The TREC-8 Filtering Track Final Report

David A. Hull
Xerox Research Centre Europe
Meylan, France
hull@xrce.xerox.com

Stephen Robertson
Microsoft Research
Cambridge, UK
ser@microsoft.com

Abstract

The TREC-8 filtering track measures the ability of systems to build persistent user profiles which successfully separate relevant and non-relevant documents. It consists of three major subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system begins with only a topic statement and must learn a better profile from on-line feedback. Batch filtering and routing are more traditional machine learning tasks where the system begins with a large sample of evaluated training documents. This report describes the track, presents the evaluation results in graphical format, and provides a general commentary on lessons learned from this year's track.

1 Introduction

A text filtering system sifts through a stream of arriving information to find documents relevant to a set of user profiles. Unlike the traditional search query, user profiles are persistent, and tend to reflect a long term information need. With user feedback, the system can learn a better profile, and improve its performance over time. The TREC filtering track tries to simulate on-line time-critical text filtering applications, where the value of a document decays rapidly with time. This means that potentially relevant documents must be presented immediately to the user. There is no time to accumulate and rank a set of documents according to their relevance. Evaluation is based only on the quality of the retrieved set, which is scored using a utility measure. The utility measure assigns a positive score for each relevant document retrieved and a negative score to each retrieved document that is not relevant.

Filtering differs from search in that documents arrive sequentially over time. The TREC filtering track consists of three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system starts with only a user profile and must begin filtering documents without any other prior information. Each retrieved document is immediately judged for relevance, and this information can be used by the system to adaptively update the filtering profile. In batch filtering and routing, the system starts with a large set of evaluated training documents which can be used to help construct the search profile. For batch filtering, the system must decide to accept or reject each document, while routing systems can return a ranked list of documents. The core tasks have remained the same in TREC-7 and TREC-8.

Traditional adhoc retrieval and routing simulate a non-interactive process where users look at documents once at the end of system processing. This allows for ranking or clustering of the retrieved set. The filtering model is based on the assumption that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than create a complex simulation which includes partial batching and ranking of the document set, we make the simplifying assumption that users want to be notified about interesting documents as soon as they arrive. Therefore, a decision must be made about each document without reference to future documents, and the retrieved set is ordered by time, not estimated likelihood of relevance. The history and development of the TREC Filtering Track can be traced by reading the yearly final reports:

- TREC-7 http://trec.nist.gov/pubs/trec7/t7_proceedings.html (#3 - 2 files) [3]
- TREC-6 http://trec.nist.gov/pubs/trec6/t6_proceedings.html (#4 and #5) [2]
- TREC-5 http://trec.nist.gov/pubs/trec5/t5_proceedings.html (#5) [5]
- TREC-4 http://trec.nist.gov/pubs/trec4/t4_proceedings.html (#11) [4]

Information on the participating groups and their filtering systems can be found in the individual site reports, also available from the TREC web site.

2 TREC-8 Task Description

The basic filtering tasks have not changed from TREC-7 to TREC-8, so readers familiar with the TREC-7 task may wish to skip this section. In this section, we review the corpus, the three sub-tasks, the submission requirements, and the evaluation measures. For more background and motivation, please consult the TREC-7 track report [3]. The TREC-8 filtering experiments used the Financial Times document collection (TREC disk 4), which consists of slightly more than three years of newspaper articles covering part of 1991 and most of 1992-1994. The 210,000 documents were ordered roughly as a function of time, and all systems were required to process the collection (or a subset) in the same order. The documents average 412 words in length and cover a wide variety of subject matter. All tasks used topics 351-400, which were constructed for the TREC-7 adhoc experiments. The topics contain Title, Description, and Narrative fields and have an average length of 58 words.

