Microsoft Cambridge at TREC–10: Filtering and web tracks

S E Robertson* S Walker† H Zaragoza‡

1 Summary

This report is concerned with the Adaptive Filtering and Web tracks. There are separate reports in this volume [1, 2] on the Microsoft Research Redmond participation in QA track and the Microsoft Research Beijing participation in the Web track..

Two runs were submitted for the Adaptive Filtering track, on the adaptive filtering task only (two optimisation measures), and several runs for the Web track, both tasks (adhoc and home page finding). The filtering system is somewhat similar to the one used for TREC–9; the web system is a simple Okapi system without blind feedback, but the document indexing includes anchor text from incoming links.

2 Okapi at TRECs 1–9

A summary of the contributions to TRECs 1–7 by the Okapi team, first at City University London and then at Microsoft, is presented in [5]. In TRECs 7–9 we took part in the adaptive filtering track, initially concentrating on the thresholding problem, but by TREC–9 we had a full adaptive filtering system with query expansion as well as adaptive thresholding. This adaptation could be used to optimise performance on a number of effectiveness measures, although with one limitation discussed below, and produced good results on both the TREC–9 measures, linear utility and the 'precision-oriented' measure. In earlier TRECs on various adhoc tasks we had concentrated on the weighting schemes and pseudo relevance feedback (blind feedback), and had developed the successful BM25 weighting function but had had only limited success with blind feedback.

3 The system

At the Microsoft Research laboratory in Cambridge, we are developing an evaluation environment for a wide range of information retrieval experiments. This environment is called Keenbow. The Okapi BSS is seen as a component of Keenbow. Many aspects of the system, including the weighting scheme and the query expansion methods used, reflect the various components of the probabilistic model of retrieval discussed at length in [7].

The Okapi Basic Search System (BSS), which has been used in all Okapi and Okapi/Keenbow TREC experiments, is a set-oriented ranked output system designed primarily for probabilistic–type retrieval of textual material using inverted indexes. There is a family of built-in weighting functions collectively known as BM25, as described in [6, Section 3] and subsequent TREC papers. In addition to weighting and ranking facilities it has the usual boolean and quasi-boolean (positional) operations and a number of non-standard set operations. Indexes are of a fairly conventional inverted type. There have been no major changes to the BSS during TREC–10.

Query expansion or modification methods using known or assumed relevant documents are generally built as scripts on top of the BSS. The range of methods used is described in our TREC–9 report [3]. A characteristic of the methods is that terms are selected for the modified query by a 'term selection value'.

The filtering system is again built as scripts on top of the BSS and lower-level scripts such as for query expansion. The incoming 'stream' of documents is divided fairly arbitrarily into batches. For each topic a current state is maintained, including query formulation, threshold etc., what happened at the last batch, and some history, including docids for any documents judged relevant up to now. As a new batch of documents is processed, the current query formulation of each topic is searched against it; cumulative databases are created, and each topic goes through the adaptation process in preparation for the next batch. Adaptation includes threshold adaptation as well as query modification.

3.1 Hardware

All the TREC–10 processing was done at Microsoft Research, Cambridge. Most of the work was done on a 550MHz Xeon (512KB Cache) with 2Gb RAM and a Dell with two 400 MHz Pentium processors and 512 Mb. Both machines were running Solaris 7. The network was 100Mbps ethernet.

*Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK, and City University, London, UK. email ser@microsoft.com

†Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK. email sw@microsoft.com

‡Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK. email hugoz@microsoft.com

4 Web track

4.1 Methods

We wanted to investigate the base performance of the Okapi BSS system when used with documents and queries originating from the WWW as opposed to other more homogenous collections. For the ad-hoc task, we experimented with word phrases (adjacent word pairs) and query expansion from blind feedback. For the Home-Page finding task we experimented with anchor text as well word phrases.

The methods used to select and weight phrases is the same one introduced in [8]. All adjacent word pairs are given a *plausibility* score and sorted, retaining word pairs above a certain threshold. This threshold was set to its usual value of 10. Word pairs were weighted by the weighs of its individual terms plus a small value constant across all terms (using a term-dependant weight yields similar results [8]).

The query expansion method used is also the same as in [8]. It was only used in one run, ad-hoc retrieval using all query fields (wtnd1). The top 15 documents were considered relevant for query expansion, and 20 terms were retained (forcing original query terms to be selected). BM25's constant k3, set to 0 for all runs, is increased after query expansion.

