The TREC-6 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 details the experiments conducted in the TREC-6 filtering track. The filtering track is an extension of the routing track which adds time sequencing of the document stream and set-based evaluation strategies which simulate immediate distribution of the retrieved documents. It also introduces an adaptive filtering subtrack which is designed to simulate on-line or sequential filtering of documents. 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 attempts to analyze and explain the observed patterns. The final section suggests some ways to extend the current research in future experiments.

1 Introduction

There is increasing evidence that text filtering will become a critical tool in searching and managing the flow of data in the information age. New companies are appearing daily which offer push services or intelligent agents centered around the core technology of content-based filtering. Since the beginning of TREC, the routing task has served as a forum for the development of these algorithms. However, as the uses of this technology have changed over the intervening years, the routing problem has gradually diverged from its most common applications. The filtering track was created to provide a more realistic simulation of the on-line time-critical applications of this technology.

We will start by describing the routing task as defined in TREC and use this definition as the starting point for the filtering task. Here is a simple definition of the routing task:

> Given a topic description and a large collection of documents, a sample of which have been evaluated as relevant or not relevant for that topic, construct a query profile and a routing function which will score and rank new documents according to their likelihood of relevance.

Filtering builds upon the routing task by introducing the concept of time. The routing task has traditionally been evaluated by average precision curves and other rank-based measures which use the complete ranked list 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 these tasks are based is one where a stream of transient documents are compared to a fixed set of query profiles. This means that the routing task simulates a non-interactive process where a user looks at documents only once at the end. A more realistic situation is one where 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, the filtering track operates on the opposite end of the spectrum, assuming that the user wants to be notified about each potentially interesting document immediately after it arrives. This leads to the following definition of the filtering task:

> Given a topic description, an incoming document stream, and possibly a small historical database of documents which have been evaluated as relevant or not relevant for that topic, construct a query profile and a filtering function which will make a binary decision to either accept or reject each new document as it arrives.

Since a decision to accept or reject a document must be independent of subsequently arriving documents, evaluation by rank-based measures is not appropriate. Filtering results will consist of unordered sets of documents which will be analyzed using set-based evaluation measures.

More background on the motivation for the filtering track and some scenarios for its use can be found in the TREC-5 filtering track description [2]. We recognize that there is still a considerable gap between the TREC filtering experiments and what goes on in operational systems. In particular, the task is fairly general, and their is no user interaction in the evaluation process. There is also no control of system performance in the area of efficiency (documents filtered per second) and scalability. This is because there are certain unavoidable trade-offs that must be made to make the filtering track compatible with the standard TREC experimental framework. However, there are many advantages to the TREC approach, the foremost of which is strong quantitative evaluation of system effectiveness.

2 TREC-6 Task Description

The TREC-6 filtering task has been substantially revised from the track definition and the experiments conducted in previous years, but the goals remain basically the same. The basic goals of this year's track are: to move towards a more realistic filtering task while retaining strong quantitative evaluation and to broaden the range of possible filtering experiments. The routing task has been criticized as unrealistic for a number of reasons. First, average precision and the other routing evaluation measures assume a post-hoc evaluation conducted over the entire test document collection. Second, while filtering has always been thought of as a temporal process, no time ordering or date information has been associated with the documents. Third, the training documents have come from a variety of different collections, most or all of which do not correspond to the source of the new test documents. Fourth, the size and coverage of the training set is much greater than one would find in most realistic applications. Fifth, the training documents have come from a variety of different systems, almost all of which are different from the filtering systems that will be tested. Of these five points, only the first has been addressed in previous iterations of the filtering track. The second and third point were addressed this year by the selection of the training and test documents. The fourth and fifth points are addressed in the adaptive filtering subtrack defined later in this section.

