Microsoft Cambridge at TREC–9: Filtering track

S E Robertson* S Walker†

1 Summary

Apart from a short description of our Query Track contribution, this report is concerned with the Adaptive Filtering track only. There is a separate report in this volume [1] on the Microsoft Research Cambridge participation in QA track.

A number of runs were submitted for the Adaptive Filtering track, on all tasks (adaptive filtering, batch filtering and routing; three separate query sets; two evaluation measures). The filtering system is somewhat more advanced than the one used for TREC–8, and includes query modification and a more highly developed scheme for threshold adaptation. A number of diagnostic runs are also reported here.

2 Okapi at TRECs 1–8

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 [7]. Here we discuss only the routing and filtering task submissions.

Over the course of TRECs 1–6, we developed iterative methods of optimising the routing queries which were very successful, though computationally heavy. In successive TRECs our methods were enabled to explore more of the potentially huge space of possible queries. In TRECs 5 and 6 we used the same methods for batch filtering, again successfully.

However, we put these iterative methods aside for the adaptive filtering task in TREC–7. Here and in TREC–8 we concentrated on developing thresholding techniques, and did not in fact modify initial queries at all. This approach was relatively successful in TREC–7, but by TREC–8 many participants had better methods which additionally expanded or modified the queries adaptively, and we were somewhat left behind.

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 [8].

3.1 The Okapi Basic Search System (BSS)

The 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–9.

3.2 Query expansion/modification

Given some known relevant documents, the query may be modified (primarily by adding new terms, but weights may be adjusted and an ineffective query term might also be dropped).

The initial step is to choose terms. Prior to TREC–8 the method used was that proposed in [9] by which terms are ranked in decreasing order of a term selection value or offer weight:

$$TSV = r.w^{(1)} \tag{1}$$

(where $w^{(1)}$ is the Robertson/Sparck Jones weight [5], a component of BM25, and r is the number of (known) relevant documents in which the term occurs). The top t ranked terms are then chosen. For TREC–8 a new method was developed. This is based on a significance argument, and thus allows an absolute threshold on the offer weight, which may select different numbers of terms under different conditions. The formula is discussed in [7], and is as follows:

$$NTSV = r \log \frac{N}{n} - \log \begin{pmatrix} R \\ r \end{pmatrix} - \log V \tag{2}$$

*Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK, and City University, London, UK. email ser@microsoft.com

†Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK. email sw@microsoft.com

where r is as above; R is the total number of (known) relevant documents; n is the number of documents in the collection which contain the term; N is the size of the collection; V is the size of the vocabulary (number of distinct terms). We may use an absolute threshold criterion with this new offer weight:

$$NTSV > c \qquad (3)$$

An argument was presented last year that zero would be a suitable value for c.[1]

The basic approach to query reformulation may now be described as follows:

1. extract all terms from all documents judged or assumed to be relevant
2. rank all terms, including original query terms, in order of offer weight
3. select those terms above a threshold or cut-off, defined as a threshold on the offer weight and/or a cut-off on the number of terms
4. weight the terms according to the usual relevance weighting formula (not the same as the offer weight)

Either or both the offer weight and the relevance weight may include some bias towards query terms; thus original query terms may remain in the query even if they occur in no or few relevant documents so far. However, the bias is not normally absolute: a query term which continues not to appear in relevant documents will eventually be dropped.

The above methods might be termed "model-based", and do not cover the iterative optimisation methods used in the routing task in earlier TRECs.

3.3 Filtering system

The filtering system used from TREC–7 on consists mainly of scripts built on top of the BSS.

The incoming "stream" of documents is divided fairly arbitrarily into batches (smaller batches initially to allow fast learning; larger later for efficiency reasons). 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 query modification (term selection and weighting) and threshold adaptation; the various components are described below.

[1] The scale of this offer weight is $(-\infty, +\infty)$; a threshold of zero implies that we would expect about 1 noise term to be selected. We have discovered a bug in last year's programs, which means that last year's offer weights were offset by a certain amount; a correct zero threshold today is equivalent to a small negative threshold last year.

3.4 Hardware

All the TREC–9 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.

