The TREC-7 Filtering Track: Description and Analysis

David A. Hull

Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan France
hull@xrce.xerox.com

Abstract

This article describes the experiments conducted in the TREC-7 filtering track, which consisted of three subtasks: adaptive filtering, batch filtering, and routing. The focus this year is on adaptive filtering, where the system begins with only the topic statement and must interactively adjust a filtering profile constructed from that topic in response to on-line feedback. In addition to motivating the task and describing the practical details of participating in the track, this document includes a detailed graphical presentation of the experimental results and provides a brief overall analysis of the performance data.

1 Introduction

Text filtering is the process of sorting through large stores of textual data for information which fits a user's profile. This task is crucial for information processing in the modern age. Filtering is often described as the inverse of the traditional search problem. Instead of providing a query which is processed once on a static document collection, searchers define a persistent profile which is compared to a stream of arriving documents. Text filtering or routing has been a core task in TREC since the first year, and it continues to evolve to reflect changes in the way information is processed in the real world. The current focus of the filtering track is on the realistic simulation of on-line time-critical text filtering applications. There are many important aspects to the text filtering problem, but the TREC experiments are designed primarily to analyze the performance of document ranking and selection algorithms. Speed and scalability are important only to the extent that they enable systems to process 50 user profiles over a stream of a few hundred thousand documents in reasonable time. Evaluation is based only on the quality of the retrieved document set.

The core filtering task can be described briefly as follows:

> Given a topic description, build a filtering profile which will select the most relevant examples from an incoming stream of documents. As the document stream is processed, the system may be provided with a binary judgement of relevance for some of the retrieved documents. This information can be used to adaptively update the filtering profile.

Filtering differs from search in that documents arrive sequentially over time. The filtering track is divided into three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, systems start with only the original user profile which they must use to build a text classification rule. In batch filtering and routing, the systems can also take advantage of a large set of training documents, a sample of which have already been evaluated as relevant or not relevant with respect to that profile. The difference is that in batch filtering, the system must decide either to accept of reject each document, while in routing, the system can return a ranked list of documents. The most important change between the TREC-6 and the TREC-7 experiments is the new emphasis this year on the adaptive filtering task.

The search task has traditionally been evaluated by average precision curves and other rank-based measures which use an extensive ranked list based on the scores of all incoming documents to determine performance. This list cannot be created until all documents have been scored and ranked, which means that system performance is not measured as a function of time. However, the underlying model upon which filtering is based is one where a stream of documents are compared to a long-lasting set of query profiles. This means that adhoc retrieval and routing simulate a non-interactive process where users look at documents only once at the end. For filtering, it is more realistic to assume that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than attempt to simulate a particular task which might allow for batching and partial ranking of the document set, we choose to operate at the opposite end of the spectrum, assuming that the user wants to be notified about each potentially interesting document immediately after it arrives. This means that the decision to accept or reject a document must be independent of subsequently arriving documents. Therefore, traditional rank-based evaluation measure are not appropriate. Adaptive and batch filtering results will consist of unordered sets of documents which will be analyzed using set-based evaluation measures. More background on the filtering track, including some specific scenarios, can be found in the TREC-5 filtering track description [2].

2 TREC-7 Task Description

The primary goal for TREC-7 is to move towards a more realistic simulation of the filtering task by concentrating on adaptive filtering. Due to the large number of training documents evaluated for relevance, routing and batch filtering can be viewed as static machine learning or text classification tasks. This is an unrealistic simulation of most filtering applications for the following reasons. First, the training set is far too large. Second, the training set has been built through the TREC pooling process, meaning that the documents were retrieved from many different systems. Most filtering systems must rely on user feedback from documents they have found themselves or from the results of a single independent search. Adaptive filtering addresses this issue by eliminating the training documents. However, batch filtering and routing are still included to maintain continuity with previous years and to encourage as broad a range of systems to participate as possible. It is very important to continue these tasks because they substantially improve the quality of the document pool used for evaluation. Adaptive filtering is a much harder task, meaning that the percentage of relevant documents retrieved in these runs will be much lower. The other tasks will help us to get a broader coverage of the relevant document space. This section is devoted to a detailed description of the three subtasks and the topics and documents used in each task.

The primary corpus for the TREC-7 filtering experiments consists of 3 years of Associated Press newswire covering the period 1988-1990. There are approximately 80,000 documents from each year for a total of 240,000 documents. The AP collection is partitioned by year over the first three TREC disks. The documents are ordered roughly by date on the disks, and all systems were required to process the documents in this order. The AP newswire covers a broad variety of domains and the documents average roughly 450 words in length. This year's experiments used topics 1-50 which were constructed for the first TREC conference. These topics are quite long detailed descriptions of the information need, ranging from 50-250 words in length with an average length of roughly 100 words. They are divided into title, description, narrative and concept fields. For an example of the style of these long topics, see the introductory paper in the TREC-6 proceedings [4]. Since topics 1-50 have been used at several previous TREC conferences, there are relevance judgements available for the 1988 and 1989 AP collection. However, the 1988 judgements do not have the quality and coverage typically associated with TREC, since they come from the TREC-1 experiments. The small number of first-year systems had to struggle just to complete the task and the documents were sampled in less depth than in recent years. Access to these relevance judgements was restricted in different ways depending on the subtask. The 1990 AP collection represents a clean test set, since it has no relevance judgements for these topics.

