The MDS Experiments for TREC5

Marcin Kaszkiel Phil Vines Ross Wilkinson Justin Zobel

Department of Computer Science, RMIT
{marcin,phil,ross,jz}@cs.rmit.edu.au

1 Introduction

The Multimedia Database Systems (MDS) group at RMIT is investigating many aspects of information retrieval of relevance to TREC. Current work includes combination of evidence, Asian-language text retrieval, passage retrieval, collection fusion, and efficient retrieval from large collections. Here we report on results from three of these strands of research.

2 Dynamic Passage Retrieval

Much of the research in text retrieval has focussed on retrieval of whole documents. However, there are many contexts in which it is preferable to consider retrieval of subparts of documents, or passages, which are potentially a better mechanism for identification of relevance than ranking of whole documents. In this section we explain our new approach to passage retrieval.

Use of passages to rank documents should be particularly effective for high-precision retrieval—because they are good at identifying documents with highly relevant parts—and for collections of longer documents where, by whole-document ranking, the relevance of a passage may be obscured by low relevance overall. In earlier experiments (in the second TREC in 1993, reported in detail later [2]) we observed exactly this behaviour. We have since revisited these experiments and results are discussed below; in these experiments with non-overlapping passages of approximately 2000 bytes (or roughly 300 words) and queries on FR, recall-precision improves from 0.243 for whole-document ranking to 0.328 with ranking based on passages.

However, our analysis of the fixed-passage approach shows that, although it can result in improvements overall, it is not foolproof. For some queries, the best effectiveness was given by use of short passages, of say 50 words, while for other queries long passages were best. Fixing passage length at database creation time is not necessarily desirable.

We therefore chose to explore a dynamic approach, in which passages are determined by queries; thus for different queries a given document could be divided in different ways. Our aim, in effect, was to explore the potential of passage retrieval and allow any piece of a document to be a passage—and to worry later about practicalities!

Given arbitrary resources this exploration of passage retrieval would proceed by, for each document, extracting the passage of every length starting at every word; then computing the similarity of every such passage to the query; then ranking documents according to their highest-ranked passage. Thus a document of 1000 words would yield approximately 500,000 passages; 10,000 words would yield 50,000,000 passages. We had to concede that this might be impractical. As an approximation we chose a set of fixed passage lengths—from 50 to 600 words in increments of 50, yielding twelve different lengths—but still allowed passages to start at every word in each document. A document of 1000 words thus yielded about 8,100 passages, 10,000 words about 116,100 passages. For each passage length we kept the highest similarity value for each document, or twelve values per document overall. (These are created on the fly, compared to every query, and discarded. A full run, for 50 queries, takes about one week on a SPARC 20.)

We then ranked documents according to their highest-ranked passages. This is not the only way to evaluate a passage-retrieval mechanism, but it fits the TREC methodology and does evaluate the effectiveness of using passages to retrieval whole documents. Nor is it a full test of our "idealistic" experiment, but it should provide good evidence of whether dynamic passages can lead to performance improvement.

In the experiments passage length was one parameter. Another was similarity measure. We used the cosine measure, as this was most successful in preliminary experiments, defined by:

$$C(q,p) = \frac{\sum_{t\in q\wedge p}(w_{q,t}\cdot w_{p,t})}{W_p}$$

where q is the query, p is the passage,

$$\begin{aligned} W_p &= \sqrt{\sum_{t\in p} w_{p,t}^2}\,, \\ w_{p,t} &= \log(f_{p,t}+1)\,, \\ w_{q,t} &= \log(f_{q,t}+1)\cdot\log(N/f_t+1)\,, \end{aligned}$$

$f_{x,t}$ is the frequency of t in x, there are N documents, and f_t is the number of distinct documents containing t. We call this *C-standard*.

When comparing passages of different length, it is clear that normalisation is problematic. For fixed-length passages, having 150 to 350 words per passage provides the best overall performance. But for the above cosine formulation passages of 50 words almost always had the highest similarity values—evidence that length normalisation is poorly formulated. We therefore investigated other approaches to normalisation. One was pivoted document length normalisation [5]. Formally it is inapplicable (it requires averages over all documents in the collection, which is not meaningful in this context) but it has been argued that the length formulation

$$W_p = (1.0 - S)\cdot L + (S\cdot U_p)$$

is reasonably robust, where $L = 300$ words is a typical passage length, the slope S is 0.2, and U_p is the number of unique terms in p. We call this *C-pivot*.

