithms against a defined gold standard, a set of passages retrieved by a variant of BM25 [7].

2. METHODOLOGY

As a test collection, we used INEX 2005 IEEE collection and the query topics provided for the INEX 2005 adhoc track. The corpus contains $16,819$ files from various IEEE journals from 1995 to 2004. There are 39 topics, each containing a set of query terms. In INEX 2005, the results were assessed using the nxCG metric [5]. We only consider the generalized quantization because this is the only metric with published ranking in both INEX 2005 and 2006. In addition, we only ran the focused task because this task requires balancing exhaustivity and specificity, which is the topic of this paper.

In this section, we describe three different algorithms to obtain passages/XML elements. All these algorithms process a title into a set of disjunctive terms, separating phrases into terms, removing the plus sign, and ignoring terms preceded by the minus sign. We did not remove duplicate query terms within a topic and across topics. Finally, we used the Wumpus Information Retrieval System [2] to stem query terms and retrieve all positions of query term occurrences in the collection.

All three algorithms used a variant of Okapi BM25 [7]

SIGIR 2007 Workshop on Focused Retrieval
July 27,2007, Amsterdam, The Netherlands.

to score passages or elements. Normally, Okapi BM25 is used for scoring documents. Its effectiveness against passage retrieval is not yet fully established. When used in passage retrieval, a score of a passage or an element P is defined as follows:

$$score(P) \equiv \sum_{t \in Q} W_t \frac{f_{P,t}(k_1+1)}{f_{P,t} + k_1(1 - b + b\frac{|P|}{avgdl})}, \quad (1)$$

For a weight of a query term t, W_t, we used a *document-level* IDF value,

$$W_t = \log \frac{\text{total \# of documents}}{\text{\# of documents containing a term } t}.$$

When none of the documents contain a term t, we set the IDF of the term to zero. The average document length, *avgdl* is also computed at the document level. In our corpus, the average document length is 6147.97 terms. When computing the length of an XML element or a passage, we ignored XML tags. The number of time a term t appears in the passage P is denoted $f_{P,t}$. Parameters k_1 is positive, and $0 < b < 1$. This way, we can view scoring P as scoring a small document in the context of all documents in the collection.

2.1 Passage Retrieval

In passage retrieval, we disregarded all XML structures, and retrieved passages that start and end with query terms. First, we scored all such passages and ignored those that are less than 25 words long. Then we removed all the nested passages to return the top 1500 passages for each topic. The elimination of nesting involved adding to the ranking only when a passage does not contain the higher ranking passages within it and the passage is not contained in the higher ranking passages. We call the resulting set the *gold standard*, used as the basis for comparison because assessors look for elements that contain what they consider important passages.

2.2 Element Retrieval

In element retrieval, we computed the scores of all XML elements of interest taken from [3]. These are `abs app article bb bdy bm fig fm ip1 li p sec ss1 ss2 vt`. We ignored elements that are less than 25 words long, or have a zero score. We then eliminated the nesting of XML elements to get the top 1500 XML elements to return.

2.3 TG+ Retrieval

In this section, we describe how we implemented the TG+ algorithm, the first algorithm of Trotman and Geva [8] to convert passages into XML elements.

In the TG+ algorithm, after computing all passage scores, we assigned to an XML element the score of the highest scoring passages whose smallest ancestor is the XML element. We ignored XML elements that are not of interest, and those that are less than 25 words. Finally, we eliminated the nestings of the XML elements to return the top 1500 XML elements.

Figure 1 illustrates this approach. Suppose we have four passages with Okapi scores; passage 1 with a score of 5.2 to passage 4 with a score of 4.0. Passage 1 spans through paragraph 1, 2, and 3, which are under section 1 and an article. Similarly for other passages. After we computed Okapi