2.1 Adaptive Filtering

Previous versions of the TREC filtering task have been viewed as unrealistic because they provide too much training data, and because this training data comes from the previous search results of many different systems which use many different search algorithms. In practice, most systems can only expect relevance judgements from documents which they have been responsible for retrieving. The adaptive filtering task is designed to model this situation. In this task, each system starts only with the topic description and no evaluated documents. Documents arrive sequentially and the system can update the query profile in response to previously viewed documents. In addition, each document retrieved will be immediately evaluated for relevance, and that information will be passed on to the system.

It is not possible within the TREC framework to actually provide new relevance judgements to filtering systems as they see documents. Instead, the interactive component must be simulated using training data and previously obtained relevance judgements. In this model, the relevance judgement is available but hidden from the system until it decides whether or not it will retrieve a document. Relevance judgements from unretrieved documents are never revealed to the system. Note that test documents are evaluated in the same fashion, but relevance judgements are not immediately available (unless the system has a user providing manual feedback). This means that systems may wish to select a document not only according to its likelihood of relevance, but also according to its value as a training observation to improve future filtering performance. This makes adaptive filtering a more interesting and more complex task. The final retrieved set consists of only the documents selected during the first pass over the collection. Systems are not supposed to backtrack and use an improved user profile to retrieve previously ignored documents. Unfortunately, this point was not entirely clear in the initial track description and several systems did not completely adhere to this restriction.

Systems had the option to choose whether to use the rest of the TREC document collection (excluding AP) to generate collection frequency statistics (such as IDF) or auxilliary data structures (such as automatically-generated thesauri) or begin with no prior information. Systems were given access to relevance judgement for retrieved documents during processing, but only if the document was selected for that topic. Systems were specifically prohibited from using relevance judgements from unretrieved documents or relevance judgements for topics 1-50 from other parts of the TREC collection. The text from each processed document could be used to update term frequency statistics or auxilliary data structures whether or not the document was retrieved for any topic. There was also the option to treat unevaluated training documents which were retrieved as not relevant. Evaluation was based on two utility functions, as described in section 2.3.

2.2 Batch Filtering and Routing

In batch filtering, the documents and relevance judgements from the 1988 AP collection are available to all systems as a training set. Systems are prohibited from using relevance judgements for topics 1-50 from any other part of the TREC collection. As in adaptive filtering, systems can also have instant access to the relevance judgement (if it exists) for any document retrieved while processing the rest of the AP collection. Evaluation is based on utility. In routing, like batch filtering, the documents and relevance judgements from the 1988 AP collection are provided in advance as a training set. In addition, routing systems are free to use relevance judgements from other parts of the TREC collection (except 1989-1990 AP) for constructing the initial filtering profiles. The other difference is that routing systems return a ranked list of the top 1000 retrieved documents, and are evaluated according to average uninterpolated precision, much like adhoc search. Batch filtering and routing are included in the TREC-7 task to open participation to as many groups as possible and to improve the quality of the document pool used for evaluation.

2.3 Evaluation

Filtering systems are expected to accept or reject each document as it arrives and it is assumed that the user may well look at accepted documents immediately. Therefore, the output of the filtering system is treated as an unordered set of documents. This means that evaluation measures based on a ranked set of documents, such as precision-recall curves, are not appropriate. It is also important to handle topics with no relevant documents in a reasonable fashion, since in this case a system must keep the retrieved document set as small as possible. Classical set-based evaluation measures from information retrieval such as raw precision and recall do not behave gracefully for topics with few or no relevant documents. For example, the precision of a system which returns one non-relevant document is zero. The precision of a system which returns one thousand non-relevant documents is also zero. However, the former system is doing a far better job of filtering than the latter. As an alternative, we rely on a measure from information theory known as utility.