2.1 Topics and Documents

The corpus for the TREC-6 filtering experiments comes from the Foreign Broadcast Information Service (FBIS), which selects (and translates) text documents or transcripts from various non-American broadcast and print publications [4]. The 130,000 training documents date mostly from 1993 and early 1994 while the 120,000 test documents date come from late 1994 and 1995. All documents have date stamps attached and have been ordered according to their date. The date stamps represent (more or less) the release date of the FBIS version rather than the release date of the original source document. Due to practical constraints (the first part of the data was released last year), the documents are not cleanly separated by date. Many test documents predate some of the training documents and some test documents predate most of the training documents. All systems were required to filter test documents in date order, allowing researchers to make use of the dimension of time. In addition, both the training and the test documents come from the same source, which is viewed as a more realistic simuulation of most filtering applications.

There were 47 topics for the TREC-6 filtering experiments, which are listed here:

```
  1   3   4   5   6  11  12  23  24  44  54  58  62  77  78  82  94  95
100 108 111 114 118 119 123 125 126 128 142 148 154 161 173 180 185 187
189 192 194 202 228 240 282 10001 10002 10003 10004
```

Of these topics, 38 were used in last year's routing/filtering experiments, and 9 of the topics are new this year (listed here):

```
62 128 148 180 282 10001 10002 10003 10004
```

The topics 10001-10004 were built specially for the routing/filtering task this year. The 9 new topics have incomplete relevance judgements on the FBIS training data. In particular, only the top 100 or so documents retrieved according to the NIST ZPRISE system were judged for relevance. Therefore, there are two different types of topics in terms of the relevance judgements. The 38 old topics have hundreds of relevance judgements on documents retrieved from many different sources while the 9 new topics have fewer than 100 relevance judgements on documents all retrieved by the same source. We will explore the impact of this dichotomy on the performance of different systems. The same topics were also used for the routing task. Note however, that routing systems are allowed to use relevance judgements for these topics from any of the TREC subcollections while filtering systems are limited to the FBIS training documents only. This represents an important divergence between the routing and filtering experiments which is happening for the first time this year.

2.2 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. Instead, we apply two set-based evaluation metrics, utility and average set precision (ASP).

2.2.1 Utility and Average Set Precision

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-6, we test two different settings of the utility parameters:

```
F1 = 3*R+ - 2*N+         --> retrieve if P(rel) > .4
F2 = 3*R+ - N+ - R-      --> retrieve if P(rel) > .2
```

Filtering according to a utility function is equivalent to filtering by estimated probability of relevance. Therefore, the description above also shows the appropriate probability thresholds which correspond to the utility functions. 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.

Average set precision (ASP) is defined as the product of precision and recall, i.e. ASP = precision * recall. One can also think of ASP as a variant of average uninterpolated precision. Average uninterpolated precision computes the precision at the position of each relevant document in the ranked list, takes the sum, and divides by the total number of relevant documents. Average set precision is calculated on unordered document sets and uses the precision of the entire retrieved in place of the precision at each rank position. Otherwise, the calculation is the same, as this number is multiplied by the number of relevant documents retrieved and divided by the total number of relevant documents.

Utility is not an ideal measure for judging the performance of filtering systems for a number of reasons. First, utility scores will vary widely from topic to topic based on the number of relevant documents, and there is no valid way to normalize them, meaning that the scores cannot easily be averaged or compared across topics. Second, utility treats all relevant documents as equally important, no matter how many have already been retrieved or how many exist for a given topic. There is no sense of diminishing returns, which seems counter-intuitive in many situations. Third, a user would probably have a different utility function for each topic, based on the topic's difficulty and the number of relevant documents. For practical and administrative reasons, it is too difficult to define a separate utility function for each topic.

However, ASP also has its problems. In particular, when no relevant documents are returned (and relevant documents exist), a system receives a score of zero, irrespective of the size of the retrieved set. This means that retrieving no documents is equivalent to retrieving an arbitrary number of non-relevant documents. Clearly, the former result is much preferred over the latter, and a major part of the challenge in text filtering is knowing when not to retrieve any documents. It should be noted that van Rijsbergen's E-measure also suffers from this problem. So far, no one has developed a good general measure of set-based retrieval effectiveness that does not suffer from at least some of these drawbacks. A better approach for the future may be to select a particular task and a particular user model, then define a model- and task-specific measure.