Table 1: Query track runs on Okapi

Query set	AveP	P@5	RPrec	Recall
acs1a	0.261	0.544	0.305	0.529
Sab1c	0.261	0.528	0.306	0.544
Titles	0.259	0.500	0.298	0.516
Sab1b	0.255	0.560	0.296	0.530
UoM2	0.254	0.564	0.305	0.573
pir1a	0.252	0.584	0.302	0.541
Sab1d	0.252	0.568	0.296	0.514
INQ1f	0.246	0.488	0.289	0.503
Sab1a	0.242	0.572	0.291	0.514
Sab2a	0.242	0.548	0.293	0.535
INQ1c	0.240	0.516	0.290	0.518
UoM1a	0.232	0.516	0.284	0.484
INQ2e	0.224	0.508	0.276	0.475
INQ1e	0.224	0.436	0.259	0.446
INQ2c	0.223	0.516	0.278	0.493
Sab3a	0.221	0.536	0.276	0.504
INQ1i	0.219	0.464	0.257	0.496
INQ1b	0.217	0.488	0.269	0.498
INQ1j	0.216	0.500	0.264	0.470
UoM1b	0.215	0.516	0.263	0.478
INQ1g	0.213	0.520	0.268	0.475
INQ1h	0.199	0.460	0.246	0.498
INQ1d	0.197	0.452	0.245	0.490
INQ2f	0.196	0.460	0.256	0.485
INQ2d	0.185	0.444	0.243	0.474
INQ1a	0.185	0.420	0.228	0.449
APL1a	0.182	0.432	0.233	0.433
INQ2g	0.180	0.428	0.242	0.423
INQ3e	0.175	0.432	0.214	0.440
APL2a	0.171	0.344	0.231	0.436
INQ2i	0.166	0.436	0.223	0.456
INQ2b	0.165	0.392	0.227	0.413
INQ2h	0.165	0.380	0.217	0.428
INQ2j	0.149	0.340	0.196	0.415
INQ3d	0.147	0.372	0.204	0.381
INQ3j	0.144	0.340	0.206	0.383
INQ3f	0.135	0.348	0.190	0.384
INQ2a	0.132	0.348	0.192	0.365
INQ3i	0.120	0.316	0.179	0.370
INQ3c	0.116	0.300	0.170	0.333
INQ3g	0.116	0.292	0.171	0.328
INQ3b	0.107	0.312	0.152	0.304
INQ3a	0.106	0.264	0.162	0.310
INQ3h	0.096	0.276	0.154	0.324

4 Query track

We did not take full part in the query track: that is, we did not generate queries. We did however run the queries that other participants had generated.

The system used to run these queries was an absolutely standard Okapi system, parsing the queries as provided in a standard manner and using BM25 weighting with $k_1 = 0.8$, $b = 0.4$, and $k_3 = 0$. No expansion was used. Some results for different query sets are shown in table 1, together with a corresponding run on topic titles only, sorted by average precision. Only two of the query sets outperformed topic titles on average precision, although several of them do better on other measures, particularly precision at 5 documents.

5 Filtering and routing

5.1 System design

For the last two years, the Keenbow/Okapi team has concentrated on the setting of thresholds for the adaptive filtering task. This year's effort is a much more rounded one, bringing together the thresholding methods and previously developed methods of query expansion and reweighting. At the same time, the introduction of the new target and measure into the adaptive filtering task has stimulated a significant expansion of the thresholding ideas, in a way which complements the previous approaches.

5.2 T9P thresholding: basic ideas

In the precision-oriented task, we have to attempt to retrieve the best 50 documents over the simulated life of the profile. The primary requirement is to set the threshold so as to retrieve close to that number of documents over the period (adjusting it as we go as necessary), while relying on the query to get us as close as possible to the best 50 documents.

Given a profile, some history of the stream of documents, and an expected rate of incoming new documents, we can relate the threshold to the number of documents in a model-free fashion, thus: we run the query against the accumulated collection so far, and rank the documents in the usual way; then the future number of documents whose score will reach a given threshold may be estimated from the number retrieved in the past at that threshold, adjusted pro-rata.