2.3.1 Utility

Utility assigns a value or cost to each document, based on whether it is retrieved or not retrieved and whether it is relevant or not relevant, as shown in the contingency table below:

	Relevant	Not Relevant
Retrieved	R+ / A	N+ / B
Not Retrieved	R- / C	N- / D

```
Utility = A*R+ + B*N+ + C*R- + D*N-
```

The variables R+/R-/N+/N- refer to the number of documents in each category. The utility parameters (A,B,C,D) determine the relative value of each possible category. A positive utility parameter can be thought of as the value of each document in that category, while a negative utility parameter is the cost of classifying a document in that category. Therefore, the larger the utility score, the better the filtering system is performing for a given query profile. For TREC-7, we test two different settings of the utility parameters:

```
F1 = 3*R+ - 2*N+        --> retrieve if P(rel) > .4
F3 = 4*R+ - N+          --> retrieve if P(rel) > .2
```

Note that the second utility function is labelled F3 to distinguish it from the slightly different F2 utility function which was used last year. The F2 function had a negative C parameter. This function was changed because the true number of unretrieved relevant documents is difficult to estimate and because most systems returned a negative F2 utility. This does not necessarily mean that they were performing badly, rather many topics had a large number of relevant documents. Filtering according to a utility function is equivalent to filtering by estimated probability of relevance. Therefore, the utility functions above are listed with the appropriate probability thresholds. Readers can find the general formula for converting a utility function into a probability threshold in Lewis [3], and a derivation of the formula can be found in any general book on decision theory.

Utility is not an ideal measure for judging the performance of filtering systems for a number of reasons. Utility scores will vary widely from topic to topic based on the number of relevant documents, and it is difficult to average or compare them across topics. Since we use linear utility functions, all relevant documents are equally important, no matter how many exist or have been retrieved for a given topic. There is no sense of diminishing value for the thousandth relevant document. In practice, users might well have different utility functions for each topic, based on the nature of the topic and their information need. For administrative and practical reasons, it is too difficult to define a separate utility function for each topic. However, the utility measure handles topics without relevant documents easily, which is an important advantage. We will start to look at techniques for scaling and normalizing utility in this year's evaluation. The most appropriate evaluation strategy for real filtering systems depends on the task and the user model.

2.3.2 Pooling vs. Sampling

In general, the size of the retrieved set is unbounded, making accurate evaluation of performance difficult for some topics, regardless of measure. The traditional pooling approach to document assessment has been augmented with a random sampling strategy. In pooling, the top N retrieved documents from each run and each topic are merged to create a single document pool which is assessed. All documents which do not appear in that pool are assumed to be not relevant. This strategy is reasonably fair for all participants, but results in performance estimates that are biased downwards. For filtering, the retrieved set is unranked, so one cannot simply select the top N documents. The approach for filtering is therefore to select a random sample of size N from the retrieved set for each system. If the retrieved set is smaller than N, all documents are selected.

Pooling is less than ideal for filtering topics where the retrieved sets are much larger than N. First of all, the filtering pool is of lower quality because the documents are randomly sampled from a large retrieved set rather than obtained by selecting the top ranked documents. Fortunately, this effect is mitigated somewhat in the TREC-7 experiments because some routing runs, which are based on ranked retrieval, also contribute to the pool. Second, the topics with large retrieved sets are the ones which will tend to have the most relevant documents, and thus will suffer from the most bias due to incomplete assessment. Fortunately, we know from sample theory that the proportion of relevant documents in a simple random sample is an unbiased estimate of the proportion of relevant documents in the population. For the utility function F1, we can convert an estimate of the proportion of relevant documents directly into an estimate of utility via the following formula ([2], p.81, eq 1):

$$\mathrm{F1} = ((A - B) * (r/n) + B) * N$$

where n is the size of the sample, r is the number of relevant documents in the sample and the other terms are the same as defined above. Another nice property of the sampling approach to evaluation is that we can also calculate the standard error of the utility estimate, which is given by

Utility	R1	R2	R3	R4
T1	0	0	-18	4
T2	150	276	160	75
T3	-6	-44	-43	-11

Rank	R1	R2	R3	R4
T1	2.5	2.5	1	4
T2	2	4	3	1
T3	4	1	2	3

Table 1: Pseudo-example of ranking runs R1-R4 for topics T1-T3.

([2], p.87, eq 30):

$$\mathrm{SE(F1)} = ((A-B)^2 N \frac{(N-n)r(n-r)}{n^2(n-1)})^{1/2}$$

For more details on the mathematics of sampling, see Lewis [2]. Sampling provides unbiased estimates of utility, but there is a price. Since a lot of information is thrown away (all the relevance judgements for documents sampled by other systems but not the one being evaluated), the sampled estimates tend to have much more variation.

2.3.3 Averaging and Comparing Systems with Utility

When evaluation is based on utility, it is difficult to compare performance across topics. Simple averaging of the utility measure gives each retrieved document equal weight, which means that the average scores will be dominated by the topics with large retrieved sets (as in micro-averaging). Therefore, we use two different techniques to average performance, ranking within topics and scaling of the utility scores. Average rank statistics have been used for the last two filtering tracks while scaling of the utility function is new this year. Each method has its advantages and disadvantages, as will be described below.