2.2.2 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-6 experiments because the 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 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 ([2], p.87, eq 30):

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

Unfortunately, sampling cannot provide us with an estimate of recall, which is necessary for F2 Utility and ASP, so these measures can only be estimated via pooling. 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 Basic Task Summary

Each participating group could submit up to two filtering runs for each of the three evaluation measures: F1 Utility, F2 Utility, and ASP. Participating groups were required to submit at least one F1 Utility run and at least one F2 Utility run. 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 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.

In practice, only one manual run was submitted and no one submitted any runs based on manual feedback. Previous TREC routing and filtering experiments have found no consistent advantage to using manual techniques for profile construction, so all submitted runs were treated as a single category. For most topics, there are enough evaluated documents already that the cost (in terms of human effort) of additional manual assessment would not be justified by the returns. In addition to the three evaluation measures, there were two optional subtasks that groups could participate in, as will be described in the next two subsections.

Due to assessment constraints, NIST promised to evaluate a maximum of 100 documents per topic per participating group. These documents were obtained as follows. Only the mandatory runs contributed to the assessment pool. If the F1 Utility set for topic T is greater than 100 documents, 100 documents are sampled for assessment. Otherwise, all documents are assessed. Furthermore, a sufficient number of document from the F2 Utility set are sampled to bring the total set size up to 100 documents. If the union of the F1 Utility and F2 Utility sets is less than 100 documents and the group submitted a ranked retrieval run (see the next subsection), then the sampling program went down the list in rank order adding new documents until a sample of size 100 was extracted. This final step was added to improve the quality of the document pool as much as possible given the assessment constraints.

2.3.1 Comparison to Ranked Retrieval

One can view the routing task as a subset of the filtering task, where routing consists of defining a document scoring function which tends to rank relevant documents above non-relevant ones, while filtering consists of building a thresholding function on top of the scoring function to optimize some set-based evaluation criterion. In the past, most systems participating in filtering have also participated in routing and used the same scoring function for both tasks. For systems which follow this model, we can try to some extent to separate the performance of the thresholding function from the performance of the scoring function. Participation in this subtask is optional, since there is no a priori reason why a system must use a scoring algorithm in their filtering system. One could use a binary rule-based algorithm instead. In addition, more complex systems may change the scoring function as they view test documents, meaning that scores generated at the beginning of the document stream may not be comparable with scores generated later on in the document stream.

For TREC-6, we try to learn about this issue by applying two different simple tests. For each test, we take the ranked list of documents returned by the scoring function and calculate the score obtained by selecting the optimal threshold. This is computed by exhaustive search over the ranked list of relevant documents. In the first test, we ask the simple question, did the system overestimate or underestimate the threshold? By accumulating this information over the full topic set, we gain a general sense about whether the system is biased towards retrieving too many or too few documents. In the second test, we compare performance based on the optimal thresholds to the observed performance to determine whether this changes the relative ranking of participating systems.

2.3.2 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 that they have been responsible for retrieving. The adaptive filtering subtrack 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 the training data and the previously-released 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 evaluate 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 filtering a more interesting and more complex task. Since many of this year's topics have only been partially evaluated over the training document collection, they are not suitable for adaptive filtering. Therefore, this subtask will use only the 38 TREC-5 topics which have been fully evaluated. Systems may choose whether they wish to use the rest of the TREC document collection (excluding the FBIS training document set) to generate collection frequency statistics (such as IDF) or auxilliary data structures (such as automatically-generated thesauri) or begin with no prior information. Systems may also decide whether they wish to treat unevaluated training documents as not relevant or whether they assume that the user simply declined to make a judgement in this situation. Performance will be judged on both the training and the test document set, and the results will not be compared directly to the other filtering tasks. This is because adaptive filtering runs will operate at a substantial disadvantage, since they only have access to relevance judgements from documents which they retrieve. Performance on the training set must also count to prevent groups from retrieving far too many documents on the training set to increase the pool of relevance judgements available to the system and thus improve performance on the test documents. Since this task is in an experimental stage, participants are also free to choose from among the evaluation measures.