Such an estimate may not be very good, and will need adapting. So the principle is that after every batch of documents, we do a new retrospective search of the accumulated collection so far, and choose the threshold which is estimated to give us the right number of documents in the future, given what we have retrieved in the past. Since the evaluation measure penalizes under-retrieval more than over-retrieval, we aim a little higher than the nominal target of 50; in the current experiments, the margin is 25%, that is we aim for 62.5 documents. What happens if we hit the target before the end of the period is discussed below.

5.3 T9U thresholding: basic ideas

For the utility-oriented task, however, we go back to our work of TRECs 7 and 8. The basic requirement is to retrieve if the probability of relevance exceeds a certain figure; so we need a model to calibrate the score into a probability value. In TREC–7 we used quite a simple formula; in TREC–8 we tried something a little more complex, which gave us no performance improvement. This year we reverted to the TREC–7model.

The basic model for calibration is:

$$\log \frac{p_d}{1-p_d} = \beta + \gamma \frac{s_d}{ast1} \tag{4}$$

where p_d is the probability of relevance of document d, s_d is its score, and $ast1$ is the average score of the top 1% of retrieved documents (actually, $ast1$ is in itself an example of model-free quantitative prediction). Initial values of β and γ were originally estimated from a logistic regression on old TREC data. For TREC–9, we simply re-used the TREC–7 initial values. Adaptation of β follows the method used at TRECs 7 and 8, summarized in the next section, and takes place after any new documents have been retrieved and/or the query has been reformulated.

Given a document score and an estimated *ast1*, equation 4 can be used to estimate the log-odds of relevance of any specific document. The calibrated score c_d is on a log-odds scale, but can be converted back to a probability p_d:

$$c_d = \beta + \gamma \frac{s_d}{ast1}; \qquad p_d = \frac{\exp c_d}{1 + \exp c_d} \tag{5}$$

for some estimated β, γ and *ast1*.

As we obtain feedback, as well as re-estimating *ast1*, we adjust the calibration by correcting β (γ is left unchanged). We assume a set $\mathcal{F}$ of feedback documents whose relevance is known, of which r are relevant. A Bayesian prior is also assumed, represented by m mythical documents (in addition to those in $\mathcal{F}$), whose estimated probabilities of relevance are assumed to be correct at 0.5. We suppose an iterative sequence of estimates $\beta^{(n)}$ and corresponding values $c_d^{(n)}$ and $p_d^{(n)}$ for each document. Then the gradient descent formula is:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{r - \sum_{d\in\mathcal{F}} p_d^{(n)} + m \frac{1-\exp(\beta^{(n)}-\beta^{(0)})}{2(1+\exp(\beta^{(n)}-\beta^{(0)}))}}{\sum_{d\in\mathcal{F}} p_d^{(n)}(1-p_d^{(n)}) + m \frac{\exp(\beta^{(n)}-\beta^{(0)})}{(1+\exp(\beta^{(n)}-\beta^{(0)}))^2}} \tag{6}$$

$\beta^{(0)}$ is the estimate of β taken from TREC–7.

In the last two TRECs, we ran this correction only once each time. Because the query may have changed substantially since the last adjustment of β, we now (on each occasion we want to modify β) iterate the correction until the change is less than some small constant ϵ. Sometimes (after a substantial change in the query) the old β is badly out, and the gradient descent process becomes unstable. This can be resolved by setting a maximum correction to β in one iteration. In the experiments reported below, m is set at 3 (T9U runs) or 6 (T9P runs); ϵ is 0.01, and the maximum correction in one iteration is 1.0.

5.4 The cross-over: T9P task

A somewhat deeper analysis reveals an interesting cross-over between these two approaches of quantitative and qualitative prediction.

In the T9P task, we may reach the target before the end of the period. After this point the aim is to estimate the threshold score that will maximise the accumulated precision achieved at the end of the period. This requires both qualitative and quantitative prediction. The algorithm is essentially as follows:

1. perform a search with the current query on the accumulated collection so far, and rank the output
2. for the next document in this ranking, predict the number of documents achieving the same score in the future
3. predict the probability of relevance of these documents (from the score calibration)
4. estimate the overall precision that would be achieved if the threshold were set at this score
5. return to step 2
6. when the documents are exhausted, choose the score that gave the highest predicted overall precision as the threshold.