No HTML information was used, except for the extraction of anchor text. This was done using several heuristics to speed up the pre- processing of the data and reduce the size of the resulting index. First of all, the outlink records were used to index the relevant anchor text contained in every document of the collection. Then, using the inlink records, each document was augmented with the anchor text found in the links of other documents pointing to it. We limited the number of inlinks for each document, allowing at most 5 (chosen randomly when there were more available). Furthermore, a matching heuristic was used to parse relative addresses; full URL resolution was carried out only on potential links, greatly decreasing the pre-processing time (this also introduced some errors, estimated at less than 0.1%).

4.2 Web track results

Table 1 summarises the results obtained for the ad-hoc retrieval task. The use of 2-word phrases improved average precision slightly, while query expansion was clearly beneficial.

Table 2 summarises the results obtained for the home page finding task. The use of 2 word phrases lead to a slight increase in performance. The use of anchor text yielded more significant gains of performance, while decreasing the number of home pages not found by 2%. However, overall results are not impressive; using the Okapi system *off-the-shelf* for the task of home page finding is not ideal (in [2] different techniques are used to adapt the Okapi system to this task).

5 Adaptive Filtering

5.1 System and parameters

Last year's report [3] contains a fairly detailed account of the filtering system and the adaptation methods used, in particular the relation between the optimisation measure and the threshold. This year's system is very similar to last year's; Table 3 is an attempt to summarise the large number of parameters used.

5.2 Optimising Fbeta

The new Fbeta measure measure presents an interesting problem for optimisation. For last year's target-based measure, we used a fairly general optimising method, which involved estimating the value of the measure at different cut-off points in the collection-so-far, and then extrapolating to the estimated size of the final collection. In one respect this procedure is simpler for Fbeta, but in another it is more complex.

The simplicity arises from the fact that Fbeta is independent of absolute numbers of documents, in the sense that if every quantity (document count, such as total retrieved or total relevant) is doubled, Fbeta does not change. Thus the simplest estimate of Fbeta at a given score cutoff in the whole final collection is just the actual or estimated Fbeta at that score in the collection-so-far. This estimate assumes that the proportion of relevant documents does not change – in general this is the kind of assumption that is made in adaptive systems, although it might be possible for example to take account of a trend to get a better estimate.

The complexity arises from the fact that Fbeta depends on recall, which in turn depends on the total number of relevant documents in the collection. Unfortunately we do not usually know this even for the collection-so-far, let alone for the full final collection.

The methods that we have used in setting thresholds do give us some kind of handle on this question. As discussed in previous TREC reports and in [4], we calibrate the score to give an explicit probability-of-relevance estimate for each document. If this calibration is accurate, then (given a query and a collection) that the total number of relevant documents in the collection is:

$$\sum_{\text{all docs}} p_{\text{doc}}$$

where p_{doc} is the calibrated probability of relevance of the document. While this would be quite a heavy calculation, we can select those documents with highest probabilities

Table 1: Ad-hoc task (WEB track).

Run	Avg. Precision	Run Description	Used for evaluation pool
ok10wt1	0.1908	title only	Yes
ok10wt3	0.1952	title only, word-pairs	Yes
ok10wtnd0	0.2512	title and description	No
ok10wtnd1	0.2831	title and description, query expansion	No

Table 2: Home page finding task (WEB track).

Run	Avg. Reciprocal Rank	Top 10	Not Found	Run Description
ok10whd0	0.312	58.6%	15.2%	plain
ok10whd1	0.340	60.7 %	15.9%	word-pairs
ok10wahd0	0.362	62.1%	13.1%	anchor text
ok10wahd1	0.387	64.1%	13.1%	word-pairs + anchor text

of relevance by simply retrieving the top-scoring documents in the usual fashion, and then assume that the remainder will contribute little to the total. Given that we know that many of the Reuters topics have substantial numbers of relevant documents, it is appropriate nevertheless to retrieve a substantial number for this purpose. The number chosen for the present experiments was 10% of the collection. The results below contain some comments on this limit.

It should be noted that the calibration of the score is intended primarily to provide accurate estimates of the probability of relevance of documents scoring around the chosen threshold. It is very likely that those much further down the ranking are very much less accurate. (More specifically, the calibration assumes a linear model on the log-odds scale, and adapts the intercept but not the slope, to fit the documents of known relevance, which are those that have already been retrieved and are therefore high up the ranking. It is likely that the slope is critical for accurate estimation of the probabilities lower down the ranking.) Thus we should not expect the estimated total relevant to be very accurate either. Some experiments to determine the accuracy of these estimates are described below.

Given an estimate of the total number of relevant documents, we can estimate Fbeta for any score threshold by discovering the number retrieved at that threshold and estimating the number relevant. We can then choose the score threshold which maximises Fbeta. Because of the assumed independence of Fbeta from the absolute number of documents, we perform this procedure for each topic on the collection-so-far, and the resulting threshold is a candidate for applying to the next batch.

