Indian Statistical Institute at INEX 2007 Adhoc Track: VSM Approach

Sukomal Pal and Mandar Mitra

Information Retrieval Lab, CVPR Unit,
Indian Statistical Institute, Kolkata
India
{sukomal_r,mandar}@isical.ac.in

Abstract. This paper describes the work that we did at Indian Statistical Institute towards XML retrieval for INEX 2007. As a continuation of our INEX 2006 work, we applied the Vector Space Model and enhanced our text retrieval system (SMART) to retrieve XML elements against the INEX Adhoc queries. Like last year, we considered Content-Only(CO) queries and submitted two runs for the FOCUSED sub-task. The baseline run does retrieval at the document level; for the second run, we submitted our first attempt at element level retrieval. This run uses a very naive approach and performs poorly, but the relative performance of the baseline run was fairly encouraging. After the official submissions, we conducted a few more experiments involving both document-level and element-level retrieval. These additional runs yield some improvements in retrieval effectiveness. We report the results of those runs in this paper. Though our document-level runs are promising, the element-level runs are still far from satisfactory. Our next step will be to explore ways to improve element-level retrieval.

1 Introduction

Traditional Information Retrieval systems return whole documents in response to queries, but the challenge in XML retrieval is to return the most relevant parts of XML documents which meet the given information need. INEX 2007 [1] marks a paradigm shift as far as retrieval granularity is concerned. This year, arbitrary passages are also permitted as retrievable units, besides the usual XML elements. A retrieved passage can be a sequence of textual content either from within an element or spanning a range of elements. INEX 2007 also classified the adhoc retrieval task into three sub-tasks: a) the FOCUSED task which asks systems to return a ranked list of elements or passages to the user; b) the RELEVANT in CONTEXT task which asks systems to return relevant elements or passages grouped by article; and c) the BEST in CONTEXT task which expects systems to return articles along with one best entry point to the user.

Each of the three subtasks can be based on two different query variants: Content-Only(CO) and Content-And-Structure(CAS) queries. In the CO task,

N. Fuhr et al. (Eds.): INEX 2007, LNCS 4862, pp. 122–128, 2008.

the user poses the query in free text and the retrieval system is supposed to return the most relevant elements/passages. A CAS query can provide explicit or implicit indications about what kind of element the user requires along with a textual query. Thus, a CAS query contains structural hints expressed in XPath-like [2] syntax, along with an *about()* predicate.

Our retrieval approach this year was based on the Vector Space Model which sees both the document and the query as bags of words, and uses their *tf-idf* based weight vectors to measure the inner product *similarity* between the document and the query. The documents are retrieved and ranked in decreasing order of the similarity value.

We used the SMART system for our experiments at INEX 2007 and submitted two runs for the *FOCUSED* sub-task of the Adhoc track considering CO queries only. We performed some additional experiments after the submission. In the following section, we describe our general approach for all these runs, and discuss results and further work in Section 3.

2 Approach

To extract the useful parts of the given documents, we manually shortlisted about thirty tags that were found to contain useful information: <p>, <ip1>, <it>, <st>, <fnm>, <snm>, <atl>, <ti>, <p1>, <h2a>,<h>, <wikipedialink>, <section>, <outsidelink>, <td>, <body>, etc. Documents were parsed using the libxml2 parser, and only the textual portions included within the short-listed tags (see above) were used for indexing. Similarly, for the topics, we considered only the *title* and *description* fields for indexing, and discarded the *inex-topic*, *castitle* and *narrative* tags. No structural information from either the queries or the documents was used.