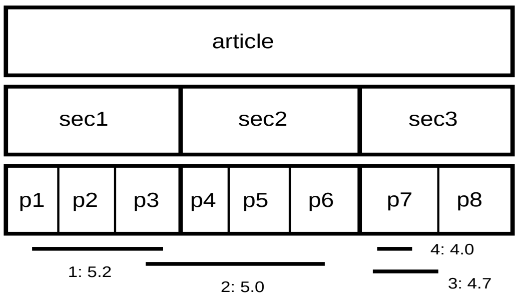

Figure 1: Assigning Scores and Nesting Elimination in TG+ Retrieval

Table 1: TG+ and Element Retrieval at $k_1 = 10$ and $b = 0.9$ in INEX 2005 CO-focused

	nxCG[10]	nxCG[25]	nxCG[50]	MAep	iMAep
TG+	0.1856	0.1774	0.1633	0.0612	0.0387
Element	0.2586	0.2323	0.217	0.0929	0.0715

scores for these passages, we assigned scores to corresponding XML elements. The score of 5.2 is assigned to `sec[1]`, the smallest element containing passage 1. Similarly, the score of 5.0 is assigned to `article`. Because passage 3 has a higher score than passage 4, we assign the score of passage 3, 4.7 to `p[7]`. Next, we get rid of nesting while taking the top scored elements. We first take the element with the highest score, `sec[1]` and assign a rank of 1. The element with the second highest score, `article` is eliminated because it causes a nesting of `sec[1]` within it. We can safely take the element with the next highest score, `p[7]` because it does not cause a nesting with `sec[1]`, and assign a rank of 2.

3. EVALUATION AND ANALYSIS

3.1 Performance Against INEX 2005

We trained both element retrieval and TG+ retrieval over the INEX 2005 corpus and the query set. Figure 2 and Figure 3 show the scores for different values of k_1 with $b = 0.8$ using the nxCG metric, mean average precisions (MAep) and interpolated MAep (iMAep) in the CO.Focused task. We then chose $k_1 = 10$ for TG+ retrieval and $k_1 = 4$ for element retrieval for training b as seen in Figure 4 and Figure 5. The results of nxCG and MAep/iMAep metrics seem correlated as both have similar curves and give maximum values at the same parameters. The values of both k_1 and b need to be quite large for both algorithms to perform well. Clarke [3] also points out this phenomenon for his Okapi-based passage retrieval algorithm that is quite different from these two. Having a large k_1, therefore, seems to be necessary for using Okapi BM25 for passage retrieval.

These figures suggest that the simple element retrieval performs much better than TG+ retrieval. To see why, we compared the results of both algorithm at $k_1 = 10$ and $b = 0.9$, which are optimal parameters for TG+ retrieval. Table 1 shows that even if at an optimal setting, TG+ re-