3 TREC-6 results

The TREC-6 filtering track had 10 participating groups who submitted a total of 59 runs, divided as follows:

	# groups	# runs
Total	10	59
-----	--	--
F1	10	17
F2	10	17
ASP	7	15
ranked	7	11
adaptive	1	2

Note that the sum of the runs column adds up to more than the total number of runs because some groups submitted the same run for more than one evaluation measure. The participating groups [abbreviations] (run identifiers) were AT&T Labs Research [AT&T] (att97f), Australian National University [ANU] (anu6flt), City University London [City] (city6f), CLARITECH Corporation [CLARITECH] (CLRoute/CLComm), Daimler Benz AG Research Center Ulm [Daimler Benz] (dbulm1), Queens College CUNY [CUNY] (pircsF), Siemens AG [Siemens] (teklis6), University of

California Berkeley [Berkeley] (BKYT6f), University of Massachusetts Amherst [UMass] (INQ4), University of North Carolina [UNC] (isf).

3.1 Summary of approaches

In this section, we briefly describe the techniques used by each of the groups for the filtering track[1]. For more information, please consult the individual participants' papers included in this volume [5]. Almost all participants treat filtering as a special case of routing, using a ranked retrieval system, followed by thresholding based on the document score. The vast majority of systems use the same basic technique for finding the threshold: observe the score on the training set where the evaluation measure is maximized and use that score as the threshold for the test set. The exceptions will be noted below. Several system also use logistic regression to convert the scores to probability estimates and filter on the probability estimates directly. The major differences between systems come from their scoring algorithms.

AT&T builds a feature set of terms, phrases (adjacent pairs), and non-adjacent pairs based on the term weights from Rocchio expansion. It then optimizes the feature weights for average uninterpolated precision using a technique called Dynamic Feedback Optimization (DFO). Their experimental run (att97fe) takes the lowest scoring half of the relevant documents and the highest scoring non-relevant documents based on the initial query and constructs a second query via the same methods. The final routing profile is a weighted combination of these two queries.

ANU selects terms and phrases (adjacent pairs) from the topics and from the training documents using independent algorithms. The topic selection algorithm assigns higher weight to title words, words with high frequency, and words which are all in capitals. The document selection algorithm scores terms based on the difference between their probability of occurrence in the relevant documents and their probability of occurrence in the non-relevant documents. The number of features and the relative weight of the feature set returned by each algorithm are then passed on to a Generalized Reduced Gradient (GRG2) nonlinear optimization program, which optimizes these parameters with respect to the utility measures.

City orders terms and adjacent pairs based on their weight according to a probabilistic model, then applies forward stepwise feature selection using a fixed increase in average uninterpolated precision as the selection criterion. Term weights are then optimized using two methods: deterministic adhoc weight adjustment and a simulated annealing procedure. In each case, the training set is partitioned into two halves; the term selection is based on one half while the weight optimization is based on the other half. The partitions are then inverted and the procedure is repeated. Additional repetitions are generated by splitting the training set randomly into different partitions. All the resulting term sets are merged to build the final query profile. The difference between their two runs is that city6f1 optimizes weights on half the database and selects thresholds on the other half while city6f2 optimizes weights on both halves and selects thresholds on the entire database.

CLARITECH uses a probabilistic model in the first pass for term selection followed by a second pass of the Rocchio algorithm for term selection and term weighting. Thresholds are chosen via logistic regression for F1 Utility and using the optimal score on the training documents for the other measures. CLARITECH tested both threshold selection methods on the training set and then picked the one that worked best for each measure. CLROUTE is based on this method while CLCOMM divided the training data into two parts and applied the terms selection algorithms to each part separately. Only the terms that appeared in both sections were retained for the final profile.