The adaptive filtering task is designed to model the text filtering process from the moment of profile construction. No training documents are provided. However, once a document is retrieved, the relevance assessment (when one exists) is immediately made available to the system. Unfortunately, it is not feasible in practice to have interactive human assessment by NIST.[1] Instead, assessment is simulated by releasing the pre-existing relevance judgement for that document. Judgements for unretrieved documents are never revealed to the system. Once the system makes a decision about whether or not to retrieve a document, that decision is final. No back-tracking or temporary caching of documents is allowed. While not always realistic, this condition reduces the complexity of the task and makes it easier to compare performance between different systems.

Systems are allowed to use the rest of the TREC document collection (excluding the Financial Times) to generate collection frequency statistics (such as IDF) or auxiliary data structures (such as automatically-generated thesauri). While access to relevance judgements for topics 351-400 was restricted to the final run, systems could be trained using other topics on the rest of the TREC collection. As documents were processed, the text could be used to update term frequency statistics and auxiliary document structures even if the document was not matched to any profile. Groups had the option to treat unevaluated documents as not relevant. Evaluation is based on utility, as described in the next section.

[1] Individual participants have the option to assess documents manually, although all such runs are evaluated in a different category. No groups submitted manual runs for TREC-8.

In batch filtering, documents and relevance judgements from the 1991-1992 Financial Times are available in advance as a training set. The 1993-1994 Financial Times documents form the test set. As in adaptive filtering, systems may use the relevance judgement from any retrieved document to update the filtering profile. Evaluation is based on utility. Routing is similar to batch filtering, with the following two differences. Systems are allowed to use relevance judgements for topics 351-400 from other parts of the TREC document collection as part of the training set. For routing, systems return a ranked list of the top 1000 retrieved documents and are evaluated according to average uninterpolated precision, as in adhoc search. Batch filtering and routing are included to open participation to as many different groups as possible and to improve the quality of the document pool used for evaluation.

2.1 Evaluation

For the TREC experiments, filtering systems are expected to make a binary decision to accept or reject a document for each profile. Therefore, the retrieved set consists of an unranked list of documents. The primary evaluation measure is 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. For linear utility, the score is a linear combination of the elements in the contingency table shown below:

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

```
Linear 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-8, we use two different linear utility functions:

