Published in: *Information Retrieval* vol.5, pp 239–256 (2002)
Available from: www.springerlink.com
DOI: http://dx.doi.org/10.1023/A:1015702129514

Threshold setting and performance optimization in adaptive filtering

Stephen Robertson

Microsoft Research
St George House
1 Guildhall Street
Cambridge CB2 3NH
UK

Abstract

An experimental adaptive filtering system, built on the Okapi search engine, is described. In addition to the regular text retrieval functions, the system requires a complex set of procedures for setting score thresholds and adapting them following feedback. These procedures need to be closely related to the evaluation measures to be used. A mixture of quantitative methods relating a threshold to the number of documents expected to be retrieved in a time period, and qualitative methods relating to the probability of relevance, is defined. Experiments under the TREC–9 Adaptive Filtering Track rules are reported. The system is seen to perform reasonably well in comparison with other systems at TREC. Some of the variables that may affect performance are investigated.

1 Introduction

This paper describes an experimental adaptive filtering system, built on top of the Okapi search engine, and used for experiments in the TREC adaptive filtering track. We concentrate on the major feature which distinguishes filtering from adhoc retrieval, namely the necessity for thresholding to allow the system to make a binary retrieval decision on each incoming document (as opposed to ranking the documents). We address the issues of initialisation and adaptation following feedback, covering both query formulation and threholding. We also discuss the relation between adaptation and the performance measure to be used. Experiments conducted as part of TREC–9 are reported.

This work may be seen as related to a number of other efforts in the area of adaptive filtering, including particularly the work by a number of groups associated with the TREC adaptive filtering track. The conception of the filtering task is very close to that adopted for the track. Partly as a result of the track, the thresholding problem has stimulated other work (see e.g. Arampatzis and van Hameren 2001, Zhang and Callan 2001). Fuller information about the track is provided in Robertson (2002).

2 The basic system

The search engine is the Okapi BSS, described in detail in successive TREC reports (see e.g. Robertson et al. 1995, Robertson and Walker 2000b). 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 Sparck Jones Walker and Robertson (2000). The BSS forms part of the Keenbow evaluation environment being developed at the Microsoft Research laboratory in Cambridge, and the filtering components discussed in this paper are also part of that environment.

The following subsections describe general components of the system, suitable for adhoc retrieval (including interactive searching with relevance feedback) or filtering.

2.1 Weighting function

The default weighting function is the Okapi **BM25** function first used in TREC–3 (equation 1).

$$\sum_{t \in \mathcal{Q}} w^{(1)} \frac{(k_1+1)f(t,d)}{K+f(t,d)} \frac{(k_3+1)f(t,Q)}{k_3+f(t,Q)} \tag{1}$$

where

$\mathcal{Q}$ is a query, containing terms t
d is a document
$w^{(1)}$ is the Robertson/Sparck Jones weight (Robertson and Sparck Jones 1976) of t in $\mathcal{Q}$

$$w^{(1)}(t) = \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \tag{2}$$

N is the number of items (documents) in the collection
n is the number of documents containing the term
R is the number of documents known to be relevant to a specific topic (zero if no such documents are yet known)
r is the number of relevant documents containing the term (again, initially zero)
K is $k_1((1-b)+b*l(d)/\bar{l})$
k_1, b and k_3 are parameters which depend on the on the nature of the queries and possibly on the database; k_1 and b default to 1.2 and 0.75 respectively, but smaller values of b are sometimes advantageous; in long queries k_3 is often set to 7 or 1000 (effectively infinite)
$f(t,d)$ is the frequency of occurrence of the term within a specific document (often known as tf)
$f(t,Q)$ is the frequency of the term within the topic from which $\mathcal{Q}$ was derived
$l(d)$ and $\bar{l}$ are respectively the document length and average document length measured in some suitable unit.

2.2 Query expansion/modification

Given some relevance information (or some top-ranked documents that are assumed to be relevant for the purposes of blind expansion), the query may be modified or reformulated; the primary purpose of this reformulation is usually to expand the query by adding in new terms not in the original topic, and to give new weights to all the terms to be included. An ineffective query term might also be dropped.