The average rank measure is computed in two steps: (1) for each topic, systems are ranked in order of their performance, (2) the ranks are averaged by system over all topics. Systems are ranked from lowest utility to highest utility, meaning that the larger the average rank score, the better the system is performing with respect to its competitors. Table 1 presents a pseudo-example of the ranking process. The advantage of the average rank measure is that all topics are equally important in determining a system's performance. The measure is insensitive to topics with very large retrieved sets and wide variation in the utility scores. Average rank scores generated by the same set of systems are directly comparable, even if they are based on different evaluation measures or different retrieval tasks. This makes it possible to compare systems in situations where it would otherwise be difficult (as is the case with utility). On the negative side, all topics are equally important, meaning that topics with large variation which reflect real differences between systems do not receive higher weight. There is no absolute standard of performance and scores are only meaningful relative to the set of systems being analyzed. The results depend on the systems being compared, so adding or removing a system will change the scores of other systems.

We can use non-parametric statistical tests to determine the significance of average rank differences between systems. In order to keep the results simple and readable given the large number of experiments and competing systems, we will only present the pairwise comparisons with respect to the best performing system. For each experiment, we apply two different tests: a conservative test (a non-parametric variant of the Newman-Keuls multiple comparisons test) and a more powerful test (a non-parametric variant of the Least Significant Difference (LSD) test). For more information on the statistical tests and their performance characteristics in TREC-style IR experiments, please consult our NIST technical report [1]. The alpha level is set at 0.05, which corresponds to the error rate for each pairwise comparison in the LSD. The true pairwise error rate for the Newman-Keuls test is an order of magnitude smaller.

While direct averaging of utility scores is undesirable, due to the wide variation in the number of relevant documents per topic, it is possible to scale the utility scores prior to averaging.[1] The most obvious scaling strategy is to divide by the maximum possible utility score for each topic. However, this approach is seriously flawed for negative utility scores. A system which returns one hundred non-relevant documents will receive a score of -100 for a topic with one relevant document and -1 for a topic with one-hundred relevant documents. Since the maximum possible positive score on a topic is 1, topics with negative utilities will dominate the average. Therefore, a more complex utility scaling function is required.

For the TREC-7 experiments, we test the following utility scaling function:

$$u_s^*(S,T) = \frac{\max(u(S,T), U(s)) - U(s)}{MaxU(T) - U(s)}$$

where $u(S,T)$ and $u_s^*(S,T)$ are the original and scaled utility of system S for topic T, $U(s)$ is the utility of retrieving s non-relevant documents, and $MaxU(T)$ is the maximum possible utility score for topic T. This scaling function assigns a lower bound to the utility function which can be set with the parameter s. There is a reasonable justification for this approach. Assume that a filtering profile is performing really poorly. At some point, the user will get fed up with reading non-relevant documents and delete the profile entirely. The parameter s sets the number of non-relevant documents at which the user's tolerance is exhausted. All utility scores less than $U(s)$ are set to $U(s)$. Therefore, utility scores can range between $U(s)$ and $MaxU(T)$ and the scores are renormalized to range between 0 and 1 and then averaged.

The parameter s is set once for all topics, and unlike average rank, a system's scaled utility score is absolute, remaining the same regardless of the other systems are being evaluated. The scaled score can be interpreted relative to the perfect system (utility of 1.0) and the worst possible system (0.0). Unfortunately, the average scaled utility score is highly dependent on the definition of the worst possible system, as determined by the parameter s, and so is only meaningful in relation to a particular lower bound. Since the decision on where to put the lower bound is fairly arbitrary, we will plot the average scaled utility scores over a range of values for s (25-800). A low value for s differentiates more between systems that do well on topics with few relevant documents. On the other hand, it reduces the penalty for systems which do very poorly on these or other topics. One could define a two-parameter scaling model which sets the zero point for utility scaling and the lower bound for acceptable performance to different values, thus allowing for negative scaled utility. This would make it possible to distinguish to the two types of behavior described above. However, for simplicity, we choose to use the same value for the zero point and the lower bound in these experiments.

2.4 Submission Requirements

Each participating group could submit up to four adaptive filtering runs and up to two runs each for batch filtering and routing. There were no required runs this year. Runs were classified into one of three categories:

(A) Automatic - Any run which uses fully automatic methods for profile construction and updating. This can include automatic learning from test documents as they are filtered.

(B) Manual - Any run which uses manual techniques for profile construction, up to and including making additional relevance judgments on training documents. No manual intervention

[1] Paul Kantor first proposed that scaling is possible, while Stephen Robertson suggested the initial variant of the scaling method described here.

based on information from the test documents is allowed, although automatic learning is still permitted.

(C) Manual Feedback - Any run which uses manual techniques for updating profiles based on previously viewed test documents. The run may or may not also use manual techniques for profile construction.