The extracted portions of the documents and queries were indexed using single terms and a controlled vocabulary (or pre-defined set) of statistical phrases following Salton's blueprint for automatic indexing [3]. Stopwords were removed in two stages. First, we removed frequently occurring common words (like *know*, *find*, *information*, *want*, *articles*, *looking*, *searching*, *return*, *documents*, *relevant*, *section*, *retrieve*, *related*, *concerning*, etc.) from the INEX topic sets. Next, words listed in the standard stop-word list included within SMART were removed from both documents and queries. Words were stemmed using a variation of the Lovins stemmer implemented within SMART. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package (NSP)[1] on the English Wikipedia text corpus and selected the 100,000 most frequent word bi-grams as the list of candidate phrases. Documents and queries were weighted using the *Lnu.ltn* [4] term-weighting formula. For each of 130 adhoc queries(414-543), we retrieved 1500 top-ranked XML documents or non-overlapping elements.

[1] http://www.d.umn.edu/∼tpederse/nsp.html

2.1 Document-Level Run

For the baseline run, *VSMfb*, we retrieved whole documents only. We had intended to use blind feedback for this run, but ended up inadvertently submitting the results of simple, inner-product similarity based retrieval.

Later, we conducted the actual feedback run, in which we applied automatic query expansion following the steps given below for each query (for more details, please see [5]).

1. For each query, collect statistics about the co-occurrence of query terms within the set $\mathcal{S}$ of 1500 documents retrieved for the query by the baseline run. Let $df_{\mathcal{S}}(t)$ be the number of documents in $\mathcal{S}$ that contain term t.
2. Consider the 50 top-ranked documents retrieved by the baseline run. Break each document into overlapping 100-word windows.
3. Let $\{t_l, \ldots, t_m\}$ be the set of query terms (ordered by increasing $df_{\mathcal{S}}(t_i)$) present in a particular window. Calculate a similarity score *Sim* for the window using the following formula:

$$Sim = idf(t_1) + \sum_{i=2}^{m} idf(t_i) \times \min_{j=1}^{i-1}(1 - P(t_i|t_j))$$

   where $P(t_i|t_j)$ is estimated based on the statistics collected in Step 1 and is given by

$$\frac{\#\ documents\ in\ \mathcal{S}\ containing\ words\ t_i\ and\ t_j}{\#\ documents\ in\ \mathcal{S}\ containing\ word\ t_j}$$

   This formula is intended to reward windows that contain multiple matching query words. Also, while the first or "most rare" matching term contributes its full idf (inverse document frequency) to *Sim*, the contribution of any subsequent match is deprecated depending on how strongly this match was predicted by a previous match — if a matching term is highly correlated to a previous match, then the contribution of the new match is correspondingly down-weighted.
4. Calculate the maximum *Sim* value over all windows generated from a document. Assign to the document a new similarity equal to this maximum.
5. Rerank the top 50 documents based on the new similarity values.
6. Assuming the new set of top 20 documents to be relevant and all other documents to be non-relevant, use Rocchio relevance feedback to expand the query. The expansion parameters are given below:

$$\begin{aligned} \text{number of words} &= 20 \\ \text{number of phrases} &= 5 \\ \text{Rocchio } \alpha &= 4 \\ \text{Rocchio } \beta &= 4 \\ \text{Rocchio } \gamma &= 2. \end{aligned}$$

Finally, for each topic, 1500 documents were retrieved using the expanded query. This unofficial run is named *VSMfeedback*.

2.2 Element-Level Run

This year, we also attempted element-level retrieval for the first time. Since SMART does not natively support the construction of inverted indices at the element level, we adopted a 2-pass strategy. In the first pass, we retrieved 1500 documents for each query using query expansion.

In the second pass, these documents were parsed using the libxml2 parser, and leaf nodes having textual content were identified. Figure 1 shows a fragment of a file from the wikipedia collection. The leaf nodes that have textual content are enclosed in rectangles in the figure. The total set of such leaf-level textual elements obtained from the 1500 top-ranked documents were then indexed and compared to the query as before to obtain the final list of 1500 retrieved elements.