Term ranking and selection

The initial step is to choose terms. Prior to TREC–8 the method used was that proposed in Robertson (1990) by which terms are ranked in decreasing order of a term selection value or offer weight:

$$TSV = r * w^{(1)} \tag{3}$$

and the top n terms are 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 Robertson and Walker (2000b), and is as follows:

$$NTSV = r \log \frac{N}{n} - \log \binom{R}{r} - \log V \tag{4}$$

where r, R, n, N have the same meanings as in the weighting function defined above. V is the size of the vocabulary (number of distinct terms). We may use an absolute threshold criterion with this new offer weight:

$$NTSV > c \tag{5}$$

Zero would be a suitable value for c. However, see below for some experimental results.

Query reformulation

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. In an interactive system, some set of terms may be shown to the user, who can be invited to decide on inclusion. In an automatic system, the selected set of terms can be included automatically.

Model-free methods

Both the offer weight method and the usual relevance-weighting formula may be characterised as model-based. The alternative approach to query reformulation is to perform an iterative optimisation on a training set of documents of known relevance. We used various iterative optimisation methods successfully in the routing task in TRECs 3 to 6. This involves exploring a large space of possible query formulations (choices of terms and weights), and discovering which formulations perform best in a training set. Such an approach relies on having a reasonable number of known relevant items, and is very heavy computationally.

3 System design considerations

In developing a filtering system based on a best-match search engine, one of the primary concerns is with thresholding methods. Following the experimental design for the TREC Filtering Track, we assume that the system has to make a binary decision about each incoming document: whether to retrieve it (send it to the user) or not. This requires that we set a threshold against which to compare the document score which the best-match engine calculates. The usual adhoc retrieval process of simply ranking documents in score order is not open to us.

Some work on thresholding methods for an Okapi-based filtering system was reported in Robertson and Walker (2000a). The basic idea of those methods was to model the probability of relevance associated with a matching score, or in other words to calibrate the score so that it corresponds more closely to a probability. The present set of methods subsumes the ones discussed there.

Essentially, the problem is to predict the future performance of a profile on the basis of whatever data we have about the past, and use this prediction to take decisions about the profile – to optimise it in some fashion, according to some measure. We divide this process into two parts. First we have a conventional query formulation process, resulting in a standard form query such as might be used for adhoc retrieval, consisting of terms and weights. Second, we have a threshold setting stage, which assumes a specific query. This second stage is further subdivided.

We might roughly categorise prediction methods into two classes. On the one hand we have model-based methods, where the prediction is based on an explicit model of the process; the models we are concerned with here are essentially probabilistic. We also have model-free, empirical methods, where some empirically-observed parameter of past performance is simply assumed to apply to the future. This categorisation is certainly over-simplified, but serves to give a general feel for different methods.

Query modification

The model-based methods of query modification which we typically use in Okapi systems were described above. These methods are applicable to filtering query modification. One consideration here is that for different profiles and at different times we may be making use of widely varying numbers of relevant documents, from none at all up to many hundreds. The method therefore has to be robust with respect to this number. This is a reason for choosing the second Offer Weight formula described above (equation 4), since while the number of relevant documents is small it will select only the small number of terms (perhaps none) for which there is sufficient evidence that they are predictive of relevance. As we get more relevant documents, this number naturally grows. There is also a numerical limit on the number of terms, so that after reaching this limit it remains stable.

The same procedure is used at different times, on whatever relevant documents we have accumulated for the profile. There is some argument that older relevant documents should be dropped out of the list at some time, to allow for the possibility of drift in the user's interest or in the characteristics of relevant documents. The present method has a limit on the total number of relevant documents used (so that older ones over this limit are indeed dropped). For most of the experiments reported below, this limit was set high and seldom came into effect, but some experiments with lower limits are described.

In general, however, we expect methods of query modification that have been found to work in other settings will transfer well to filtering. No special methods have been developed for this application. The model-free methods mentioned were not used in the filtering experiments reported here, partly because of their computational demands.

Thresholding and evaluation measures

The problem of setting and adapting thresholds for filtering is a considerable one. Some aspects are discussed by Callan (1998), whose methods are based on the idea of starting with a low threshold (in order to make sure that some documents are retrieved initially), and then adapting it by raising it to make the profile more selective. Some such mechanism is incorporated into the methods discussed below; however, it does seem risky to have an algorithm which can *only* raise the threshold.