Another approach to normalisation is to standardise the similarities for each query and passage length. By, for each passage length, dividing each similarity by the highest similarity for that passage length, similarities from different passage lengths can in effect be compared according to their ranking amongst passages of the same length. (In a practical implementation this information would not be available, but should it be successful it would provide a direction in which to search for better solutions to this problem.) We call this *C-scaled*.

The full set of experiments were as follows. For baselines we applied C-standard to full documents and to passages as defined by fixed-length "pages" of 2000 bytes and the TextTiling

	Precision at N documents					Avg.	Avg. (R)
	5	10	20	30	200	prec.	prec.
C-standard:							
Documents	0.2222	0.2333	0.1595	0.1540	0.0724	0.2434	0.1775
Pages	0.2619	0.2619	0.2214	0.1905	0.0688	0.3278	0.3183
TextTiles	0.2349	0.2238	0.1929	0.1635	0.0650	0.2692	0.2529
C-pivot	0.2556	0.2238	0.1714	0.1571	0.0667	0.2852	0.2461
350 words	0.2762	0.2714	0.2405	0.2048	0.0748	0.3413	0.3364
Passages:							
C-standard	0.2508	0.2333	0.2024	0.1714	0.0712	0.2953	0.3053
C-pivot	0.2635	0.2524	0.2143	0.1921	0.0743	0.3304	0.3297
C-scaled	0.2587	0.2429	0.2262	0.1936	0.0745	0.3487	0.3554

Table 1: *Experiments with FR, disks 1 and 2, queries 51–100.*

method [1]. We applied C-standard to passages of each of the twelve lengths separately, to identify which passage length worked best. We then applied C-standard, C-pivot, and C-scaled to passages of all lengths together; these final experiments emulate dynamic passage retrieval, since in each case a passage of any length can determine the rank of a document. We did not explore combination of evidence. In all these experiments we used the full queries.

We used three data sets, focussing on FR because we would expect the greatest improvement to come from databases of long documents. The first data set was the FR data from disks 1 and 2 and the 21 queries between 51 and 100 that had at least one relevant document in FR. Results are shown in Table 1. Pages of 2000 bytes are the most effective of the previous methods, considerably improving on C-standard and C-pivot on whole documents. Slightly better was fixed-length passages of 350 words, the most effective performance observed for fixed-length passages. Dynamic passages did not significantly outperform fixed-length passages.

The second data set was the FR data from disks 2 and 4 and the 26 queries between 251 and 300 that had a relevant document in FR—these are the "blind" experiments from this year's TREC. Results are shown in Table 2. In this case passages of 150 words worked best, markedly outperforming the previous methods, the strongest of which was TextTiles. C-pivot, our submitted run, has worked best. The greatest improvement is seen in the top few documents retrieved, with precision increasing from 0.1513 for C-standard to around 0.19 for the passage methods.

The third data set was the full contents of disks 2 and 4 and queries 251–300, that is, this year's TREC run. We would not expect passage retrieval to result in large improvements on this data set, since most of the documents are fairly short; and, according to the relevance judgments, a disproportionate number of the judged documents are short (yet more evidence, perhaps, of problems with normalisation). In comparison to other retrieval techniques, we would not expect to perform especially well; C-standard is fairly effective, but we have not applied any refinements—phrase extraction and so on—that might yield increased effectiveness. Such techniques are, we believe, orthogonal to passage retrieval. However, we expected to perform at least as well as *C-standard*. Results are shown in Table 3; passages give a marginal overall improvement, but significantly improve precision for the first few documents retrieved. MDS001 was our most successful run.