[1] This section is based on the material provided by the groups in their draft papers.

Daimler Benz generates 5 different feature sets: 2 based on n-grams (3- and 4-grams), 3 based on substrings extracted from terms (a stemming variant) and selected for using the following measures: chi-square, TF-IDF, and correlation (with relevant documents?). Each feature set is transformed and reduced in dimension (to 600 for filtering) by taking an eigen decomposition of the feature covariance matrix (Principal Component Analysis - equivalent to Latent Semantic Indexing). Filtering is treated as a 47-class categorization problem (each topic is one category) and a single classification rule is built for all categories simultaneously using linear regression. The final profile is created by merging the decision vectors constructed for each feature set.

CUNY combines two runs, one based on terms (and term pairs?) extracted from the topics, and one based on terms (and term pairs?) extracted from the training documents, in their submission. The only difference between the two submissions is that one of them lowers the threshold by 10% to reflect the fact that subsequent relevant documents are likely to be less similar than training documents which are used both for profile creation and for threshold selection.

Siemens uses a true filtering system which computes document scores based on term correlations and a probabilsitic model. The feature set is single terms only and the topic statement is not used.

Berkeley first performs term selection using a chi-square measure (positive association with relevance only) with a significance cut-off of 0.001. These terms are put into an expanded query which is then scored against the training documents. In addition, terms are ranked according to their log-odds of relevance and the top 5 terms are selected. The RSV scores from the first step and the log term frequencies of the 5 terms extracted in the second step are passed as parameters for logistic regression, which produces the final probabilistic scoring function.

UMass builds an initial query based on the topic statement and extracts the top scoring 200 word passage from each evaluated training document. The feature set consists of the top 20 each of terms, phrases, and pairs in a 20-word window, according to the difference between probability of occurrence in relevant documents and probability of occurrence in non-relevant documents. These features are weighted and the weights are optimized for average uninterpolated precision using Dynamic Feedback Optimization. For adaptive filtering, UMass uses a true filtering system which builds a profile based on the initial topic statement and updates it with an incremental Rocchio algorithm. Thesholds are set at the mid-point between the average score on the relevant documents and the average score on the non-relevant documents.

UNC creates an "information space" for each topic by applying Principal Components Analysis (an eigen decomposition) to a co-occurrence matrix constructed from all query terms. Training documents with at least 25% of the query terms are located in this space. The threshold for F1 Utility was chosen to be the minimum distance between any evaluated document and the topic statement in the information space. Any document with at least 25% of the query terms was retrieved for the runs based on F2 Utility.

3.2 Comparative Evaluation

When evaluation is based on utility measures, 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, comparative evaluation will be based on an average rank measure which treats all topics equally. This measure is computed in two steps: (1) for each topic, rank the systems in order of their performance, (2) average the ranks by system over all topics. This means 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 average rank measure has its advantages and disadvantages. On the positive side, all

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.

topics are equally important in determining a system's performance, meaning that the scores are insensitive to outliers or topics which may have high variation due to random factors. 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 allows one to do a global comparative evaluation 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, the scores are only meaningful relative to the performance of different systems. The results depend on the systems being compared, so adding or removing a system will change the results of other systems.

Average Set Precision (ASP) is comparable across topics, and will be evaluated both using the traditional average score and the average rank measure. As mentioned previously, ASP does not distinguish between systems which retrieve no relevant documents. However, using ranks allows us to introduce a tie-breaking procedure. Systems which have ASP = 0.0 for a given topic will be ranked in inverse order of the number of documents which they retrieve.

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 limit ourselves to 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 is an order of magnitude smaller.

3.3 Evaluation results

Figures 1-8 present the evaluation results for the TREC-6 filtering experiments. Each figure contains two or more experimental runs drawn on the same page. Dashed lines are drawn to link the runs from the same system. In most cases (but not always), these will correspond to the same underlying retrieval algorithm with a different threshold or thresholding strategy. 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.