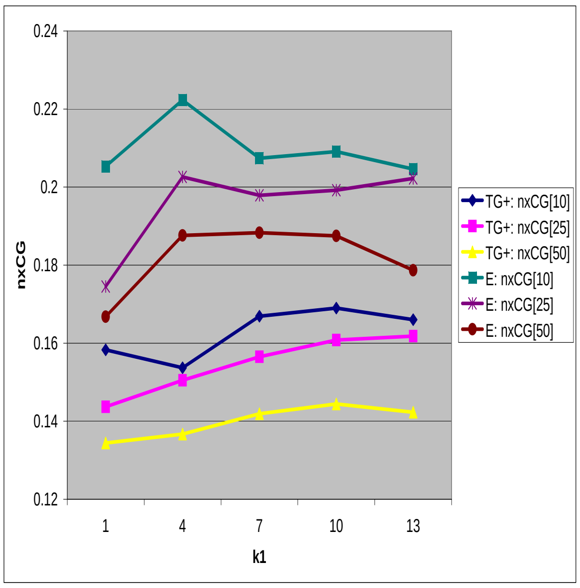

Figure 2: nxCG: Training on Different k_1 with $b = 0.8$ in INEX 2005 CO-Focused

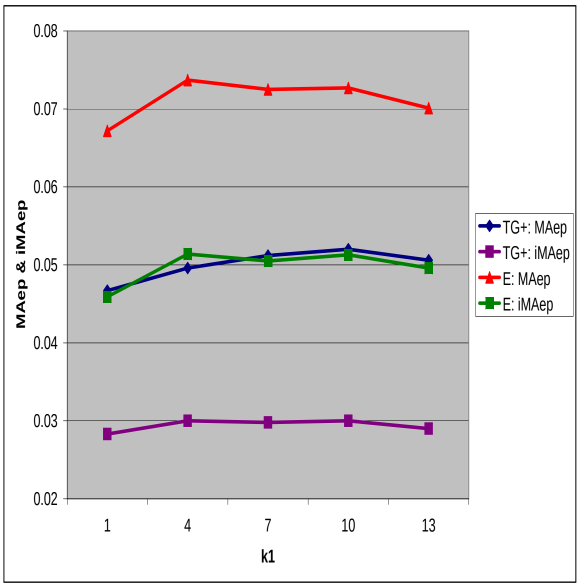

Figure 3: MAep/iMAep: Training on Different k_1 with $b = 0.8$ in INEX 2005 CO-Focused

trieval performs significantly worse than element retrieval.

We analyzed the submission files of TG+ and element retrieval and realized that in the TG+ algorithm, the majority of the elements returned had a large granularity; many of them were either `/article` or `/article/bdy`. In the element retrieval, however, the returned elements had much finer granularity such as paragraphs and sections.

For example, as wee see in Figure 6, the first element returned for the first topic in both retrieval methods is from the same file, `ex/2001/x1026` that spans through positions 27040000 and 27046835. In TG+ retrieval, the element returned was `/article/bdy`, corresponding to positions 27040246 through 27045776 with a score of 42.53. The passage corresponding to this element that gives the score of 42.53 spans through positions 27042389 and 27043619. In element retrieval, the element returned was `/article/bdy/sec[4]` corresponding to positions 27042506 through 27043559 with a score of 40.90. We see that `/article/bdy`, the smallest element that contains the highest scoring passage, is much longer than the passage. However, /article/bdy/sec[4], the element returned in element retrieval contains much of the passage without much excessive text.

The element, `/article/bdy/sec[4]`, returned by element retrieval was not returned by TG+ retrieval because the highest scoring passage within the element that spans through positions 27042550 and 27043534 only scored 41.02, lower than the highest passage contained in `/article/bdy`. On the other hand, element retrieval did not return `/article/bdy` because its score, 39.44 is lower than the score of `/article/bdy/sec[4]`, 40.90.

The above observation suggests that the TG+ algorithm is not a good approach for converting passages into an XML element because it returns a lot of excessive text. The TG− algorithm, taking the largest XML element contained in a passage, would likely perform well for the same reason element retrieval performs well; the returned elements would be unlikely to contain too much excessive text. However, both element retrieval and TG− retrieval may miss too much text to reduce the overall performance.

3.2 Performance Against the Gold Standard

In the previous section, we observed that TG+ retrieval tends to return excessive text. We also speculate that element retrieval may be missing too much text. To measure how much text is missing or in excess for both algorithms, we compared their results against our gold standard as follows.

First, because our goal is to convert passages into XML elements, and human assessors see passages when making relevance judgement, the gold standard must be passages. The best XML elements are those that cover the gold-standard passages sufficiently, but not much more. We created the gold standard using passage retrieval with the same parameters as TG+ retrieval, $k_1 = 10$ and $b = 0.9$.

Next, we compared a set of passages/XML elements returned by both TG+ and element retrieval against the gold standard at each rank up to 1500. The percent *lack* at rank r is defined as the percentage of the gold standard up to rank r that is not covered by the returned elements up to rank r. We can think of percent lack at rank r as how much text a user is missing (the user wants to read passages in