In the experiments reported below, this procedure is initiated when the total retrieved reaches 75% of the target. While the total remains less than the target, the rule is to aim for the target unless this procedure suggests trying for more documents. When the target is reached, then this procedure takes over. [2]

5.5 The cross-over: T9U task

In TRECs 7 and 8, we wanted to ensure that some documents were retrieved early on, even if their scores did not warrant it, in order to get some feedback to improve the query. The mechanism was a ladder of calibrated score values; a particular point on the ladder corresponded to the required utility, but we started lower down the ladder in order to get these initial documents. Both the ladder and the initial starting point were essentially arbitrary: we had no theory or mechanism to determine good values.

The quantitative approach now provides us with at least a way of thinking about the starting point. We would like to start in a position which would give us a small (non-zero) number of documents over the simulated period. The algorithm is essentially as follows:

1. calibrate the scores
2. determine the steps of the ladder
3. initially, or if we have not yet retrieved any documents,
   (a) estimate the threshold required to retrieve a certain target number of documents over the period
   (b) locate the ladder step closest to this threshold
4. if we have retrieved some relevant documents, then take a step up the ladder for every relevant document found so far (stopping at the top).

This procedure may be repeated at intervals. As soon as some documents have been retrieved, we stop being concerned about the target estimation, but remain on the ladder until we accumulate enough relevant documents to climb to the utility point. Because the ladder is defined in terms of the calibrated score, any intermediate stage that requires recalibration of the score (for example query reformulation) will be taken into account automatically.

The ladder currently in use is given in table 2.

Table 2: The Ladder

$P(R\|D)$	$\log O(R\|D)$	
0.33	-0.7	T9U
0.23	-1.2	
0.15	-1.7	
0.10	-2.2	
...	...	

The setting of an appropriate target number of documents is the subject of some of the experiments discussed below. It may also be noted that although there are still several arbitrary elements, this procedure should be a considerable improvement on our methods for TRECs 7 and 8, because the threshold will be set separately for each profile, in a way that relates to the particular query.

5.6 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 the context of the TREC–9 filtering task, we actually need two such collections, respectively including and excluding the training set (Ohsumed 87).

[2] We have discovered a bug in this part of the program, which may cause the threshold to be set incorrectly if no relevant documents have been retrieved by the time we apply this procedure. The effect has not yet been investigated, but will be limited to a small number of topics.

5.7 Query reformulation

In the present filtering system, queries are reformulated as relevance information becomes available, as part of the adaptation process.

The method used is essentially that described in section 3.2. The new offer weight (equation 2) was used, with an absolute threshold. We *also* have a numerical term cut-off, which comes into effect when we have many relevant documents. We use this method right at the beginning, as our way of using the learning examples provided for the TREC–9 task. We repeat it at intervals determined by the retrieval of new relevant documents, frequently initially, and then only occasionally.

We set a limit on the number of relevant documents to be processed; if we have accumulated more than this number, we take only the most recent ones. This was implemented partly as an efficiency measure; however, it could be taken as a response to the possibility that either the user's interests, or the characteristics of the document stream, or both, may drift. In the present experiments, however, we have set this limit fairly high, so that it seldom comes into effect.

The principle tunable parameters of the query expansion method are (a) the maximum number of documents used (set to 100 here), (b) the term cut-off (maximum number of terms in the resulting query, 25 here), and (c) the absolute threshold on the offer weight (zero for these experiments). However, there are several other parameters or controls, e.g. the exact source and method of term extraction, and the form and degree of bias towards query terms.

5.8 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.

As indicated in the system description, the incoming stream is batched somewhat arbitrarily, but with smaller batches initially on the grounds that the system needs to learn faster initially; later in the simulated period, profiles can be expected to have stabilized somewhat. In these experiments the test set (OHSUMED 88-91) is divided into 59 batches, initially 1000 documents per batch; the training set (OHSUMED 87) counts as batch 0.