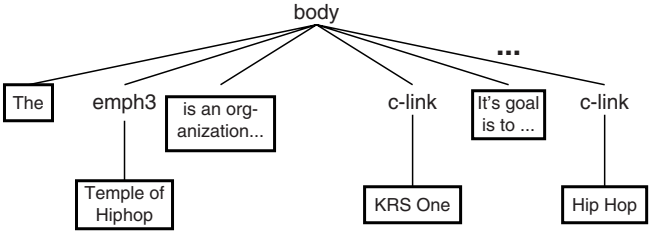

Fig. 1. Parse tree for a fragment of a wikipedia document

Since we considered only the leaf nodes as retrievable elements for *VSMfbElement*, the retrieved elements for the official run are automatically non-overlapping. However, as is clear from Figure 1, permitting only leaf-level textual elements to be retrieved has an obvious disadvantage: nodes such as <p> or <body> are very often not considered retrievable elements, because of the occurrence of nodes like <emph3> and <collectionlink> under the <p> or <body> node. It is not surprising, therefore, that the *VSMfbElement* run performs poorly (see the next section for details).

Post-submission, we incorporated within SMART the capability to retrieve elements at intermediate (i.e. non-leaf) levels of the XML tree. Retrieval results obtained using this capability are labelled *VSM-fdbk-elt* in the tables below. The *VSM-fdbk-elt* run is similar to the *VSMfbElement* run described earlier, and uses a 2-pass strategy. The difference is that, in the new run, the query is compared to all elements that contain text, instead of only the leaf-level textual nodes. In order to avoid any overlap in the final list of retrieved elements, the nodes for a document are sorted in decreasing order of similarity, and all nodes that have an overlap with a higher-ranked node are eliminated.

3 Results

The results for the official and unofficial runs (according to the updated evaluation script and relevance judgments for 107 topics) are shown in Tables 1 and 2 respectively.

Table 1. Official results for Element Retrieval (FOCUSED task, CO queries)

Run Id	P@0.00		P@0.01		P@0.05		P@0.10		MAiP	
	Score	Overall rank	Score	Overall rank	Score	Overall rank	Score	Overall rank	Score	Overall rank
VSMfb	0.4680	49	0.4524	39	0.3963	20	0.3797	14	0.1991	4
VSMfbElement	0.2406	76	0.1820	73	0.0990	74	0.0548	74	0.0159	76
BEST run	0.6056		0.5271		0.4697		0.4234		0.2238	

Table 2. Unofficial results for Element Retrieval (FOCUSED task, CO queries)

Run Id	P@0.00	P@0.01	P@0.05	P@0.10	MAiP
VSMfeedback	0.4839	0.4682	0.4236	0.3957	0.2116
VSM-fdbk-elt-slope0.2	0.4724	0.4171	0.3143	0.2497	0.0787
VSM-fdbk-elt-slope0.3	0.4873	0.4318	0.3358	0.2620	0.0803
VSM-fdbk-elt-slope0.4	0.5032	0.4558	0.3379	0.2374	0.0742
BEST run	0.6056	0.5271	0.4697	0.4234	0.2238

The first official run (VSMfb) fared quite well except at the early recall points. Since this run returns only whole documents, it compares unfavourably with other runs when evaluated using precision-oriented measures such as P@0.00 or P@0.01, but looks respectable in terms of P@0.10. Figure 2(a) (the red line) shows that it is among the top 4 runs for the later recall points, and it ranks 4th among 79 runs according to the MAiP measure. When blind feedback is used (the *VSMfeedback* run, represented by the cyan line in Figure 2(b)), results improve at all recall points, and the MAiP obtained is within about 5% of the best reported MAiP figure.

The element-level run *VSMfbElt* proved to be a damp squib. In hindsight, this is not surprising since during submission, our system did not consider elements at intermediate (non-leaf) levels. Leaf nodes are very often too small to contain any meaningful information; further, it is usually difficult to reliably rank such small pieces of text. After we implemented element retrieval at the non-leaf level (the *VSM-fdbk-elt* runs), results improved significantly.