Before discussing thresholding, it is useful to introduce one aspect of the TREC filtering track, namely the evaluation measures used. It will be very clear from the discussion below that certain aspects of the system, notably the threshold setting-and-adapting component, need to be closely tied to the measure used to evaluate the system. Unlike in adhoc retrieval, where good methods are often good for any of the various relevance-related measures of performance, threshold adaptation needs to be tuned specifically tuned for the evaluation measure. In a real-life setting, this would mean that there would have to be a mechanism for obtaining from the user some strong indication of how he/she would judge output – not so much for relevance as for quantity. This indication may at its simplest be a guess as to how many documents there are likely to be in a specified period, or how many the user would like to see.

The performance measures for TREC–9 filtering are described in Robertson (2002). The primary measures were a linear utility measure (T9U) and a 'precision-oriented' measure (T9P). The latter is based on a target number of documents to be retrieved over a period. Below we discuss threshold setting for each of these measures, with some indication of how the methods might be extended to other measures.

Thresholding for the precision-oriented task: basic ideas

In the precision-oriented task, we have to attempt to retrieve the best T 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. (As indicated above, we rely on the query and on standard query modification methods to get us as close as possible to the best T documents.) Thus we need to be able to predict what number of documents a given threshold will give us. At this point, it appears that the threshold will relate only to the number of documents, and not to their quality (which is determined by the query). We will see below that this is not necessarily the case; nevertheless, we certainly need to predict quantity.

Given a profile and some history of the stream of documents (that is, some accumulated documents taken from the stream up to this point in time), a model-free approach to the quantitative prediction is more-or-less obvious. That is, if we run the query against the accumulated collection so far, and rank the documents in the usual way, then the obvious prediction of the future number of documents achieving a given score is to look at that past number and adjust pro-rata. (This also requires an estimate of the total size of the stream

over the future period; we assume a constant rate for this purpose, though obviously one might take into account other factors, such as a growth rate of the stream.)

Of course, such an estimate may not be very good, for at least three reasons. One is that we are dealing with the very tail end of a distribution of scores – essentially the outliers, one might argue. Secondly, the topic in question may well go through peaks or troughs of interest. Third, we may be modifying the query over time as a result of relevance feedback. At the least we will have to adjust the quantitative estimate as we go. So the principle is that after every batch of documents, and after query modification if this is triggered, 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.

Since predicting the number retrieved is a little uncertain, and since the evaluation measure penalizes under-retrieval more than over-retrieval, we aim a little higher than the nominal target of T; in the current experiments, the margin is 25%, that is we aim for $1.25 * T$ documents. What happens if we hit the target before the end of the period is discussed below.

Model-based utility thresholding: basic ideas

The utility measure requires a different approach, as reported in Robertson and Walker (2000a). Utility translates directly into a retrieval rule: optimising a linear utility measure is equivalent to retrieving a document if its probability of relevance exceeds a certain value (derived directly from the parameters of the utility function). This implies that we need a direct estimate of the probability of relevance of each document.

In the probabilistic model of retrieval, the document score reflects the probability of relevance, thus in principle we should have that information. However, the usual approach to deriving a scoring method from a probabilistic model takes a fairly cavalier approach to probability – for adhoc retrieval, all that is required is a score which induces the right rank order on the documents. Typically the score is very far removed from an actual probability. Therefore our approach has been to calibrate the score as a probability. The calibration formula is:

$$\log \frac{p_d}{1-p_d} = \beta + \gamma \frac{s_d}{\overline{s_{1\%}}} \tag{6}$$

where p_d is the probability of relevance of document d, s_d is its score, and $\overline{s_{1\%}}$ is the average score of the top 1% of retrieved documents[1]. We used this formula in TREC–7, and estimated initial values of β and γ from a logistic regression on old TREC data. β (but not γ) was adapted on the basis of relevance feedback data (see below). Although the TREC–9 data were somewhat different from the old data used for the logistic regression, we did not have any better data to do new estimates on, so we simply re-used the TREC–7 initial values.

Adapting β

Given a document score and an estimated $\overline{s_{1\%}}$, equation 6 can be used to estimate the log-odds of relevance of any specific document. The result of applying the equation to the score for document d will be denoted c_d (for calibrated score). c_d is on a log-odds scale, but can be converted back to a probability p_d:

$$c_d \quad = \quad \beta + \gamma \frac{s_d}{\overline{s_{1\%}}} \tag{7}$$

[1] Actually, $\overline{s_{1\%}}$ is in itself an example of a model-free quantitative prediction

$$p_d = \frac{\exp c_d}{1 + \exp c_d}$$

for some estimated β, γ and $\overline{s_{1\%}}$.