```
LF1 = 3*R+ - 2*N+         --> retrieve if P(rel) > .4
LF2 = 3*R+ - N+           --> retrieve if P(rel) > .25
```

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.

In addition, we tested the following non-linear utility functions for the first time at TREC-8:

$$\mathrm{NF1} = 6 * (\mathrm{R}+)^{0.5} - \mathrm{N}+$$

$$\mathrm{NF2} = 6 * (\mathrm{R}+)^{0.8} - \mathrm{N}+$$

The idea behind the non-linear utility functions (originally due to I.J. Good [1]) is that the value of a relevant document depends on how many relevant documents have been seen before. In this formulation, the more relevant documents retrieved, the lower the value of each additional relevant document. From a practical perspective, our hope is that the non-linear functions will do a better job of equalizing the size of the retrieved sets for topics with large differences in the number of relevant documents.

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 scale the utility scores prior to averaging. 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.

The scaling function used for TREC-8 is:

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

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. 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. For simplicity, we choose to use the same value for the zero point and the lower bound in these experiments.

2.1.1 Evaluation measures – further discussion

During the period between TREC–7 and TREC–8, there was some continuing discussion of the problem of devising suitable evaluation measures for filtering tasks. This discussion followed several different lines, and by no means all the contentious issues were resolved. This section attempts some kind of summary of parts of the discussion. The reasons for the introduction of the utility scaling function and of the non-linear utility functions were discussed above. Both ideas were adopted in the plan for the TREC–8 filtering track, and were included in the guidelines. However, both ideas came into some question in the subsequent discussion on the track mailing list.

One characteristic of utility in general that attracted comment was its relationship to recall and precision. With any utility function, it is possible to imagine two systems X and Y which have the following characteristics:

$$\text{Precision}(X) \quad > \quad \text{Precision}(Y)$$

$$\begin{aligned} \text{Recall}(X) &> \text{Recall}(Y) \\ \text{but } U(X) &< U(Y) \end{aligned}$$

With the linear functions, this arises only when both utilities are negative; this can be demonstrated as follows. Suppose the system retrieves R^+ relevant and N^+ non-relevant documents (as above), and (again as above) the utility function is:

$$U = AR^+ + BN^+ = R^+(A + B\frac{N^+}{R^+})$$

(B will of course be negative). Then we assume for precision that $\frac{R_X^+}{(R_X^+ + N_X^+)} > \frac{R_Y^+}{(R_Y^+ + N_Y^+)}$ which is equivalent to $\frac{N_X^+}{R_X^+} < \frac{N_Y^+}{R_Y^+}$. For recall we have $R_X^+ > R_Y^+$. If $U(Y)$ is positive (which implies that $A + B\frac{N_Y^+}{R_Y^+}$ is positive), then

$$\begin{aligned} U(X) &= R_X^+(A + B\frac{N_X^+}{R_X^+}) \\ &> R_Y^+(A + B\frac{N_X^+}{R_X^+}) \\ &> R_Y^+(A + B\frac{N_Y^+}{R_Y^+}) \quad \text{since } B < 0 \\ &= U(Y) \end{aligned}$$

However, if $U(X)$ is negative, we can infer that $U(Y)$ is also negative, but that the above inequalities are reversed, and $U(X) < U(Y)$. Non-linear utility functions exhibit similar behaviour under different conditions, and in particular may do so in the positive range. The behaviour was in fact in evidence in some of the TREC–8 results (see section 3.3).

The significance of this fact was the subject of some debate in the group. It is clear that this reflects a genuine characteristic of user preference, provided that user preference is well described by the utility function. However, it also implies that a system might be improved by introducing a random element to it, which suggests that what we are measuring is less a characteristic of a system and more of a particular user preference.

Also, one justification for the scaling function is that the zero of the utility scale is essentially arbitrary: given any decision rule based on probability of relevance, such as those indicated in section 2.1 above, it could be derived from any number of different utility functions with different zero points. This suggests that an arbitrary redefinition of the zero point (which the scaling function achieves by setting the utility of the worst possible system to zero) will not interfere too much with the essential properties of utility. However, the present argument casts doubt on this conclusion, because even if two different utility functions with two different zeros generate the same decision rule, they will have different relationships to recall and precision. Scaling by the method chosen is *not* equivalent to choosing a utility function with a different zero, and produces different behaviour.

This discussion took place shortly before TREC–8, and therefore did not affect the track guidelines or way the results were evaluated. However, it is clear that the issues raised will have to be revisited for TREC–9.

2.2 Submission Requirements

Each participating group could submit up to six adaptive filtering runs and up to two runs each for batch filtering and routing. Each adaptive or batch filtering run was evaluated according to a pre-specified utility function. There were no required runs this year, although we asked each group that participated in adaptive or batch filtering to submit one run optimized for the LF1 measure. 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.