However, this run was also far from satisfactory, especially at the early recall points. This suggests that the system is retrieving larger, less focused pieces of text than it should[2]. Increasing the degree of document length normalization could be one way to address this problem. The term-weighting scheme that we use – pivoted document length normalization [6] – gives us an easy way to test this hypothesis. Under this term-weighting scheme, the document length normalization factor is given by

*(1 - slope) * pivot + slope * length*

[2] Paradoxically, our document-level runs do significantly better than our element-level runs, an observation that is corroborated by other groups.

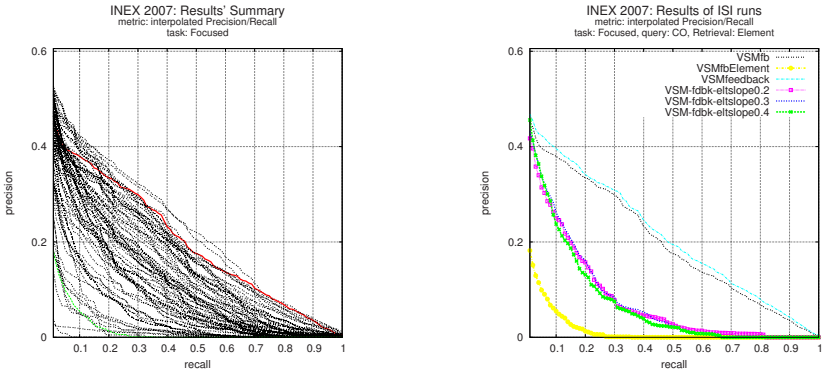

Fig. 2. Comparison of (a) all runs at INEX 07, and (b) ISI runs at INEX 07

Following [7], we had set the slope and pivot parameters to 0.20 and 80 respectively for our runs. Increasing the slope value is expected to promote elements that are shorter than the pivot length, while pushing longer elements to lower ranks. Accordingly, we experimented with two more *slope* values, viz. 0.3 and 0.4. Table 1 suggests that our intuition is correct: increasing the degree of normalization for long documents seems to improve early precision. However, the P@0.05 and P@0.10 figures reach a point of diminishing returns as the slope is increased. Also, the MAiP figures for these runs are rather dismal. More experiments are needed in order to understand the effect of normalization on precision at various recall points. We also need to explore ways to improve precision across all recall points.

4 Conclusion

This was our second year at INEX. Our main objective this year was to incorporate element-level retrieval within SMART. We started with retrieval only at the leaf level, and extended it to enable retrieval of elements at any level within the XML tree. We experimented with a 2-pass strategy where document level retrieval is done at the first pass, and the documents so retrieved are fed to the second pass for retrieval at a finer granularity. Except when considering precision at the early recall points, our baseline run was among the best, and improved further with blind feedback. However, our element-level runs were disappointing. Our intuition that short elements are victimised during retrieval seems to be validated by post-submission runs, but the effect of document length normalization on XML retrieval needs more careful study. We also hope to investigate ways to incorporate element/tag information into the term-weighting scheme. We hope these will be exciting exercises which we plan to continue in the coming years.

References

1. Fuhr, N., Lalmas, M., Trotman, A., Kamps, J. (eds.): Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007), http://inex.is.informatik.uni-duisburg.de/2007
2. W3C: XPath-XML Path Language(XPath) Version 1.0), http://www.w3.org/TR/xpath
3. Salton, G.: A Blueprint for Automatic Indexing. ACM SIGIR Forum 16(2), 22–38 (Fall 1981)
4. Buckley, C., Singhal, A., Mitra, M.: Using Query Zoning and Correlation within SMART: TREC5. In: Voorhees, E., Harman, D. (eds.) Proc. Fifth Text Retrieval Conference (TREC-5), pp. 500–238. NIST Special Publication (1997)
5. Mitra, M., Singhal, A., Buckley, C.: Improving automatic query expansion. In: SIGIR 1998, Melbourne, Australia, pp. 206–214. ACM, New York (1998)
6. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: SIGIR 1996: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 21–29. ACM Press, New York (1996)
7. Singhal, A.: Term Weighting Revisited. PhD thesis, Cornell University (1996)