	Precision at N documents					Avg.	Avg. (R)
	5	10	20	30	200	prec.	prec.
C-standard:							
Documents	0.1513	0.1423	0.1135	0.1051	0.0308	0.1534	0.1247
Pages	0.1680	0.1654	0.1192	0.1026	0.0281	0.1766	0.1677
TextTiles	0.1603	0.1577	0.1192	0.0974	0.0267	0.1817	0.1794
C-pivot	0.1859	0.1731	0.1212	0.0897	0.0287	0.1755	0.1459
150 words	0.1936	0.1731	0.1346	0.1115	0.0281	0.2416	0.2133
350 words	0.1923	0.1731	0.1288	0.0987	0.0275	0.2219	0.2130
Passages:							
C-standard	0.1949	0.1846	0.1385	0.1039	0.0288	0.2002	0.1640
C-pivot	0.1872	0.1885	0.1365	0.1077	0.0277	0.2268	0.2151
C-scaled	0.1821	0.1654	0.1327	0.1103	0.0308	0.1984	0.1463

Table 2: *Experiments with FR, disks 2 and 4, queries 251–300. C-pivot was the FR component of the run submitted as MDS001.*

	Precision at N documents					Avg.	Avg. (R)
	5	10	20	30	200	prec.	prec.
C-standard:							
Documents	0.3480	0.3480	0.3090	0.2680	0.1288	0.1798	0.2199
Passages:							
C-pivot	0.3707	0.3608	0.3050	0.2760	0.1221	0.1804	0.2292

Table 3: *TREC 5 experiments, disks 2 and 4, queries 251–300. C-pivot was the run submitted as MDS001.*

3 Combination of Evidence

Term expansion is a popular and successful method to improve retrieval performance. There are many strategies employed, however many can be related to Rocchio's formula for relevance feedback[4]:

> α(Original Query Vector) + β(Average of Relevant Document Vectors) - γ(Average of Irrelevant Document Vectors)

Each of α, β, and γ are non-negative real numbers. Typical values might be $\alpha = 2$, $\beta = 1$ and $\gamma = 0$. Rather than using relevance assessments, it is often assumed that the top N documents are relevant, and these documents only are used in term expansion. Instead of regarding this information as a set of vectors to average, we may regard terms from a query, and terms from documents as two different sorts of evidence – evidence that can be combined.

As with our experiments in TREC4, we started with retrieval runs based on a cosine measure and a measure derived from the Okapi experiments. For the cosine we used:

$$COS(Q, D) = \frac{\sum_{t \in q \wedge d} (w_{q,t} \cdot w_{d,t})}{\sqrt{\left(\sum_{t \in q} w_{q,t}^2 \cdot \sum_{t \in d} w_{d,t}^2\right)}}$$

with $w_{q,t} = log(N/f_t) + 1$ and $w_{d,t} = log(f_{d,t} + 1)$. For Okapi we used:

$$OKA(Q, D) = \left(\frac{N - f_t}{f_t}\right)\left(\frac{f_{d,t}}{f_{d,t} + \sqrt{f_D}/av(\sqrt{f_D})}\right)$$

where $f_{x,t}$ is the frequency of t in x, and f_D is the length of D. N is the number of documents in the collection, and f_t is the number of documents containing t.

Again we were interested in only very short queries. Thus we used the description field only. After stopping and stemming, there were on average 8 terms per query.

The top 15 documents for each query were obtained using the COS measure. The text of these documents was agglomerated and the 45 best terms were extracted according to the formula:

$$f_{15d,t}/log(f_t + C)$$

where $f_{15d,t}$ is the frequency of t in the top 15 documents, and C = 20, a constant to ensure only statistically significant variation was taken into account. We call this new query, Q'. Using these new queries, without using the original terms unless they were in the top 45 terms, a ranking was obtained using the COS measure given above.

The results of the two cosine measures were combined. The combination was based on a sum of angles. The method used was to weight the sum so that the original measure was given four times as much weight as the expanded query. This is a rough reflection of the weights used by others when applying, say, the Rocchio measure. If we let $COS1 = COS(Q, D)$, the cosine of the angle between the description field and the document, and let $COS2 = COS(Q', D)$, the cosine of the angle between the expansion terms and the document, the formula is:

$$Comb1(Q, D) = \frac{(COS1 + \alpha COS2)}{((COS1 + \alpha COS2)^2 + (((1 - COS1^2) + \alpha(1 - COS2^2)^2)^{0.5})^{0.5}}$$

The geometric interpretation is that this value is the cosine of the angle between the document vector and a vector obtained by adding the normalized original query vector, and α times the normalized derived query vector. We set $\alpha = 0.25$. We now had a result based on a short original query and and a derived expanded query based on the database.