As we obtain feedback, as well as re-estimating $\overline{s_{1\%}}$, we adjust the calibration by correcting β, but not γ. The reason for this decision has to do with the nature of the feedback. We have relevance information only for documents which scored sufficiently highly to be retrieved: essentially a set of data points at one extreme end of the regression line. Re-estimating the slope of the line from these points would be highly unreliable, whereas we have some chance of getting a reasonable re-estimate of the intercept, given the slope.

We assume a set $\mathcal{F}$ of feedback documents whose relevance is known; assume R of them are relevant. β estimation is based on a Bayesian argument, in order to prevent it going wild with few data (Bookstein 1983). The Bayesian prior is 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:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{R - \sum_{d\in\mathcal{F}} p_d^{(n)} + m\,\frac{1-\exp(\beta^{(n)}-\beta^{(0)})}{2(1+\exp(\beta^{(n)}-\beta^{(0)}))}}{\sum_{d\in\mathcal{F}} p_d^{(n)}(1-p_d^{(n)}) + m\,\frac{\exp(\beta^{(n)}-\beta^{(0)})}{(1+\exp(\beta^{(n)}-\beta^{(0)}))^2}} \tag{8}$$

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

On each occasion we want to modify β, we iterate the correction until the change is less than some small constant ϵ. On occasion (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.

The cross-over: precision-oriented task

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

In the precision-oriented task, what happens if the target is achieved before the end of the period? One alternative is simply to shut down the profile, but this is clearly not optimal, since there may be some very good documents to come later which would have the effect of boosting precision, even though the number of documents retrieved would go above the target. The next obvious idea is to retrieve any document whose probability of relevance exceeds the presently achieved precision. This immediately suggests using the model-based score calibration. But this is still not quite right: if the target is the highest precision achievable, it may well be better to go for a few documents with very high probabilities of relevance, which would significantly boost precision, rather than to go for a lot which might have only slightly higher probabilities of relevance.

So 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 (up to some limit)

2. for the first [next] document in this ranking, predict the number of documents achieving the same score in the future (a pro-rata method which may give a fractional number)

3. predict the probability of relevance of these documents (from the score calibration)
4. from these two data [and that from previous iterations], 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.

Such a process needs to start at least when we have reached the target. It may also be that we could start trying it a little before we reach the target – it may already suggest that we can get better precision by going further than the nominal target. It needs to be repeated every so often as conditions may have changed, and in particular it must be repeated if we have modified the query.

Note that this procedure may also be taken as a model for the optimization of *a*ny relevance-based measure of performance. The predictions of numbers of documents and probabilities of relevance at each score threshold may be used to derive a predicted value of any such measure, and thus to choose the threshold which optimizes the measure. The one additional prediction that might be required for some measures would be an estimate of the total number of relevant documents in any time period, which might also be derived from the distribution of calibrated scores (we have not tested this possibility).

Nevertheless, it is a slightly odd mixture of model-free and model-based methods. The quantitative part of the prediction (how many documents can we expect at a certain threshold) is model-free, while the qualitative part (what is their probability of relevance) is model-based. A more strongly model-based approach, modelling the distributions of scores of relevant and non-relevant documents, is illustrated by the Nijmegen work (Arampatzis et al. 2001).

The cross-over: utility-oriented task

One problem which arises in setting an initial threshold is the high degree of sensitivity involved. A threshold that is set too low will swamp the user with documents in the early stages; one that is set too high will result in the retrieval of nothing at all. While the former case will become obvious very rapidly, the latter is more difficult to detect. If the user receives nothing in the first month, is it because the threshold is too high or because there has simply been nothing relevant in the stream? It is the fact that we are looking for documents with extreme scores only (in effect, the outliers, in statistical terms) that makes threshold setting so sensitive. Another aspect is that in an adaptive system we need some feedback in order to improve the performance. A document that does not warrant retrieval on the basis of its individual probability of relevance may nevertheless contribute overall if retrieved by helping to improve the model.