There are no training documents for adaptive filtering, so manual intervention (B) is limited to profile construction for this task. In practice, none of the groups submitted manual runs. Groups were also asked to indicate whether they used other parts of the TREC collection to build term collection statistics or other resources.

Due to assessment constraints, NIST only promised to evaluate a maximum of 100 documents per topic per participating group. In the end, NIST had enough resources to evaluate two sets of 100 documents per group. These documents were obtained as follows. Each group was asked to select one run based on the F1 utility function and one run based on the F3 utility function for sampling. If the document set for the F1 utility run on topic T was greater than 100 documents, 100 documents were sampled for assessment. Otherwise, all documents were assessed. Furthermore, a sufficient number of document from the F3 Utility set were sampled to bring the total set size up to 100 documents. If the union of the F1 Utility and F3 Utility sets was less than 100 documents and the group submitted a ranked retrieval run, then the sampling program went down the list in rank order adding new documents until a sample of size 100 was extracted. If the group submitted two sets of adaptive filtering runs, this process was repeated for the second set.

3 TREC-7 results

The TREC-7 filtering track had 12 participating groups who submitted a total of 46 runs, distributed as follows:

	# groups	# runs
Total	12	46
-----	--	--
adaptive	8	24
batch	6	12
routing	6	10

Here is a list of the participating groups, including [abbreviations] and (run identifiers). Participants will generally be referred to by their abbreviations in this paper. The run identifiers can be used to recognize which runs belong to which groups in the plotted results.

- AT&T Labs Research [AT&T] (att98f)
- Canadian Imperial Bank of Commerce / Active On-Line Systems [CIBC] (sigma)
- City Univ. London / Univ. Sheffield / Microsoft [CitySM] (ok7ff)
- CLARITECH Corporation [CLARITECH] (CLARIT)
- IBM Almaden Research Center [IBM-A] (arc98cs)
- IRIT / University of Toulouse (IRIT) (Mer)
- University of Iowa [UIowa] (IAHK)

Organization	1st yr?	Adaptive	Batch	Routing	Resources
AT&T	-	-	X	X	-
CIBC	X	X	-	-	-
CitySM	-	X	-	-	TREC
CLARITECH	-	X	X	-	TREC
IBM-A	X	-	-	X	N/A
IRIT	X	X	X	X	-
UIowa	X	X	X	-	-
UMass	-	X	-	-	TREC
NTT Data	X	-	X	X	E/J dict
CUNY	-	X	X	X	TREC
Rutgers-K	-	-	-	X	N/A
TNO	X	X	-	-	TREC

Table 2: Listing of subtask participation and resources used for each group (X = participant).

- University of Massachusetts Amherst [UMass] (INQ5)
- NTT Data Corporation [NTT Data] (nttd7)
- Queens College CUNY [CUNY] (pirc8)
- Rutgers University [Rutgers-K] (AntRout)
- TNO-TPD/HFRI [TNO] (TNO)

Table 2 lists the subtasks each group participated in, the resources they used (TREC = documents from other parts of the TREC collection, E/J dict = English/Japanese phrase dictionary), and whether this is their first year participating.

3.1 Summary of approaches

In this section, we briefly describe the techniques used by each of the groups in their experiments. The information presented here is based on the material provided by the participants in their draft papers, and so should be considered preliminary and incomplete. For more information, please consult the site report papers included in this volume [5]. IBM-A and Rutgers-K did not provide notebook papers so there is no information available on these systems. Both participated only in the routing subtask.

AT&T participated in batch filtering and routing. One routing run (fr5) was based on a scaled down version of their TREC-6 routing system. The other routing run (fr4) and the batch filtering runs were generated from their new ATTICS toolkit for machine learning of text classifiers. The TREC-7 runs use a two-pass Rocchio algorithm, the first pass based on only the judged documents and the second passed based on the judged and top 5000 unjudged documents (assumed not relevant). The version of ATTICS used for TREC-7 had no feature selection, stop listing, or term weighting.

CIBC participated in adaptive filtering. Their system uses an adaptive Rocchio relevance feedback algorithm with initial feature weights based on the AT&T TREC-6 system. The initial profile is represented by its 20 most important terms and query expansion is limited to 20 additional terms. The system is built on top of a multi-agent architecture, with agents assigned to feature extraction, profile generation, and profile selection (thresholding).

CitySM participated in adaptive filtering. They concentrated on accurate threshold selection. The runs ok7ffx2 use the same initial query throughout while runs ok7ffx3 use probabilistic reweighting. No query expansion was performed. Thresholds are initially calibrated by applying logistic regression to the ratio of the observed score to the average score of the top 1% of retrieved documents (ast1). A two parameter logistic model is fit to an independent topic and document set. Both ast1 and the intercept term are adaptively adjusted on a weekly basis as training data becomes available. CitySM assumed that there is an advantage to selecting more documents at the beginning to help calibrate the thresholds more accurately. Therefore, the system uses a ladder of thresholds. Each profile starts at the bottom of the ladder and the threshold is increased as relevant documents are discovered.