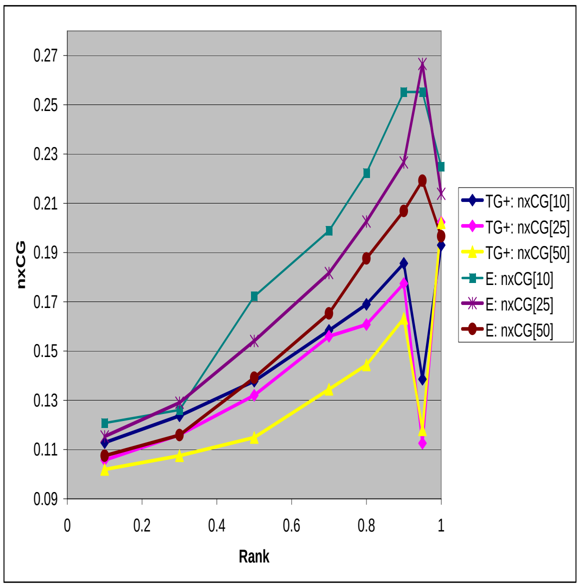

Figure 4: nxCG: Training on Different b in INEX 2005 CO-Focused

the gold standard) when the user reads from rank 1 to rank r. Similarly, the percent *excess* at rank r is defined as the percentage of the returned elements up to rank r that does not cover the gold standard up to rank r. We can think of percent excess as how much text a user reads that the user did not have to read (because the user only wants to read passages in the gold standard) when the user reads from rank 1 to rank r. We averaged both the percent lack and the excess over all topics.

Figure 7 show that overall, the excessive text returned by TG+ retrieval is larger than the excessive text returned by element retrieval. However, as the rank increases, the amount of excessive text for element retrieval increases to the point that towards the end of the ranking, the level of excess for both algorithms are about the same. Figure 8 shows that element retrieval misses much more text than TG+ retrieval does, and the general trend for both algorithms is to have less missing text as rank increases.

The reason that element retrieval performs better on the nxCG, MAep, and iMAep metric for the focused task is because having excessive text is punished more than missing text. In the INEX focused task, specificity is preferred to exhaustivity [6]. In the same manner, the TG− algorithm, that takes the largest element contained in the passage, will likely score high in the focused task. The TG+ algorithm would score high in other tasks that place preference on exhaustivity over specificity.