In Robertson and Walker (2000a) we suggested a method of dealing with this problem, based on a ladder of thresholds, with the top point of the ladder as the threshold that would optimise the utility measure. A profile would start lower down the ladder, but would take one step up for each relevant document retrieved. But this method was very adhoc, with no basis for the choice of either the starting-point on the ladder, or the step size. The method proposed here provides a mechanism for choosing the starting point, though not the step size.

We would like to start in a position which would give us a small (non-zero) number of documents over the simulated period. As soon as we have feedback we can start adjusting to

achieve the required utility, and this will in effect respond to various factors that are initially invisible to us, like how many relevant documents there are and how good the query is. But initially we have only the quantitative estimates to go on.

The algorithm is essentially as follows:

1. calibrate the scores in the usual way for the utility-oriented task
2. determine the steps of the ladder
3. initially, or if we have not yet retrieved any documents,
   (a) estimate (in the usual way for the precision-oriented task) the threshold required to retrieve a certain target number of documents over the period
   (b) locate the ladder step closest to this threshold (the ladder stops at the utility point, so if the target threshold is higher, we simply choose the utility point)
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 1.

Table 1: The Ladder

Threshold expressed in terms of		
p_d	$c_d = \log \frac{p_d}{1-p_d}$	
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 the method reported in Robertson and Walker (2000a), because the threshold will be set separately for each profile, in a way that relates to the particular query.

Overview of thresholding

Thus we see that for the precision-oriented task, we start with purely quantitative prediction but may bring in a mixed qualitative/quantitative prediction later on; for the utility-oriented task we end up with an almost purely qualitative prediction but may start with a different mixed qualitative/quantitative prediction. We can see these two as examples of a general

pattern to tackle any performance measure: in the absence of feedback information, quantitative considerations will dominate, but as feedback information becomes available, implicit or explicit prediction of an optimal threshold for this particular measure will take over.

We re-iterate also that the qualitative prediction is strongly model-based, while the quantitative prediction that we use is model-free. This set of methods might well gain from a model-based approach to the quantitative part.

The accumulated collection

As argued in Robertson and Walker (2000a), 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. It will be very clear that this collection is required for some of the adaptations proposed above. For reasons discussed below, relating to the structure of the test collection, we actually have two such cumulated collections at any one time.

Overview of the filtering procedure

At a particular iteration of the process, any query modification needs to take place before any threshold setting, as the scaling of the threshold will depend very much on the particular combination of terms and weights in the query. Furthermore, it may be necessary, after query reformulation but before threshold setting, to recalculate the scores of the previously-retrieved relevant documents. This is because their scores will be different under the reformulated query than when they were originally retrieved, and the adaptation of β requires these scores.

4 Experiments

The experiments to be reported have two main purposes:

1. A "proof of concept": to discover whether the somewhat complex set of mechanisms described above can function together reasonably effectively; and

2. an evaluation of the importance and effect of some of the many system parameters.

Environment and procedures

The data set used was that defined for the TREC–9 filtering track, based on the OHSUMED test collection, as described in Robertson (2002). Also as discussed there, there are two sets of topics, OHSU which are traditional requests with relevance judgements, and MSH which are constructed from the MeSH subject headings.

For these experiments 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, and for efficiency reasons we make the batches larger. 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.

Some of the query modification processes make use of the accumulated collection, as discussed above. In the circumstances of this experiment, some of these require the exclusion of batch 0, whereas others can make use of batch 0. Therefore we actually have at any stage

two accumulated collections, one including and one excluding batch 0. This is clearly an inefficient way to implement what we require, but serves for the experiment.

For similar reasons, query modification is initiated frequently initially and then less often: the mechanism for doing this is based on checkpoints: in these experiments, 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. After any batch in which this query reaches a new checkpoint, it is modified.

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 $\overline{s_{1\%}}$ and scores of previously retrieved relevant documents;
      - re-estimate β using equation 8;
   (b) set threshold (using methods described above).