We should mention that two groups had problems with their submitted runs. CUNY (pircs) optimized for the wrong utility thresholds, so their F1 and F2 Utility runs are not accurate. Daimler Benz (dbulm) did not understand how sampling would be used for evaluation and artificially truncated all of their runs to 100 documents. This will have a much higher impact on F2 Utility and ASP than on F1 Utility. Both groups were given the opportunity to submit revised results, and a revised (unofficial) comparison is presented in Figure 8.

Figure 1 compares system performance for F1 and F2 Utility. The system rankings remain more or less the same, with a few small shifts. The results indicate that there is not a lot of interaction between system performance and filtering threshold. Figure 2 compares system performance over the two utility measures and average set precision. Other than the two systems with errors in their runs (pircs and dbulm), there are not a lot of differences. We do notice that UMass (INQ) and Siemens (teklis) do better on the utility measures than ASP. These systems retrieved a uniformly small number of documents for all topics. Small retrieved sets are often good for the utility measures but rarely good for ASP.

Figure 3 compares evaluation based on average score to evaluation based on average rank for ASP. The average rank scores are rescaled to match the end points of the average precision scores. We find that both measures yield basically the same results. Figure 4 compares the pooling versus the sampling strategy for evaluation. Note that the results are presented only for the 12 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 12 topics are also minimal, it is fair to conclude that pooling introduces no bias in favor of any individual system. In most cases, the pooled scores fall within the 95% confidence intervals for the sampled results. There are three topics (142, 189, and 202) for which the pooled scores fall outside the sampling confidence bounds with some regularity. In all these cases, the pooled estimate is significantly lower than the sampled estimate, indicating that there are probably still a fair number of unevaluated relevant documents for these topics.

Along with their filtering runs, groups were also allowed to submit a ranked list of documents which would correpond to the output of their filtering system without thresholding. By comparing this ranked list to the evaluation functions, we can determine the optimal threshold in a retrospective fashion. Figure 5 compares observed performance to performance based on choosing the optimal threshold for F1 Utility. Once again, we note very little difference between the two graphs. This should not be surprising since almost all participants used more or less the same strategy for threshold selection. CLARITECH did a direct comparison of the two most common strategies for threshold selection: choosing the retrieval score which optimizes performance on the training set and logistic regression. For CLRoute, they found that logistic regression worked better for F1 Utility while optimal score worked better for F2 Utility. For CLComm, they found that the two worked about the same. A more detailed analysis of threshold bias is presented in the next section.

The TREC-6 filtering topics can be divided into two categories: the 38 old topics which were heavily evaluated over the training FBIS data at TREC-5 and the 9 new topics, where evaluation over the training data is limited to roughly 100 documents retrieved by the PRISE system. Figure 6 breaks down the average rank scores into the old and new topic sets for F1 Utility. Performance remains roughly constant for most systems (ignoring the pircs runs which have errors). However, the cityf2 run jumps upward by more than one rank position. While the topic sample is too small to say that the difference is statistically significant, one can speculate that the approach adopted by City might have some advantage for small training sets. While cityf1 does not show the same jump, this result can be easily explained. cityf1 uses only half the training data for optimization, reserving the other half for threshold selection. With smaller training sets, one might expect that ignoring half the data would offset any other advantages inherent to the technique.

It is interesting to examine whether there is any interaction between system performance and the inherent difficulty of the topics. For this experiment, we defined difficulty according to the median F1 utility score for the topic. All topics with a median utility greater than zero were classified as easy and all topics with a median utility less than zero were classified as hard. This resulted in 21 easy and 17 hard topics (the remainder had median utility zero). Figure 7 breaks down performance as a function of easy and hard topics. We note immediately that the systems which retrieved few documents for all topics do much better on the hard topics than the easy topics. This reflects the fact that for the hard topics, it is generally correct to retrieve very few documents. The opposite pattern emerges for three of the four top-ranked systems. These systems are getting almost all of their gain on the easy topics and are performing significantly worse than most of the remaining systems on the hard topics. This means that these systems (cityf1, cityf2, att97fc) need to work on being more robust for hard topics. The only exception is att97fe, which performs well for all topic categories. We should note that only 2 of the 9 new topics are hard, meaning that the size of the training set is not strongly linked to topic difficulty.