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.

Topics 351-400 were evaluated against the Financial Times collection for the TREC-7 adhoc experiments, and these relevance judgements served as the basis for the filtering tasks. However, we wanted to reduce the risk that systems which retrieved many unevaluated documents might be unfairly penalized, so we asked NIST to perform a second round of evaluation. We are grateful to NIST for following through on this request, given the heavy assessment load for TREC-8. In the end, NIST managed to evaluate the top 57 documents from every submitted run, although documents with existing relevance judgement were not re-assessed. A significant number of new relevant documents were found in the second round of assessment, indicating that it was a worthwhile endeavor[2].

3 TREC-8 results

Fourteen groups participated in the TREC-8 filtering track and submitted a total of 55 runs. This represents roughly a 20% increase over the TREC-7 participation level.

	# groups	# runs
Total	14	55
-----	--	--
adaptive	8	33
batch	6	11
routing	6	11

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.

- City Univ. London / Microsoft [City] (plt8f)
- CLARITECH Corporation [CLARITECH] (CL99)

[2] As reported in the overview presentation of the TREC-8 Conference by Ellen Voorhees.

Organization	Runs	1st yr?	Adaptive	Batch	Routing
City	plt8f	-	-	X	X
CLARITECH	CL99	-	X	X	-
DSO	dso99r	X	-	-	X
ICDC	S2N2	X	-	-	X
IRIT	Mer8	-	-	X	X
KDD	kdd8f	X	X	-	-
Microsoft	ok8f	-	X	-	-
CUNY	pir9	-	X	X	X
Rutgers-K	Ant	-	X	X	-
Seoul	Scai8	X	-	X	-
TNO	uttno8	-	X	-	-
UIowa	IOWAF	-	X	-	-
UMass	INQ	-	X	-	-
UMaryland	umr	X	-	-	X

Table 1: Summary of task participation (X = participant).

- DSO National Laboratories, Singapore [DSO] (dso99r)
- Informatique-CDC - Groupe Caisse des Dépôts / ESPCI [ICDC] (S2N2)
- IRIT / University of Toulouse (IRIT) (Mer8)
- KDD [KDD] (kdd8f)
- Microsoft Research - Cambridge [Microsoft] (ok8f)
- Queens College CUNY [CUNY] (pir9)
- Rutgers University [Rutgers-K] (Ant)
- Seoul [Seoul] (Scai8)
- TNO-TPD / Univ. Twente [TNO] (uttno8)
- University of Iowa [UIowa] (IOWAF)
- University of Massachusetts Amherst [UMass] (INQ)
- University of Maryland [UMaryland] (ume)

Table 1 summarizes the tasks each group participated in and whether or not this is their first year participating.

3.1 Summary of approaches

In this section, we present a one-paragraph description of the techniques used by each of the groups in their TREC-8 experiments. Essentially, this information is just a summary of the final papers. The goal is to enable readers to quickly identify the papers they wish to read and to provide background information to motivate the commentary on the evaluation results which follows. We were unable to find final papers for IRIT, Rutgers-K, Seoul, and UMass at the time this document was written.

City participated in batch filtering and routing. They focused on measuring the efficiency of their Pliers system for text filtering using an architecture of 16 Pentium II machines in parallel. Their routing system is based on the Okapi TREC-5 strategy of partitioning the training set into two parts: one for term extraction and one for term selection. Their system indexed the training set (FT91-92) in 5 minutes and filtered the test set in less than 11 minutes.

CLARITECH participated in adaptive and batch filtering. For TREC-7, they concentrated on threshold selection and updating. For TREC-8, they added profile-specific updating (as opposed to their former strategy of updating all profiles at the same time after a batch of documents has been processed) and worked to optimize term selection and the profile scoring function. They chose to update each profile individually every n (usually 2 or 4) documents retrieved or after a fixed amount of time has passed. The former update allows the system to quickly take advantage of new training data (and improved performance) while the latter update allows the system to make adjustments to profiles which have retrieved no documents (and had mixed results). They also found that changing the terms in the profile in response to new training data improves performance even though it makes accurate threshold calibration more difficult.

DSO participated in routing. They select terms using the AT&T TREC-6 routing method then assign weights to the terms using a specialized perceptron learning algorithm (dso99rt1). Their second run (dso99rt2) merges the results of perceptron learning with Dynamic Feedback Optimization.

ICDC participated in routing. Each document is initially represented by terms which have a high frequency in the document relative to their frequency in the corpus. In order to reduce correlation among previously selected terms, a Gram-Schmidt orthogonalization technique is used to find the most descriptive term at each step, given the terms already chosen by the model. The learning algorithm is a linear neural network with early stopping to reduce overfitting.

KDD participated in adaptive filtering. They define the contribution of a term to the similarity of a query document pair as the difference between the similarity scores with and without that term. Words with negative scores are prime candidates for query expansion. The basic filtering model is Rocchio relevance feedback with terms also weighted and selected as a function of their contribution as defined above.