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 i of incoming documents

1. Run current profiles against batch i
2. Update both cumulative databases (batches 0–i and batches 1–i)
3. For each topic:
   (a) if checkpoint has been reached,
      - reformulate query
      - recalculate $ast1$ and scores of previously retrieved documents
      - re-estimate β using equation 6

   (b) set threshold (using methods described above)

6 Filtering and routing results

6.1 Topic sets

OHSU: 63 queries from the original OHSUMED queries, with relevance judgements from the requesters (no distinction was made between the two "relevant" categories)

MeSH: 4903 MeSH headings, treated as topics. The text of the topic is taken from the scope notes in MeSH; relevance judgements are the assignments of these headings to documents by the NLM indexers

MeSH-Sample: a sample of 500 of the MeSH topic set

6.2 Measures

T9U: linear utility, with relevant document credit set at 2 and non-relevant document debit set at 1, and a minimum utility of -100 for the OHSU topics and -400 for the MeSH topics.

MeanT9U: mean of T9U across topics, no normalisation

MeanSU: mean scaled utility across topics, where scaled utility is T9U divided by the maximum possible value for the topic, namely 2*(Total relevant)

T9P: precision, but with a minimum denominator of the target total number of documents retrieved, namely 50.

MeanT9P: mean of T9P values for each topic

MacR: macro average recall, that is the mean of recall values for each topic

MacP: macro average precision

and for routing runs, AveP (mean average precision) and P@50 (precision at 50 documents retrieved).

6.3 Submitted runs

See table 3. The rules for Batch and Routing allow the use of all the relevant documents in the training set for training, while for Adaptive Filtering 2 (OHSU) or 4 (MeSH) positive examples are provided. Those runs coded bfr or rfr did not make use of all the relevant documents, but only of all those retrieved in the top 100 documents in an initial search on the training set. (See next section for settings of some other parameters, which will explain the differences between some of these runs.)

Table 3: Submitted run results

Run	Type	Topics	Measure	MeanT9U	MeanT9P	AveP
ok9f1po	Adaptive	OHSU	T9P		0.294	
ok9f2po	Adaptive	OHSU	T9P		0.288	
ok9f2pm	Adaptive	MeSH	T9P		0.419	
ok9f1uo	Adaptive	OHSU	T9U	9.70		
ok9f3uo	Adaptive	OHSU	T9U	10.75		
ok9f1us	Adaptive	MeSH-Sample	T9U	46.53		
ok9f3us	Adaptive	MeSH-Sample	T9U	40.10		
ok9bf2po	Batch-adaptive	OHSU	T9P		0.305	
ok9bfr2po	Batch-adaptive	OHSU	T9P		0.305	
ok9bfr2ps	Batch-adaptive	MeSH-Sample	T9P		0.433	
ok9rf2po	Routing	OHSU				0.326
ok9rfr2po	Routing	OHSU				0.317
ok9rfr2ps	Routing	MeSH-Sample				0.245

6.4 Optimization runs

The system as described above contains a large number of settable parameters (tuning constants). Most of the parameters were set on the basis of guesswork, but some adjustments were made following some tests on some "pre-test" topics which had been provided. These pre-test topics are not actually supposed to be representative – indeed they consist of MeSH headings and OHSUMED queries which had been rejected from the main test for one reason or another, and exhibited some very different characteristics. Therefore this tuning process had to be a judicious mixture of experiment and guesswork. A very few of the parameters have been subjected to further testing, after the submission of the official runs, with the main test sets. These are reported here (table 4 and 5).

Note from Table 4 that there appears to be an optimum initial target for utility optimization, but that it depends on both the query set and the evaluation measure. It is higher for MeSH than for OHSU and higher for MeanT9U than for MeanSU. These results would be consistent with the hypothesis that the optimum depends on the number of relevant documents for the topic. The average number of relevant documents for MeSH topics is higher than for OHSU, and the MeanT9U measure is much more affected by topics with more relevant documents, while MeanSU weights all topics equally. But it seems that this difference cannot explain the full extent of the variation: the average number of relevant documents is approximately 50 (OHSU) and 250 (MeSH), but the difference in the optimum initial target is much greater. Another possibility is that the quality of the initial query is also important: a good initial query does not need much priming with relevant documents whereas a poor one does.