As with the utility measure, we have to start the process off. We do this (again as with utility) by starting with a target based simply on the number of documents to be retrieved over the period. Once we have retrieved a few relevant documents, we can consider an Fbeta-based threshold. However, there remains a danger that an early and inaccurate Fbeta-based threshold will be too stringent and will not retrieve any more documents. We therefore push up the target-based threshold gradually, using the ladder method discussed in previous TREC reports, until it is higher than the Fbeta-based threshold.

5.3 The accumulated collection

As in previous years, we assume that we accumulate the documents as they come into the system, so that we always have a cumulated collection of everything received up to now. Such a collection is needed for some of the forms of adaptation discussed; in TREC–10 (as in TREC–9), we actually need two such collections, respectively including and excluding the training set.

5.4 Overview of the filtering procedure

At a particular iteration of the process, any query modification needs to take place before any threshold setting. It may also be necessary, after query reformulation but before threshold setting, to recalculate the scores of the previously-retrieved documents, for the adaptation of β.

The Reuters collection is batched by day (as it comes on the CD), with batch 0 as the training set. However, for efficiency reasons several days' documents may be searched without the adaptive procedures being initiated. The adaptive procedures were initiated every day for the first 10 days, every three days up to 40 days, and every 10 days thereafter.

For similar reasons, query modification is done after any batch in which a new checkpoint is reached for the particular topic. In these experiments, the checkpoints are defined in terms of the number of relevant documents retrieved so far, and are set at 1,2,4,8,16... relevant documents.

So the basic procedure is as follows: for each batch (day) i of incoming documents

Table 3: Parameters for adaptive filtering

See notes below and [3] for explanations of these parameters	
BM25 parameters:	
k_1	1.2
b	0.8
Score calibration:	
Initial beta	-0.68
Mythical reldocs for beta re-calibration	3
Gamma	2.7
Threshold adaptation:	
Initial target no. of documents	100
Ladder step	0.2
Query modification:	
Reldocs used for modification	100
Maximum terms	25
Minimum terms	3
Absolute term selection value threshold	5

Notes
For both the Utility and Fbeta measures, the initial threshold is set to retrieve a specified target number of documents over the whole period of the test set.
For Utility, the threshold calibrated as a log-odds probability is raised by one ladder-step for each relevant document retrieved, until it reaches the level defined by the utility function.
For Fbeta, again the threshold is raised by one ladder-step for each relevant document. However, after 4 relevant documents have been retrieved, the ladder threshold is compared with the estimated optimum Fbeta threshold, and the lower of the two is chosen.
Query modification is based on the original text query, the training sample and the reldocs retrieved so far. If there are more reldocs than the limit, only the most recent are taken.
Terms are ranked by absolute term selection value (new offer weight). All those exceeding the threshold are chosen, subject to both a minimum and a maximum number of terms.

1. Run current profiles against batch i
2. Update both cumulative databases (batches 0–i and batches 1–i)
3. If adapt today, then for each topic:
   (a) if new checkpoint has been reached,
   - reformulate query
   - recalculate scores of previously retrieved documents and re-estimate β (needed for score calibration)

   (b) set threshold (using methods described above)

5.5 Filtering results

As with the official track results, the measures reported are T10SU (scaled utility), T10F (van Rijsbergen's Fbeta measure with beta=0.5), set precision and set recall.

Submitted runs

Just two runs were submitted, one optimised for each of the two main measures, T10SU and T10F. The results are shown in Table 4.

These results are somewhat disappointing. The most obvious problem is that the run optimised for T10F actually does better on T10SU than the run optimised for T10SU. Some examination of the results suggests that in quite a number of topics, the profile, having retrieved a small number of poor documents initially, gets locked into a state with a high threshold, and fails to retrieve any other documents. Although this is exactly one of the conditions that the procedures were designed to avoid, it seems that they have failed on several occasions. The problem is exacerbated by the fact that many of these topics actually have very large numbers of relevant documents – thus once out of this trap, the optimisation could be expected to do very well.

The T10F optimisation appears to fall into this trap less often. A pragmatic response might be to start the utility runs off with an Fbeta target rather than a document number target, and only switch to the utility target when a reasonable number have been retrieved. However, it is clear that some rethinking is required.

Table 4: Official results for submitted runs

Run	Optimisation measure	T10SU	T10F	Set Precision	Set Recall
ok10f2br	T10F	0.137	0.330	0.427	0.273
ok10f2ur	T10SU	0.104	0.254	0.368	0.214

Estimating the number of relevant documents

Some experiments were done with the TREC-9 filtering collection (based on OHSUMED) and the Fbeta measure in order to assess the estimation of the total number of relevant documents in the collection. We first look at the estimate given towards the end of the process, that is for the next-to-last batch. Note that the system attempts to estimate the total relevant only when it has retrieved some relevant documents; initially, when it is using the target method or the ladder, no estimate is required or made. In the OHSUMED collection with the OHSU queries, a number of topics never reach that stage. Table 5 shows the estimates from the next-to-last batch and the first ten topics, compared with the actual total number of relevant in the collection, for a typical run (one of many trial runs).