Measures

As discussed in Robertson (2002), T9U is essentially a linear utility measure, but with a minimum (maximum negative) value, in order to avoid penalizing a run too harshly as a result of the odd topic which generates thousands of documents. In practice, with reasonable threshold adaptation, this is very seldom invoked, and in fact never in the results reported here. Formally, the measure is defined as:

$$\begin{aligned} \text{utility} &= 2 * (\text{relevant retrieved}) - (\text{non-relevant retrieved}) \\ \text{T9U} &= \begin{cases} \text{utility} & \text{if utility} > C \\ C & \text{otherwise} \end{cases} \\ C &= \begin{cases} -100 & \text{for OHSU topics} \\ -400 & \text{for MSH topics} \end{cases} \end{aligned} \tag{9}$$

In the reported results, MeanT9U is the mean of T9U across queries, with no normalisation. MeanSU is the mean scaled utility across topics, where scaled utility is T9U divided by the maximum possible value for the topic, namely $2 * (\text{Total relevant})$.

The precision-oriented measure is T9P, formally defined as follows:

$$\text{T9P} = \frac{\text{relevant retrieved}}{\max(50, \text{total retrieved})} \tag{10}$$

MeanT9P is the mean of T9P values for each topic. MsP is the mean set precision, that is the mean of precision values for each topic, based on the set of documents retrieved (also known as macro average precision). These will be the same if every topic reaches the target of 50 documents total demanded by T9P; if some topics do not reach this target, MeanT9P will be less than MsP. In fact, in almost all the runs based on optimising T9P (though not those for T9U), the two are the same. We also report MsR, mean set recall.

TREC–9 submitted runs

Table 2 gives the official results of runs submitted at TREC–9 (Robertson and Walker 2001). The rules for Batch allow the use of all the relevant documents in the training set for training; for Adaptive Filtering only 2 (OHSU) or 4 (MeSH) positive examples are provided. Those runs coded bfr did not make use of all the relevant documents, but only of 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 2: Submitted run results

Run	Type	Topics	Measure	MeanT9U	MeanT9P	MsP	MsR
ok9f1po	Adaptive	OHSU	T9P	-6.32	**.294**	.294	.388
ok9f2po	Adaptive	OHSU	T9P	-7.29	**.288**	.288	.383
ok9f2pm	Adaptive	MSH	T9P	14.79	**.419**	.419	.185
ok9f1uo	Adaptive	OHSU	T9U	**9.70**	.125	.346	.253
ok9f3uo	Adaptive	OHSU	T9U	**10.75**	.244	.330	.212
ok9f1us	Adaptive	MSH-SMP	T9U	**46.53**	.224	.436	.182
ok9f3us	Adaptive	MSH-SMP	T9U	**40.10**	.344	.441	.162
ok9bf2po	Batch-adaptive	OHSU	T9P	-4.40	**.305**	.305	.395
ok9bfr2po	Batch-adaptive	OHSU	T9P	-5.40	**.305**	.305	.388
ok9bfr2ps	Batch-adaptive	MSH-SMP	T9P	16.92	**.433**	.433	.187

Looking at the bold figures (which represent the measure for which each run was optimised), in each of the categories optimised for T9P, the best of the runs above achieved the highest T9P result among participants in that category at TREC–9. In each of the categories optimised for T9U, the best of the runs above achieved the second highest T9U result among participants in that category at TREC–9.

However, if we look also at the figures in normal type, it is clear that a run optimised for one measure cannot be expected to perform well on another measure. Any of these runs, entered into the corresponding category for the other measure, would have come well down the list.

Investigating the parameters

The system as described above contains a large number of settable parameters (tuning constants). For the runs reported above, 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 for TREC–9. 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.

Some of the parameters have been subjected to further testing. These are reported here.

Effect of adaptation

We may see the effects of adaptation on the performance by switching it off. We consider the some runs with or without query modification and/or threshold adaptation. In fact query modification without threshold adaptation does not make sense for this system, because the threshold depends on the query. Therefore we consider no adaptation, threshold only, and both: see Table 3.

Table 3: Adaptation (T9P runs)

Topics	No adaptation		Threshold only	Full adaptation	Notes
	1	2	3	4	
	MsP	MeanT9P	MeanT9P=MsP	MeanT9P=MsP	
			(change over col 2)		
OHSU	.281	.251	.268 (+7%)	.288 (+15%)	ok9f2po
MSH-SMP	.428	.375	.413 (+10%)	.430 (+15%)	

Note that the runs with no adaptation show considerable differences between MsP and MeanT9P – in other words, for many topics the target number of documents was not reached. Threshold adaptation solves this problem. Query modification adds further benefit. Some more extensive results on these lines are presented in Robertson (2002).