The possible effect of the total number of relevant documents raises the question of whether one might be able to make any useful kind of prediction of the optimum for a given topic. A plausible scenario for a real system would be to obtain an estimate from the user (which might or might not be good enough to help).

Table 4: Initial target for utility optimization

Target	MeanT9U	MeanSU	Notes
OHSU topics			
500	-2.37	-.410	
200	6.81	-.144	
150	8.86	-.082	
100	9.69	-.045	ok9f1uo
60	10.22	-.009	
30	10.75	.008	ok9f3uo
15	11.41	.029	
8	11.49	.032	
4	11.22	.033	
2	11.15	.033	
1	11.18	.034	One "zero return"
MeSH-Sample topics			
500	49.55	.076	
200	49.31	.099	
150	48.17	.102	
100	46.53	.102	ok9f1us
60	42.56	.101	
30	40.10	.098	ok9f3us
15	39.53	.097	

The "zero return" noted in Table 4 is a topic for which no documents were retrieved in the entire period. We regard this as a failure, but that run was the only one of the runs reported here that produced any zeros at all.

The data set is rather peculiar in terms of document length: about two-thirds of the documents have abstracts, while the other one-third do not; in the latter case, in effect the only text available is the title. There is therefore a huge discrepancy in document length between the two. b is the parameter in BM25 which controls the effect of doc-

Table 5: Document length

b	MacR	MacP	MeanT9P	Notes
OHSU topics				
0.8	.383	.288	.288	ok9f2po
0.4	.388	.294	.294	ok9f1po
MeSH-Sample topics				
0.8	.189	.430	.430	ok9f2ps
0.4	.181	.412	.412	

ument length. It was hypothesized that a high-precision task might benefit from concentrating on the documents with abstracts; reducing b would have that effect. Therefore in addition to the $b = 0.8$ value which is a good default, we tried a $b = 0.4$ run. This appeared to have some slight benefit with the OHSU topics, but the opposite effect in the MeSH topics (probably not significant).

Note also that the MacP and MeanT9P values are the same for each run. This reflects the success of the target setting and adaptation: either all topics retrieved over 50 documents, or the few which did not quite do so did not show up in the average. This is the case for all adaptive and batch-adaptive runs reported here, though not for non-adaptive batch runs.

6.5 Some comparisons

The new T9P measure provides an interesting opportunity to compare the results of filtering runs with traditional ranked-retrieval runs. The evaluation program trec_eval for ranked retrieval calculates P@n values – precision at n documents retrieved – for various values of n. T9P is a sort of P@50[3]. However, the comparison needs to be qualified, as discussed in the overview paper [12].

This analysis concentrates on the effect of using different amounts of relevance information at different stages. For the routing results, we have pure P@50 on the test set (we include AveP in the table also); no threshold is involved. For batch filtering (non-adaptive), we use exactly the same queries as for routing, but set a once-only threshold intended to retrieve 50 documents. For batch-adaptive or adaptive filtering, we may either adapt only the threshold, or we may adapt both the threshold and the query. All of these may be done starting with all relevant documents in the training set, or with only those which would have been found in an initial search, or with the 2 or 4 positive examples provided for adaptive filtering, or with none. All these conditions are represented in table 6. In the last two columns, MacP figures are in all cases the same as the corresponding MeanT9P figures given. The adaptive filtering rules allowed 4 training relevants for MeSH topics; we have included a '2 relevants' row for comparison with the OHSU topics. The two were chosen as the first two of the four provided.

[3] trec_eval does not by default include $n = 50$; however, a simple modification of a header file allows it

Within each topic set, P@50 reflects AveP quite closely.

Comparing MacP and MeanT9P figures for the 'No adaptation' column, we see that MacP is consistently higher than MeanT9P. This reflects two factors: first, there is a lot of variation in the number of documents retrieved, so that many topics failed to retrieve 50 documents. Second, the initial threshold is often too high. Some further experiments suggest that query adaptation to the relevant documents in the training set tends to interfere with the initial threshold setting to cause this effect.