CLARITECH participated in adaptive and batch filtering. They used their TREC-6 Rocchio relevance feedback algorithm for term selection and weighting and concentrated on accurate threshold selection for adaptive filtering. The initial thresholds are set to a fixed delivery ratio of documents (e.g. 1 out of 2000) tuned on an independent corpus. Thresholds are allowed to vary between a lower bound zero utility value and an upper bound optimal value. The optimal threshold gives the best utility on the training data. The zero utility threshold is the largest score threshold which generates negative utility. The actual threshold is determined by two parameters, a fixed threshold bias and a correction factor which is based on the system's confidence in its estimate of the optimal threshold. Documents are batched in gradually increasing batch sizes ranging from 3000 initially up to 24,000. The runs (afFxb) stop retrieving documents on AP90 for topics which have a negative accumulated utility over AP88 and AP89.

IRIT participated in all three subtasks. Their system is based on a 3-layer neural network model, with layers consisting of queries and documents connected by a layer of terms. For batch filtering and routing the system computes the difference between the observed output of the network and the desired output based on the judged training documents, and this error term is propagated back through the network. This process is repeated over many iterations. A similar approach is used for adaptive filtering. Unfortunately, they misunderstood the adaptive filtering task and thought that they were supposed to rerun the final adaptive filtering profile over the entire AP collection to generate the retrieved set. Clearly, this approach leads to much better performance than selecting documents only during the first pass, so their runs were deleted from the adaptive filtering comparison graphs. However, the results of their system can still be found in the system by topic performance tables.

UIowa participated in adaptive and batch filtering. Their system introduces a two-level clustering approach to filtering. The original topic statements are used to construct a *primary* cluster profile with a low fixed threshold for membership. Documents which exceed this threshold are then clustered dynamically into a set of *secondary* sub-clusters. When the similarity between the secondary cluster and the primary cluster profile exceeds a certain visibility threshold, all documents in the secondary cluster are presented to the user for evaluation. Note that this actually violates the conditions of the filtering track which state that a binary decision to accept or reject must be made immediately after it is filtered. Each secondary cluster is then labelled red (don't retrieve) or green (retrieve) according to the relative frequency and density of relevant, non-relevant, and evaluated documents. Future members of these sub-clusters are treated accordingly.

UMass used the same incremental Rocchio filtering algorithm (InRoute) presented at TREC-6. This year the system used more complex initial queries (similar to adhoc) and a *high precision* setting for the filtering threshold, resulting in a conservative document selection strategy.

NTT Data participated in batch filtering and routing. Their system uses Rocchio relevance feedback and dynamic feedback optimization based on terms and term pairs, following the model of the AT&T TREC-6 routing/filtering system. In addition, they extracted conjunctions of more than two terms by running a classification tree algorithm (C4.5) over the training data to find larger groups of terms strongly associated with relevance. No information about threshold setting for batch filtering is provided in the notebook paper.

CUNY participated in all three subtasks, although only the adaptive filtering experiments are described in the notebook paper. They adapted the profiles every 10,000 documents filtered using a probabilistic term reweighting scheme. No query expansion was performed. Retrieval thresholds are allowed to vary between a fixed upper and lower bound. The thresholds are set initially to the upper bound, meaning that the system is starting with a conservative document selection strategy. Thesholds are adjusted every 10,000 documents as a function of three parameters: number of documents which have passed through the system, number of documents examined, and number of documents relevant.

TNO participated in adaptive filtering. They used an adaptive Rocchio relevance feedback algorithm with the profile being built from the last N relevant and the last N non-relevant documents. Each profile starts with a fixed initial threshold. The threshold is adjusted to be the mid-point between the average similarity of the last N relevant and the average similarity of the last N non-relevant documents. The difference between the two submitted runs is the choice of the initial threshold (0.35 vs. 0.40). There is no tuning of thresholds to the utility functions.

3.2 Evaluation results

Figures 1-10 summarize the evaluation results for the TREC-7 filtering track. There are two types of graphs. Figures 1, 4, and 7-10 use average rank as the performance measure, and represent 2-3 different experimental runs on the same plot. Figures 2-3 and 5-6 use average scaled utility as the performance measure, and show a range of different values for the lower bound. In all plots, the run labelled *Baseline* is computed by retrieving no documents for every topic (i.e. a utility score of zero for all topics). The average rank plots have dashed lines connecting the same system over the different experiments. The vertical arrows link all the systems which are not distinguishable from the top performing system according to the statistical significance tests described in the previous section. The longer arrow always corresponds to the more conservative test. In the average scaled utility plots, the horizontal axis represents the number of non-relevant documents used to define the lower bound utility, plotted on a log scale. The vertical axis represents the difference in average scaled utility between each system and the baseline. Therefore, the baseline is represented by the straight horizontal axis marked with index points for the lower bound.