Figure 8 shows the revised comparisons after Daimler Benz (dbulm) and CUNY (pircs) submitted corrected results. It is reassuring to note that the relative positions of all the other systems remain roughly the same.

3.4 Absolute Performance and Threshold Bias

The graphs presented in the previous section are missing a key component. They tell us about the relative performance of systems but say nothing about performance according to an absolute standard. It is hard to decide on a standard which is appropriate for all topics, so we fall back on a very simple approach for the F1 utility measure. A system which retrieves no documents at all receives an F1 utility score of zero. Therefore, any system which has a positive utility score for a given topic is providing some added value. This may seem like a fairly minimal standard, but it is sobering to see how hard it is in practice to measure up to this standard. The median TREC-6 filtering system has positive utility for 20 topics, zero utility for 11 topics, and negative utility for 16 topics. The best TREC-6 filtering system has positive utility for 33 topics. This means that the typical TREC-6 filtering system can barely justify its own existence. The same question is more complex for F2 utility, since systems are penalized for unretrieved relevant documents, so retrieving no documents leads to a negative score. However, we can compute how many systems have a utility score which is greater than what they would receive by retrieving no documents. The median system does better than the empty document set on 42 topics and worse on 5 topics, while the best system is better on 46 topics and worse on 1 topic. With respect to ASP, it is always better to retrieve at least some documents.

When we look at the actual utility scores, we are left with the impression that filtering systems in general are performing quite poorly. For example, the typical system has a positive F1 utility for slightly over half the topics and positive F2 utility for only 10 of 47 topics. There are two possible explanations for these results: (1) threshold selection really is a very hard problem, and/or (2) the filtering thresholds (and/or the measure itself) are simply an unrealistic standard of performance. We can get at this question by looking at the utility scores obtained by using the optimal thresholds. The optimal performance of the typical system results in positive F1 utility for about 35 topics and positive F2 utility for about 16 topics. The optimal performance of the best systems results in positive F1 utility for about 40 topics and positive F2 utility for about 30 topics. From these results, it seems that both factors are important. A system must be performing very well to return positive utility for the majority of topics, particularly for the F2 measure. Penalizing for unretrieved relevant documents is setting a very high standard. Perhaps it might be better to penalize only for retrieved non-relevant documents. On the other hand, there is clearly still a lot of room for improvement in threshold selection algorithms.

We can also use the optimal thresholds derived from ranked retrieval to determine if there is bias in the threshold selection algorithms. For each run, we simply ask whether the observed threshold is lower, higher, or the same as the optimal threshold and count up the number of topics which fall into each category. Table 2 presents the results for the median TREC-6 filtering system. We

Median	too few	exact	too many
F1	21	5	20
F2	28	2	17
ASP	37	1	9

Table 2: Number of topics where systems retrieved too few, the right number of, or too many documents

note immediately that there is very little bias for F1 Utility, but systems have a strong tendency to retrieve too few documents for F2 Utility and ASP. We can only speculate as to why this might be the case. The training set is not comprehensively evaluated so there will always be missing relevant documents. If systems underestimate the density of relevant documents on the training set, they are likely to do the same on the test set, which may lead them to select thresholds which are too restrictive. Most systems use the training documents both for building the filtering profile and for threshold selection. Therefore, the scores of relevant training documents will be biased upwards and this bias may be passed on to the threshold selected by those systems. For example, CLComm attempts to reduce this bias by splitting the training data into two parts and merging profiles built on both parts independently, while CLRoute does not. The results confirm that CLComm has less threshold bias than CLRoute for both F2 Utility and ASP.