Comparing the last two columns for the 'All relevant' training, we notice a small decline in performance. The 'all relevant' set may be seen as an unbiased sample of relevant documents; the extra relevants used to modify the query during adaptation are to some extent biased towards the query. However, modifying the query becomes progressively more useful as we start from less relevance information, and adapting the threshold appears always to be beneficial.

There seem to be some differences between the two topic sets as to how useful each level of relevance information is. This may perhaps reflect two things: differences in the quality of the initial queries and differences in the total number of relevant documents per topic.

Comparing P@50 with final MeanT9P in the '2/4 relevants' case, we see that full adaptation just about compensates for the inherent difficulty of MeanT9P.

A different kind of analysis may be made by considering the utility measure. We regard T9P as a high-precision task; however, in order to score above zero on T9U we have to obtain a precision of at least 33%. The difficulty of this task is reinforced by looking at the precision values obtained for T9P runs. For example, in the case of OHSU, these seldom reach 33%. Linear utility (with the sort of parameter values used for the last 3 years) is indeed a hard, high-precision task, and it is not so surprising that we had such difficulty in doing even reasonably well at it.

6.6 Computational load

Running the 60 batches and 4900 MeSH topics is a heavy computational task. Although each batch is quite small, it involves both the 4900 basic searches and all the additional work (including possibly more than one search on the accumulated collection) required for adaptation of a topic, again 4900 times. The scripts used for this task take approximately one week to run on the 550 MHz, 2Gb machine. They could no doubt be made considerably more efficient; nevertheless, the adaptive filtering task must be regarded as computationally heavy – considerably more so than, say, the 100Gb VLC or Web track.

Table 6: Relevance information and adaptation

	No threshold			No adaptation		Threshold adaptation	Threshold and query adaptation	
Training	AveP	P@50		MacP	MeanT9P	MeanT9P	MeanT9P	
OHSU topics								
All training set relevants	.326	.336	ok9rf2po	.357	.274	.309	.305	ok9bf2po
Relevants in top 100	.317	.324	ok9rfr2po	.342	.266	.296	.305	ok9bfr2po
2 training relevants	.277	.294		.281	.251	.268	.288	ok9f2po
No relevants	.228	.260		.240	.221	.236	.280	
MeSH-Sample topics								
All training set relevants	.283	.490		.506	.412	.472	.461	
Relevants in top 100	.253	.455	ok9rfr2ps	.458	.366	.429	.433	ok9bfr2ps
4 training relevants	.245	.437		.428	.375	.413	.430	
2 training relevants	.201	.397		.385	.344	.373	.415	
No relevants	.135	.301		.290	.262	.285	.390	

7 Conclusions

The adaptive filtering task continues to be an interesting and fruitful one to investigate. The new T9P optimization measure has been very successful, both in encouraging the development of the thresholding methods (which are now much stronger for both measures), and in allowing some comparison of traditional ranked-retrieval performance with threshold-based filtering.

References

[1] Elworthy, D. Question answering using a large NLP system. In these proceedings. 2001.

[2] Mitev, N.N., Venner, G.M. and Walker, S. *Designing an online public access catalogue: Okapi, a catalogue on a local area network*. British Library, 1985. (Library and Information Research Report 39.)

[3] Walker, S. and Jones, R.M. *Improving subject retrieval in online catalogues: 1. Stemming, automatic spelling correction and cross-reference tables.* British Library, 1987. (British Library Research Paper 24.)

[4] Walker, S. and De Vere, R. *Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion.* British Library, 1990. (British Library Research Paper 72.) ISBN 0-7123-3219-7

[5] Robertson, S.E. and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, 1976, p129–146.

[6] Robertson, S.E. *et al.* Okapi at TREC–3. In: [10], p109–126.

[7] Robertson, S.E. and Walker, S. Okapi/Keenbow at TREC-8. In: [11], p151–162.

[8] 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).

[9] Robertson, S.E. On term selection for query expansion. *Journal of Documentation* 46, 1990, p359–364.

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

[11] *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).

[12] Robertson, S.E. and Hull, D.A., The TREC–9 Filtering Track final report. In these proceedings. 2001.