Figure 1 compares average rank performance for F1 and F3 utility for the adaptive filtering subtask. The system rankings don't change a lot, although the baseline drops substantially. This plot demonstrates that the F1 utility function sets an extremely challenging target, since most groups fall well below the baseline. Only CLARITafF1b comes close with a strategy that shuts down the filtering profiles which have accumulated a long-term negative utility. Several systems (TNO and IAHKaf12) seem to track the baseline more closely, generally because these systems have small or empty retrieved sets for many topics.

Figure 2 presents average scaled utility for the F1 function and the adaptive filtering subtask. Systems can be divided into two categories, TNO and IAHKaf11 which perform at a relatively constant level regardless of the lower bound, and the rest of the systems, which improve substantially as the number of non-relevant documents s used to define the lower bound is increased. CLARITafF1b clearly dominates the latter group of systems, entirely on the strength of its decision to shut down the profiles which had accumulated a negative utility. Figure 3 presents average scaled utility for the F3 function and the adaptive filtering subtask. In this case, most systems track the baseline regardless of lower bound, with the notable exception of IAHKaf32. For the F3 utility

	AntRout1	AntRout2	arc98cs	nttd7rt1	att98fr4	att98fr5	MerRou	pirc8R1	pirc8R2
≤4 AP88 rel	0.314	0.422	0.166	0.512	0.010	0.352	0.218	0.087	0.048
>4 AP88 rel	0.247	0.248	0.086	0.515	0.342	0.436	0.318	0.317	0.433

Table 3: Breakdown of routing performance for topics with few relevant documents on AP88.

function, average scaled utility seems like a reasonably good evaluation measure, since there is minimal dependence on the choice of lower bound.

Figure 4 compares average average rank performance for F1 and F3 utility for the batch filtering subtask. It appears that both NTT Data and CLARITECH have more success with the F3 utility function relative to the other participants. Figures 5 and 6 present average scaled utility for the F1 and F3 function respectively and the batch filtering subtask. For F1 utility, the CLARIT system performs much better with a more generous lower bound. This is largely a function of its decision to start with a delivery ratio threshold rather than a threshold estimated from the training data. This means that they retrieved non-relevant documents for all of the hard topics where it was better off to retrieve nothing and thus their average performance suffers when the lower bound emphasizes doing well on hard topics. In both plots, the systems converge more towards the baseline as the lower bound is decreased. This is a natural progression because the majority of batch filtering systems are doing a lot better than the baseline on most topics, but this advantage is reduced as the lower bound drops and the performance of a utility score of zero is more highly rewarded.

Figure 7 compares performance on the routing task between the raw average uninterpolated precision scores on the left axis and the average ranked precision on the right axis (rescaled to facilitate comparison). The two approaches to evaluation yield more or less the same results. Note that routing systems were allowed to use any existing relevance judgement for topics 1-50 outside the test collection. It is clear that not all systems took advantage of that extra data. Some systems appear to have used only AP88 training data for routing, since this is what was required for batch filtering. If we measure performance separately on the 10 topics with 4 or few relevant documents in AP88 (29-37 and 39) versus the rest, we note that systems fall into two categories (see Table 3). The ones on the left probably used all the training data, and do at least as well on the subset with few AP88 training examples. The systems on the right see their performance drop, often drastically. There is good reason to believe that some of these systems could improve their performance by using the full set of training documents.

Figures 8 and 9 plot average rank performance by year for F1 and F3 utility respectively and the adaptive filtering subtask. For F1 utility, the relative performance of systems changes substantially between 1988 and 1989. To a large extent, this reflects the decision of some systems (like CLARIT and ok7ff) to lower the thresholds early to gather more training data in the hopes of improving performance later. For F3 utility, many systems are bunched together around the baseline in 1988, and then they spread apart as some systems manage to take much better advantage of the arriving training data than others. This may reflect the fact that some systems are retrieving more documents in 1988 than others even though their performance in terms of utility is about the same. It is interesting to note that the INQ51x had some difficulty setting their initial thresholds for both utility functions (retrieving far too many documents in 1988) but recovered in 1989-1990 to perform very respectably. This is in spite of the fact that they thought they had chosen a conservative threshold setting strategy!

Figure 10 compares the pooling and the sampling strategy for evaluation. The results presented here are limited to the 14 topics where significant sampling took place. For all other topics, the retrieved sets are almost always less than 100 documents, so the pooled and the sampled results are virtually identical. Since the score differences over these 14 topics are also small, it is fair to conclude that pooling introduces minimal bias in favor of any individual system. However, a single run, sigmaF1, does improve when evaluated by sampling, indicating that this system may be behaving differently from the others. The most extreme example of this pattern is topic 22, where sigma received a sampled score of 2197 and a pooled score of -844! The 95% confidence interval is plus or minus 930 documents so there is a lot of room for variation. However, it is likely there are still a fair number of unevaluated relevant documents for this topic which sigma was finding but not getting credit for. There are several other topics where sigma's pooled score is below the 95% confidence interval for its sampled score, indicating that it might have received unfair treatment from the pooling process.