4. DISCUSSION

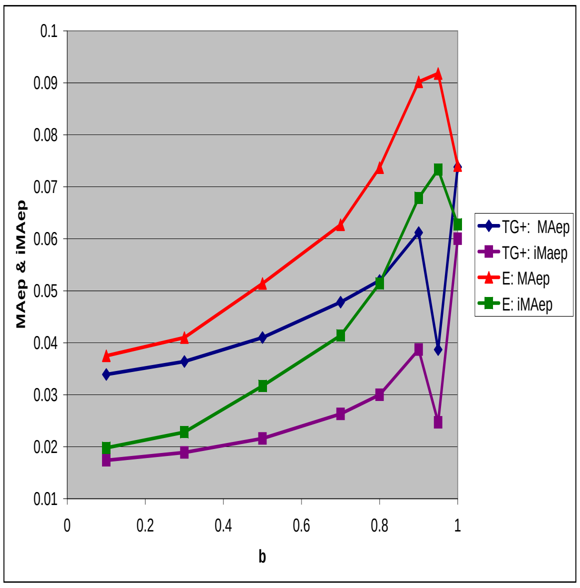

Figure 5: MAep/iMAep: Training on Different b in INEX 2005 CO-Focused

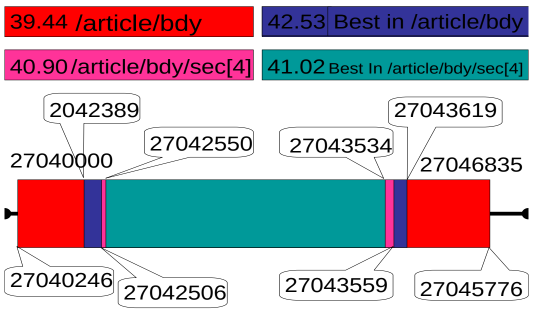

Figure 6: Elements and Passages Relating to Rank One Result

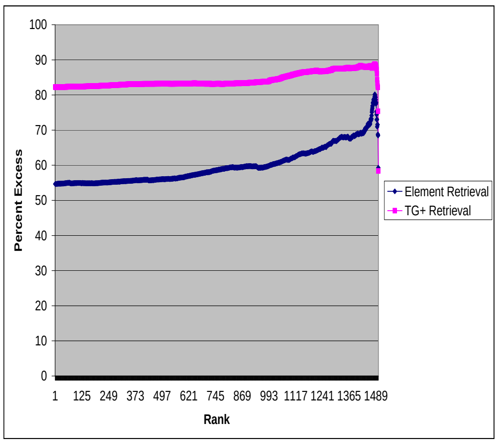

Figure 7: Percent Excess of the Gold Standard

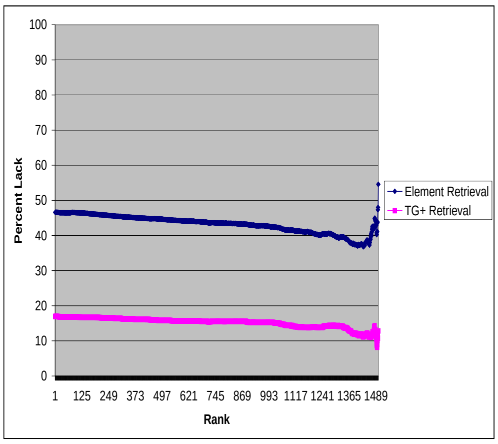

Figure 8: Percent Lack of the Gold Standard

Table 2: Average Length in Passage, Element, and TG+ Retrieval

	Final	Intermediate	Nested
Passage	1760.56	N/A	2424.85
Element	750.87	N/A	1354.69
TG+	3322.06	770.72	874.56

In this section, we discuss the behavior of parameters k_1 and b in passage retrieval. We used an average document length for $avgdl$ in computing an Okapi score for a passage in Equation 1. Because most passages are less than a size of a document, $|P|/avgdl$ is less than 1 most of the time. The result is that in the denominator, we multiply k_1 with something very small. Then if k_1 is also small, the denominator approaches to $f_{P,t}$, turning Equation 1 into

$$score(P) \equiv \sum_{t \in Q} W_t 1. \tag{2}$$

Therefore, one reason for a large k_1 value is to prevent Equation 1 from degenerating into Equation 2 when the length of P is smaller than the average document length. Similarly, a large b augments the effect of length normalization in $b|P|/avgdl$ in the denominator, and this sets off the large gap between the average document length and a passage length. Therefore, it appears that the choice of the average document length is balanced by the choices of k_1 and b.

To see how small the passage lengths are compared to the average document length, we computed various passage lengths. Table 2 shows the average length of the top 1500 final results (Final), the passages that produced the top 1500 final results (Intermediate) (only applicable to TG+), and the top 1500 results before the elimination of nesting (Nested), for passage, element, and TG+ retrieval. From the average document length of 6147.97, we compute the average passage length to be 1538.47. Then both the TG+ and the element algorithms retrieve fairly small elements/passages, 770.72 for TG+ retrieval, and 750.87 for element retrieval, compared to the average passage length of 1538.47. However, the average length of the elements returned by TG+ retrieval is about four times as large (3322.06) as the average length of the corresponding passages (770.72). This shows that the TG+ algorithm is inherently ineffective for the following reasons. The passages returned by TG+ algorithm is about the right size because this is about the same size as the elements returned by element retrieval that performs well. Therefore, it is the way we assign passage scores to elements, rather than the choice of $avgdl$, k_1, and b, that makes TG+ retrieval less effective than element retrieval.

Finally, it is interesting to note that the average length of the passages returned by the passage retrieval, 1760.56 is close to the average passage length of 1538.47. The difference between the average lengths of the passages returned by TG+ and passage retrieval may be accounted for by how nesting was eliminated. In passage retrieval, we eliminated nestings of the passages that eliminates more nesting than nesting elimination of elements in TG+ retrieval. Furthermore, the fact that the average passage length returned by passage retrieval is close (in fact more) than the average passage length implies that TG− retrieval would likely be able to find the largest element within the passages. However, we believe that the best elements to return in XML retrieval is multiple consecutive elements that contain passages just enough.

5. RELATED WORK

Huang et al. [4] implemented their own version of the TG+ algorithm, where the method of identifying passages are different from ours. Instead of considering all possible passages as we did, they considered a fixed size passages obtained from sliding windows. Moreover, they first performed document retrieval, and then ran passage retrieval on the top scoring documents, whereas we directly retrieved high scoring passages. When scoring passages, they used a simple term frequency approach and two language modeling approaches, whereas we adopted a variant of Okapi BM25, which is used for document retrieval.