Initial target for utility runs

Table 4 shows T9U results for different initial targets.

Note 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).

The "zero return" 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.

Drift

One of the concerns in adaptive filtering systems is drift over time (see Allan 1996, Arampatzis et al. 2001). This may occur because the user's interest drifts, or because the nature of the

Table 4: Initial target for utility optimisation

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" – topic retrieving nothing for the entire period
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	

documents relevant to a topic drifts, or both. The simulated TREC–9 filtering task does not allow for user interest drift, since all the relevance judgements for the OHSU set were done at one time, and the MSH set depends on relatively static categories. However, there may indeed be document drift.

This may be investigated in the present system by means of the control on the number of relevant documents used for query modification. The system simply uses the most recent R relevant documents found, or all if there are less than R. For the above experiments R was set to 100, but table 5 shows some results with lower limits.

It seems that there is little evidence of a drift effect. However, what is surprising is how robust the query modification methods are: around 10 relevant documents seem to be quite sufficient to gain all the benefit from the modification, using additional documents does not produce substantial improvement. This suggests that if there were drift (either in the documents or on the part of the user), it would be quite sufficient to restrict query modification to a few recent relevant documents.

These experiments do not address the question of whether the threshold adaptation needs to take account of drift. Because of the intimate connection between the performance measure and threshold adaptation, and because of the (certainly over-simplified) experimental assumption that the performance measure remains exactly fixed throughout the period, this is not likely to be a major issue. If the user's tolerance of volume were to change over the time period, as it well might, the situation would be different – but we have not attempted

Table 5: Using less relevant documents for query modification

Number of relevant documents used	MeanT9P (OHSU)	MeanT9P (MSH-SMP)	Notes
100	.288	.430	ok9f2po/s
50	.288	.430	
25	.288	.430	
10	.287	.427	
5	.285	.423	
0	.268	.413	No query modification

to simulate this change in the TREC filtering setup.

Expansion terms

The absolute term selection method described in section 2.2 is based on a significance argument, and uses a term scoring method (selection value or offer weight), together with a threshold similar to those used in significance calculations, namely the probability of a noise term achieving such a score. However, because of the number of terms in the vocabulary, this threshold is set very high, to correspond to a small or fractional number of noise terms. Thus the default threshold of zero corresponds to a probability of 1 in the vocabulary size V – in other words if the entire vocabulary were to consist of random noise words, this threshold could be expected to allow one single such term through. This is a very selective criterion, and will typically select only one or two terms if the number of relevant documents is less than 3. Results reported in Robertson and Walker (2000b) suggested (albeit rather weakly, and only for blind feedback) that a slight relaxation of this criterion, to accept more terms, might be advantageous.

Some runs have been conducted with different thresholds – see Table 6. 'Expected number of noise terms' is the number of terms that might be selected if the entire vocabulary consists of noise terms to which the null hypothesis applied.

Table 6: Selecting query expansion terms

Threshold	Expected number of noise terms	MeanT9P (OHSU)	MeanT9P (MSH-SMP)	Notes
-5	150	.281	.409	
0	1	.288	.430	ok9f2po/s
2	.13	.296	.438	
5	.006	.305	.442	
10	5×10^{-5}	.302	.440	
20	2×10^{-9}	.296	.434	
∞	0	.281	.427	No expansion terms (term reweighting only)

The best results in this table, with a threshold of 5, equal or outperform the batch filtering results submitted to TREC. It is likely that at least some of this benefit would accrue to other

runs reported in Table 2, if (as argued above) the query modification stage is relatively independent of the perfomance measure and the threshold setting stage. To test this hypothesis, we reran the ok9f3uo utility run (Table 2) with a threshold of 5, and obtained a T9U of 12.98, an increase of 20% over the official run result of 10.75.

We have found these results quite surprising. They point strongly in the opposite direction – that it is better to make the criterion even more stringent. The effect of this is to ensure that no new terms at all are selected from a small number of relevant documents. Effectively they suggest not doing any query expansion until we have accumulated 5 or 10 relevant documents. While we have generally, with the Okapi system, obtained good results with very selectively expanded queries, these results are extreme. They may also be hard to reconcile with the results concerning drift above – clearly if we restrict ourselves to the last few relevant documents, a high term selection threshold will always select very few terms. The interactions here, and with the maximum number of terms to be selected (set to 25 in all the above experiments), remains to be investigated.