4 General Commentary

Almost all groups attack the filtering problem by building a good ranking algorithm and then applying thresholding. This is partly because people are using the same systems for both routing and filtering. However, the best scoring systems optimize the ranking for some completely independent measure like average uninterpolated precision and still have a great deal of success on the filtering track. There does not seem to be any need to optimize the ranking specifically for the filtering evaluation measures. The key question is then how to set the filtering threshold.

The systems concentrate on two different threshold selection strategies: finding the score which optimizes performance on the training set or normalizing the scores to create probabilities using logistic regression and filtering on probability thresholds. We'll briefly describe the advantages and disadvantages of each approach. One can think of each method as looking at a plot of utility as a function of retrieval score on the training data and picking the maximum. The difference is that logistic regression smooths the curve before finding the maximum, which can be a significant advantage. Empirical estimation can suffer when the curve is extremely bumpy since the observed maximum may be only an artifact caused by an unusual clumping of relevant documents. Also, consider the case where the training data is poor (low density of relevant documents). The desired threshold score may be greater than the observed score of any of the relevant documents. Logistic regression allows the system to extrapolate a threshold which is greater than the score of the top-ranked document. Empirical methods provide no direct way to estimate a threshold outside the range of the observed data. On the other hand, logistic regression is fitting a parametric curve to data which may not have the logit distribution. This can result in biased estimates in some cases. CLARITECH demonstrates that logistic regression doesn't always work better than empirical methods.

There are alternative methods for smoothing the utility curve which have not yet been explored. Instead of logistic regression, one could adopt a very simple local smoothing algorithm. For example, one could calculate and average utility over a sliding window of n rank positions (or a range of scores [x-a,x+a]) and choose the threshold as the midpoint of the window at the position where the average utility is maximized.

Many of the groups recognized the danger of using the same training set for both profile construction and threshold selection. Relevant documents used to construct the profile will have a much greater similarity to the profile itself than relevant documents which appear in the test data. As discussed in the previous section, this may lead to biased thresholds, a problem which is common to all methods of threshold selection. Both City and CLARITECH experiment with partitioning the training data into independent segments (City uses many different partitions), but they find that the runs based on the partitioned data tend to be less successful than runs based on the full data. This is probably because the runs based on partitioned data have fewer relevant documents to work with for profile construction and optimization. Perhaps there is a way around this trade-off. If the partitioned data set can be used to estimate the amount of bias directly, one could then retrain the system on the full data set and apply the derived bias correction factor to obtain the final thresholds.

There are still many more factors to explore in future filtering experiments. For example, none of the groups made real use of the time ordering of the training data. It is possible that the distribution of a term over time may be related to its value as a feature in the filtering profile. Terms occuring frequently over a short period of time may reflect a single event and thus not have enduring predictive value. One can also model the density of relevant documents as a function of time, which is a crucial factor in threshold selection. Even if the scoring algorithm is optimal and the distribution of relevant and non-relevant documents is the same over both the training and the test data, the ideal threshold can change substantially depending upon the relative density of relevant and non-relevant documents. Experiments in adaptive filtering are only beginning. Only one group (UMass) tried adaptive filtering for TREC-6. Readers are referred to their paper for more details about the methods and the results.

The filtering track will continue at TREC-7. In fact, it will be expanding, as the routing task will be folded into the filtering track next year. There will be an increased emphasis on adaptive filtering, as this is thought by most to be a more realistic problem, and it provides the new challenge of sequential or on-line learning. However, routing and batch-style filtering will continue to exist to support research in traditional text categorization and machine learning algorithms. We encourage everyone with interest in the subject to participate in the filtering track next year.

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, 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 5th Text Retrieval Conference (TREC-5). In *The 5th Text Retrieval Conference (TREC-5), NIST SP 500-238*, pages 1–28, 1997.

[5] Ellen M. Voorhees and Donna Harman, editors. *The 6th Text Retrieval Conference (TREC-6)*, 1998. To appear.