The effectiveness of the TG+ algorithms of Huang et al. and ours can be easily compared because both of us used the same test set, the INEX 2005 IEEE collection, ran the same CO.Focused task, and used the nxCG generalized quantization to score the results. Huang et al. compared their passage based XML retrieval results against the INEX 2005 CO-Focused task submissions of IBM Haifa that ranked 4th and of University of Amsterdam that ranked 28th in the Mean Average Precision (MAep) ranking. They concluded that their algorithm ranked between these two. Our results of the TG+ algorithm at $k = 10$ and $b = 0.9$ with MAep of 0.0612 also ranks between these two. On the other hand, our element retrieval algorithm with the same parameters that has a MAep of 0.0929 easily ranks the first preceding the first result of IBM Haifa ranked 1st, that has a mean average precision of 0.0917.

Huang et al. conclude the paper by mentioning that a passage retrieval algorithm can produce effective element retrieval results because it ranked between the 4th and the 28th out of 44 submissions. However, the fact that both versions of TG+ retrieval, despite with very different implementations of passage retrieval, *only* ranked between the 4th and the 28th implies that TG+ retrieval is not a good idea for turning passages into an element. The comparison of our TG+ retrieval against our gold standard along with the average length statistics in Section 4 also implicate that it does not perform well because of excessive text inherent to the very idea of TG+ retrieval.

6. CONCLUSIONS AND FUTURE WORK

We implemented three algorithms to test the effectiveness of the first algorithm of Trotman and Geva [8], which converts the results of passage retrieval into XML elements. This TG+ algorithm takes the smallest XML elements that contains the passages. It appears that the parameter k_1 in Okapi BM25 must be high for any passage retrieval to yield a good performance. We showed that the TG+ algorithm does not perform as well as the simple element retrieval, where we scored all XML elements, not passages. By comparing the result sets of the TG+ and the element retrieval algorithm, we realized that the TG+ algorithm tends to return a lot of excessive text. Even though element retrieval performs well, we speculated that it may miss too much text.

To see how much of text is missing or excessive, we created the gold standard, a set of results obtained from passage retrieval using the same Okapi BM25 parameters. We then computed an average percent excess and an average percent lack of text for returned results over all topics. Although element retrieval does miss out a lot of text, it scores well on the focused task of INEX because the task has a preference of specificity over exhaustivity. Because the TG− algorithm that returns the largest XML elements contained in the passages will not have much excessive text as in the case of element retrieval, the performance of the TG− algorithm may be similar to the one obtained by element retrieval. Ultimately though, the best elements to return must be a series of XML elements.

Future work involves studying the effectiveness of Okapi BM25 in passage retrieval setting, including analyzing why a large k_1 value works. We also would like to implement the TG− algorithm to compare it against the TG+ and the element algorithms. Finally, we would like to implement multiple passage retrieval and measure its performance.

7. REFERENCES

[1] INEX: INitiative for the Evaluation of XML Retrieval . Accessible at `http://inex.is.informatik.uni-duisburg.de`, 2007.

[2] S. Büttcher. the Wumpus Search Engine. Accessible at `http://www.wumpus-search.org`, 2007.

[3] C. L. A. Clarke. Controlling overlap in content-oriented XML retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 314–321, New York, NY, USA, 2005. ACM Press.

[4] W. Huang, A. Trotman, and R. O'Keefe. Elemement retrieval using a passage retrieval approach. In *the 11th Australian Document Computing Symposium*, 2006.

[5] G. Kazai and M. Lelmas. INEX 2005 evaluation metrics. *Springer-Verlag, Lecture Notes in Computer Science*, 3977:16–29, 2006.

[6] S. Malik, G. Kazai, M. Lelmas, and N. Fuhr. Overview of INEX 2005. *Springer-Verlag, Lecture Notes in Computer Science*, 3977:1–15, 2006.

[7] S. Robertson, S. Walker, and M. Beaulieu. Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive track. *7th Text REtrieval Conference*, 1998.

[8] A. Trotman and S. Geva. Passage retrieval and other XML-retrieval tasks. *SIGIR 2006 Workshop on XML Element Retrieval Methodology*, August 2006.