Computational load

Running the 60 batches and 4900 MSH 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 a 550 MHz, 2Gb machine (the 500 MSH-SMP topics take about 18 hours and the 63 OHSU topics between two and three hours, on a slower but twin processor machine). It is clear that they are not primarily designed for speed, and 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 track at TREC.

This is one reason why we have not yet attempted any of the iterative query optimization techniques which we used so successfully for routing in earlier TRECs.

Other authors (e.g. Callan 1998, Arampatzis et al. 2001) place much more emphasis than we have done on light-weight systems which do not require extensive query history or an accumulated collection.

5 Conclusions

Adaptive filtering, while sharing many aspects with traditional, 'adhoc' text retrieval, is a somewhat complex task, and introduces many new complications and considerations.

It also presents new opportunities. Filtering systems may learn from the quantity of output, but far more far-reaching is the possibility of learning from relevance feedback. The potential for adaptation is considerable. Furthermore, filtering provides opportunities for testing relevance feedback ideas that are considerably more sensitive and revealing than many traditional methods.

The problem of threshold setting and adaptation looms large. There is a much stronger relation between system design and the choice of performance measure which it is desired to optimise, than between any system parameters and any of the common measures of performance in adhoc retrieval. The methods of threshold setting and adaptation need to be

designed specifically for the performance measure, and a system tuned in this way for one performance measure is quite likely to do badly according to another measure.

References

Allan, J. (1996) Incremental Relevance Feedback for Information Filtering. In: *SIGIR 96: Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval*. Edited by H.-P. Frei et al. ACM (pp 270–278).

Arampatzis, A. and van Hameren, A. (2001) The Score-Distributional Threshold Optimization for Adaptive Binary Classification Tasks. In: *SIGIR 2001: Proceedings of the 24th Annual International Conference on Research and Development in Information Retrieval*. Edited by W.B. Croft et al. ACM Press (pp 285–293).

Arampatzis, A., Beney, J., Koster, C.H.A. and van der Weide, T.P. (2001) Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering. In: *The Ninth Text REtrieval Conference (TREC–9)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST (to appear).

Bookstein, A. (1983) Information retrieval: a sequential learning process. *Journal of the American Society for Information Science* 34, pp 331–342.

Callan, J. (1998) Learning while filtering documents. In: *SIGIR '98: Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*. Edited by B. Croft et al. ACM Press (pp 224–231).

Robertson, S.E. (1990) On term selection for query expansion. *Journal of Documentation* 46, pp 359–364.

Robertson, S.E. (2002) Comparing the performance of adaptive filtering and ranked output systems. *Information Retrieval*, 5, pp xxx–xxx.

Robertson, S.E. and Sparck Jones K. (1976) Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, pp 129–146.

Robertson, S.E. and Walker, S. (2000a) Threshold setting in adaptive filtering. *Journal of Documentation*, 56, pp 312–331.

Robertson, S.E. and Walker, S. (2000b) Okapi/Keenbow at TREC-8. In: *The Eighth Text REtrieval Conference (TREC–8)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST (NIST Special Publication no. 500-246) (pp 151–162).

Robertson, S.E. and Walker, S. (2001) Microsoft Cambridge at TREC-9: filtering track. In: *The Ninth Text REtrieval Conference (TREC–9)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST (to appear).

Robertson, S.E. *et al* (1995). Okapi at TREC–3, In: *The Third Text REtrieval Conference (TREC–3)*. Edited by D.K. Harman. Gaithersburg, MD: NIST (NIST Special Publication no. 500-225) (pp 109–126).

Robertson, S.E. (this issue) Introduction to the special issue: Overview of the TREC Routing and Filtering tasks. *Information Retrieval*, 5, pp xxx–xxx.

Sparck Jones, K., Walker, S. and Robertson, S.E. (2000) A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36, pp 779–808 (Part 1) and 809–840 (Part 2).

Zhang, Y. and Callan, J. (2001) Maximum Likelihood Estimation for Filtering Thresholds. In: *SIGIR 2001: Proceedings of the 24th Annual International Conference on Research and Development in Information Retrieval*. Edited by W.B. Croft et al. ACM Press (pp 294–302).