4 General Commentary

The first, most obvious conclusion is that adaptive filtering is a much harder task than batch filtering. This is immediately evident when one looks at where the baseline ends up in Figures 1 and 4. This should come as no surprise since batch systems were able to take advantage of training sets consisting of hundreds of judged documents. In fact, we might consider whether the F1 utility function should be changed to reflect the fact that it is an unrealistically tough standard for adaptive filtering systems. Only one system managed to come within striking distance of the baseline (retrieving no documents!) and only by shutting down poorly performing profiles before the 1990 data were evaluated.

It is also clear that adaptive filtering is a very different task than batch filtering, requiring a much more careful approach to threshold selection. In batch filtering, threshold selection is important, but it is also crucial to learn a high quality initial profile based on the training data. In previous TREC filtering experiments, the same systems appear at the top of the rankings for both batch filtering and routing, indicating that a good ranking algorithm is very important. In order to build a good ranking algorithm, the system must learn effectively from the training data, and most successful systems used powerful statistical learning techniques. This is no longer necessary (or even possible) in adaptive filtering. In fact, the best performing systems from CLARITECH and CitySM did very little profile learning, using minimal (or no) query expansion and limited term reweighting. In fact, ok7ffx2 performed no profile adjustment at all and was still highly competitive with the best performing systems! All of its effort was directed towards adaptive threshold selection.

In adaptive filtering, it is important to recognize when a filtering profile is retrieving lots of non-relevant documents and adjust the threshold rapidly in response. CLARITafF3b took the extreme step of shutting down some profiles entirely and performed much better than the corresponding run which continued to retrieve documents. At TREC-7, most systems filtered a fixed size batch of documents before updating profiles. CLARITECH experimented with increasing size batches and found that this method works better, however they still used the same batch size for all topics. There are obvious efficiency advantages to updating all topic profiles simultaneously, which is a good reason to prefer this approach. However, the initial observation suggests that it might be better to update profiles as a function of documents retrieved rather than documents filtered. This makes intuitive sense, since a profile which has retrieved only a few documents is not going to change much whether 100 or 10,000 documents have been filtered. This will also allow the system to quickly adjust a profile that is going far off course and retrieving way too many documents as a result of a change in the distribution of arriving documents. However, variable topic updating would require a more complex filtering architecture.

In TREC filtering experiments, it is very important to adapt the threshold setting mechanism to reflect the utility functions used in evaluation. Several systems do not take this step, and their performance suffers accordingly. When system performance is evaluated using other measures, such as precision and recall, these systems are often quite competitive. Therefore, it is unlikely that these systems would perform badly in a practical setting, since the utility functions are constructed artificially for our simulation experiments. However, it is important to have a threshold tuning mechanism which can respond to the user's needs.

There are still many interesting research topics to pursue, particularly in the area of adaptive filtering. For example, no one has yet explored whether the distribution of a feature over time is related to its usefulness as a discriminator for relevance. The filtering track will continue for TREC-8, with much the same breakdown of tasks as we have seen for TREC-7. We want to continue to encourage anyone with interest in the topic to consider participating in the filtering track next year. The track is designed to be flexible to allow groups with very different research interests to participate, and the barriers to entry are low. Participating groups will play a strong role in determining the direction of the TREC-8 filtering track.

Acknowledgements Many people have contributed to the development of the TREC filtering track, in particular David Lewis, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, Stephen Robertson, the TREC program committee, and the team at NIST. I am merely building upon their work.

References

[1] David A. Hull and Paul B. Kantor. Advanced Approaches to the Statistical Analysis of TREC Information Retrieval Experiments. Contact the first author for a copy: hull@xrce.xerox.com, 1997.

[2] David Lewis. The TREC-5 Filtering Track. In *The 5th Text Retrieval Conference (TREC-5), NIST SP 500-238*, pages 75–96, 1997.

[3] David D. Lewis. Evaluating and Optimizing Autonomous Text Classification Systems. In *Proc. of the 18th ACM/SIGIR Conference*, pages 246–254, 1995.

[4] Ellen M. Voorhees and Donna Harman. Overview of the 6th Text Retrieval Conference (TREC-6). In *The Sixth Text Retrieval Conference (TREC-6), NIST SP 500-240*, pages 1–24, 1998.

[5] Ellen M. Voorhees and Donna Harman, editors. *The 7th Text Retrieval Conference (TREC-7)*, 1999. To appear.