In TREC5 we had found good results for short queries by combining results obtained from cosine based measures and Okapi based measures. However, we no longer have two angles to combine, nor are values actually comparable, so we had first to normalize the Okapi results. Thus for each query, we multiplied the Okapi based similarity measure by the top score for the combined cosine measure and divided by top Okapi score. This guaranteed that the top score for both measures was the same. Thus $OKA2(Q, D) = \beta_Q OKA(Q, D)$.

Because more work had been put into developing a high quality result for the cosine, compared to the Okapi measure, we gave twice the weight to the cosine combined measure compared to the Okapi measure. Even though the Okapi measure was not a cosine, it could be treated as such so we used the same combination method as before. Thus we calculated:

$$\frac{(Comb1 + \gamma OKA2)}{((Comb1 + \gamma OKA2)^2 + (((1 - Comb1^2) + \gamma(1 - OKA2^2)^2)^{0.5})^{0.5}}$$

Here $\gamma = 0.5$. This run was submitted as MDS002.

In our manual runs, we simply added more evidence! We took the narrative field, and manually extracted what appeared to be low content phrases. If we had a sophisticated

	Precision at N documents					Avg.	Avg. (R)
	5	10	20	30	200	prec.	prec.
Desc	0.2440	0.2200	0.1850	0.1660	0.0878	0.1027	0.1397
Expand	0.2520	0.2140	0.1900	0.1807	0.0972	0.1168	0.1515
Comb1	0.2480	0.2380	0.2010	0.1907	0.1059	0.1229	0.1624
Desc(Oka)	0.1840	0.1640	0.1520	0.1393	0.0751	0.0755	0.1105
MDS002	0.2453	0.2320	0.2160	0.2013	0.1044	0.1214	0.1606
Narr	0.3480	0.2960	0.2850	0.2533	0.1193	0.1654	0.1938
Comb2	0.3800	0.3240	0.2750	0.2440	0.1184	0.1526	0.1957
Comb3	0.3240	0.2960	0.2730	0.2513	0.1233	0.1558	0.2016
MDS003	0.3060	0.3060	0.2780	0.2480	0.1187	0.1480	0.1875

Table 4: *TREC 5 combination experiments.*

phrase deletion algorithm, this would again be an automatic run. We then applied standard stopping and stemming to the narratives to have a remaining set of approximately 26 words per query, approximately 4 times larger that we had using the descriptions only. The strategy of combination was as follows:

1. Combine original description query with manually altered narrative query, $\alpha = 1.0$ (Row Comb2 in Table 4.)
2. Combine result with COS2, the result from modified queries obtained earlier, $\alpha = 0.25$ (Row Comb3 in Table 4.)
3. Combine result with OKA2, the normalized Okapi results, $\alpha = 0.5$

This result was submitted as MDS003.

For all runs, only the top 1,000 documents were used for processing, combining, or evaluating. There may be some benefit to using more documents in the base runs, but we expect that the effect would be extremely marginal. The results of these experiments are given in Table 4.

4 Chinese Retrieval

In the Chinese language each character represents a syllable. "Words" usually consist of one, two, or three syllables. Our past research [3] has concentrated on looking at whether individual characters or complete words are more effective as the basic indexing and retrieval units for IR. Complete words usually have a much more specific meaning than individual characters. For example when the character "dian" (electric) combines with "nao" (brain) the resultant word is "computer". Consequently we had initially expected that word based retrieval would be more effective than character based.