Table 5: Estimating total relevant – OHSU

Topic	Estimated R	Actual R
1	46.6	44
2	23.2	44
3	168.9	165
4	(no estimate)	12
5	139.8	44
6	(no estimate)	27
7	7.9	19
8	(no estimate)	11
9	50.1	44
10	(no estimate)	19

These estimates were moderately promising but not wonderful – most are in the right ball-park, but topics 5 and 7 are fairly poor. The corresponding estimates for the TREC–10 Reuters data, the official ok10f2br run, are as in Table 6

Here the obvious problem topic is number 2, where the estimate is out by a factor of approximately 200. This is an instance of the problem mentioned above, where the profile, after retrieving a very small number of documents at the beginning, got stuck with a too-high threshold and failed to retrieve anything else. The others, while being within an order of magnitude of the correct figure, nevertheless may diverge from it by a factor of up to 4.

If we look at the progress of the estimates over the time-period of the simulation, it is clear that they may be more wildly out near the beginning. Visual inspection

Table 6: Estimating total relevant – Reuters

Topic	Estimated R	Actual R
1	10,342	23,651
2	59	11,563
3	9,041	36,463
4	12,109	7,250
5	26,973	22,813
6	3,316	1,871
7	18,388	17,876
8	27,871	11,202
9	9,180	2,560
10	17,165	5,625

of the OHSU data suggested that the early estimates were often (not always) much too high. However, they tended to move into the right order of magnitude fairly quickly.

There are a number of variables whose effects on the estimates have not been investigated. One of these is the cut-off number of documents used for the estimate (see section 5.2 above. It appears (impression only) that we may get better estimates by limiting this number further. This does suggest that the linear calibration model, with fixed slope, is not working very well further down the ranking.

This observation is interesting, because it gives us another way of testing the calibration. It seems clear that, for this purpose at least, we need to adjust the slope as well as the intercept. However, it is not at all clear how this may be done.

5.6 Computational load

We commented last year that with about 5000 MeSH topics the OHSU filtering task was computationally very heavy. Even with less than 100 topics, the same comment can be made about the Reuters filtering task. Partly this is because many of the Reuters topics have very large numbers of relevant documents, and therefore retrieved sets tend also to get large. But the general point remains: adaptive filtering is computationally very much more demanding than adhoc searching.

6 Conclusions

The web track presents some interesting challenges. The use of anchor text of incoming links as a way of providing

additional information about a page clearly has potential, but we have only scratched the surface in the present experiments. The tiny benefit from phrases (word pairs) and the somewhat greater benefit from blind feedback are both consistent with previous experiments.

The adaptive filtering task continues to be an interesting and fruitful one to investigate. The new Fbeta optimisation measure has been interesting, particularly because of the need to estimate the total number of relevant documents in the collection. Clearly further work in this area is required. Also, despite our attempts of the last three years to devise a robust system for setting and adapting thresholds, the Reuters collection has presented conditions which under some circumstances cause our methods to fail.

References

[1] Brill, E. and Dumais, S. [MSR QA track.] In this notebook

[2] Gao, Jianfeng et al. TREC-10 Web Track Experiments at MSRCN. In this notebook

[3] Robertson, S.E. and Walker, S. Microsoft Cambridge at TREC-9: Filtering track. In: [9], p361–xxx.

[4] Robertson, S.E. and Walker, S. Threshold setting in adaptive filtering. *Journal of Documentation*, 56, 2000 p312–331.

[5] Robertson, S.E. and Walker, S. Okapi/Keenbow at TREC-8. In: [10], p151–162.

[6] Robertson, S.E. *et al.* Okapi at TREC–3. In: [12], p109–126.

[7] Sparck Jones, K., Walker, S. and Robertson, S.E. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36, 2000, p779–808 (Part 1) and 809–840 (Part 2).

[8] Beaulieu, M.M. *et al.* Okapi at TREC–5. In: [11], p143–165.

[9] *The Ninth Text REtrieval Conference (TREC–9).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 2001 (NIST Special Publication 500-249).

[10] *The Eighth Text REtrieval Conference (TREC–8).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 2000 (NIST Special Publication 500-246).

[11] *The Fifth Text REtrieval Conference (TREC–5).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1997.

[12] *Overview of the Third Text REtrieval Conference (TREC–3).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995 (NIST Special Publication 500-225).