Our experiments were run using *mg*, which has been used extensively for English language information retrieval. Because *mg* is not set up to handle 16 bit characters, we pre-processed the documents and queries to turn the text into ASCII strings. Each Chinese character was converted to a four character ASCII string, using a hexadecimal representation. Thus the sixteen bit code $\mathrm{a01c01c}_{16}$ was converted to the character string "a01c". This was the only pre-processing done for character retrieval. Indexing and retrieval was then done using both the cosine and Okapi similarity measures. (Cosine-char and Okapi-char in Table 5.)

In Chinese there is no white space between words such as occurs in English. Consequently character sequences must first be *segmented* into words before word based indexing can be performed. In the past successful segmentation has been thought to be a significant impediment to the indexing and hence retrieval of Chinese text [7]. The segmentation problem in Chinese has been extensively researched, both in relation to information retrieval [7] and for other applications, such as speech recognition. No algorithm has been developed that has achieved perfect results. Indeed, much Chinese humour derives from the fact that some phrases may be legitimately segmented in different ways, giving rise to completely different meanings. The best segmentation schemes yield about 96% accuracy [6].

We used a dictionary based parsing method to segment the text. Dictionary methods may work by finding the longest substring in a sequence of text which is contained in a lookup dictionary. Such an approach is known as greedy parsing. We have used this method in previous work, with satisfactory results. Once again indexing and retrieval was then done using both the cosine and Okapi similarity measures. (Cosine-word and Okapi-word in Table 5.)

In the set of combination experiments, all runs were normalized so that the maximum score for one run for a particular query was made equal to the maximum score for the other run for that query. As well, a more common weighted sum was used:

$$\frac{W1 + \alpha W2}{(1+\alpha)}$$

We combined as follows:

1. Combine character based cosine run with character based Okapi run, $\alpha = 0.5$

This result was submitted as MDS004.

1. Combine word based cosine run with word based Okapi run, $\alpha = 0.5$
2. Combine character based combined run with word based combined run, $\alpha = 0.5$

This result was submitted as MDS005.

The results are shown in Table 5. The performance is awful. After a lot of investigation we have still not found the bug in our system.

References

[1] M.A. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In R. Korfhage, E. Rasmussen, and P. Willett, editors, *Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval*, pages 59–68, Pittsburg, U.S.A., June 27 – July 1 1993. ACM.

[2] A. Moffat, R. Sacks-Davis, R. Wilkinson, and J. Zobel. Retrieval of partial documents. Technical report, Key Centre for Knowledge Based Systems, Departments of Computer Science, RMIT and the University of Melbourne, Melbourne, Australia, 1993.

	Precision at N documents					Avg.	Avg. (R)
	5	10	20	30	200	prec.	prec.
Cosine-char	0.0842	0.1316	0.1158	0.1053	0.0539	0.0369	0.1021
Okapi-char	0.0105	0.0105	0.0184	0.0158	0.0116	0.0043	0.0213
Cosine-word	0.1368	0.1316	0.1237	0.1053	0.0661	0.0408	0.1066
Okapi-word	0.0842	0.1053	0.0921	0.0842	0.0497	0.0251	0.0798
Combined-char	0.0947	0.1158	0.0921	0.0930	0.0476	0.0268	0.0867
Combined-word	0.1368	0.1474	0.1211	0.1070	0.0655	0.0406	0.1068
Combined-all	0.1474	0.1211	0.1237	0.1123	0.0558	0.0371	0.1045

Table 5: *Chinese experiments. Combined-char was submitted as MDS004, Combined-all was submitted as MDS005.*

[3] V. B. H. Nguyen, P. Vines, and R. Wilkinson. A comparison of morpheme and word based document retrieval for asian languages. In *International Conference on Database and Expert System Applications — DEXA 96*, page To Appear, Zurich, Switzerland, 1996.

[4] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART RETRIEVAL SYSTEM*, pages 243–264. Prentice Hall, New Jersey, 1966.

[5] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H.-P. Frei, D. Harman, P. Schauble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval*, pages 21–29, Zurich, Switzerland, August 18–22 1996. ACM.

[6] R. Sproat and C.L. Shih. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, 4(4):336–351, 1990.

[7] Z. Wu and G. Tseng. Chinese text segmentation for text retrieval: Achievements and problems. *Journal of the American Society for Information Science*, pages 532–542, October 1993.