Microsoft participated in adaptive filtering. Their basic system uses Okapi term weighting with no query modification or term reweighting based on feedback data. Initial thresholds (probability estimates) are set via logistic regression on a separate training set of documents and topics. The probabilities are a function of the retrieval score, the average score of the top 1% of retrieved documents (also initially estimated from the training set), the maximum possible score, and the length of the query. The thresholds are set low initially to allow more profile learning. Documents are processed in weekly batches and the intercept term in the regression is updated. In general, they find that starting with higher initial thresholds works better.

CUNY participated in all three tasks. For adaptive filtering, they concentrate on threshold updating. Their system defined a starting threshold (Thi), a lower bound threshold (Tlo), a precision threshold (G), and a selection rate threshold (SRT). The current threshold T is updated every 2000 documents. If no relevant docs have been seen, the current threshold (T) is decreased if the proportion of documents retrieved is less than SRT, but never less than Tlo. Otherwise, the threshold is increased if the precision of the current retrieved set is less than G and decreased if precision is greater than G. For batch Filtering and Routing, CUNY's Pircs system uses genetic algorithms to select and modify multiple profiles. Logistic regression is used for threshold selection.

TNO participated in adaptive filtering. They use a probabilistic retrieval model which assumes that the user generates the query from an ideal internal representation of a relevant document. The initial thresholds are set to a large multiple of the probability of selecting the query from a random document. Each time a document is selected for a profile, the threshold is adjusted to a value slightly lower than the one which achieves maximum performance on the retrieved document set. The model defines a relevance weight for each term (ranging from 0 to 1) which is updated for new data using the EM algorithm.

UIowa participated in adaptive filtering. They use a dynamic clustering approach with two acceptance thresholds. The first acceptance threshold is based on similarity to the topic. All documents scoring higher than this threshold are dynamically clustered. When a cluster's similarity first exceeds a second visibility threshold, the most recently retrieved document is sent to the user. Clusters are then tagged as relevant or not relevant based on the assessment of this document, and future documents from relevant clusters are also passed on to the user. A document judged not relevant in a relevant cluster spawns an independent non-relevant cluster.

UMaryland participated in routing. They are interested in using Latent Semantic Indexing (LSI) for collaborative filtering. Routing queries are constructed by relevance feedback and then an LSI dimension reduction is performed in the query space. The goal is to find and reuse common structure in the query set. They submitted one run based on relevance feedback alone (umrqz) and one run with relevance feedback plus LSI (umrlsi). Overall, LSI tended to slightly hurt performance, perhaps because there is minimal topic overlap in the TREC-8 query set.

3.2 Evaluation results

Figures 1-8 summarize the evaluation results for the TREC-8 filtering track. Not all runs appear on all graphs due to scaling problems. Missing runs scored below the lower bound on the vertical axis. There are three different types of graph. Figures 1-2 and 6-7 plot average scaled utility for a range of values for the lower bound. The horizontal axis represents the number of non-relevant documents used to define the lower bound (logarithmic scale). The vertical axis represents the difference in average scaled utility between each system and the baseline (retrieving no documents at all). Therefore, the baseline is represented by the straight horizontal axis marked with index points for the lower bound. Figures 3-5 plot average scaled utility by year for adaptive filtering. Only a small fraction of the documents come from 1991, so it can be ignored. The lower bound in these plots is fixed (the value of s is shown at the bottom of the plot). Figure 8 shows the average uninterpolated precision scores on the left side of the plot. For the ride side, a system's score is replaced by its rank with respect to that topic, and these ranks are averaged across topics. The results are then rescaled to cover the same range as the raw scores.

The reader will note that there are no plots for non-linear utility. Only two groups submitted runs optimized for these measures, one by CLARITECH for NF1 and adaptive filtering, one by IRIT for NF1 and batch filtering. Therefore, there are no meaningful comparisons that can be made. We had hoped that the non-linear utility functions would do a better job of equalizing the size of the retrieved sets. However, a comparison of the variation in the size of the retrieved set between CLARITECH's NF1 run and their four LF1 runs shows that there is no significant difference. Given the lack of interest in the non-linear utility functions, it is unlikely that this experiment will continue next year. several groups expressed the sentiment that the non-linear utility functions do not represent a good user model for filtering.

Figure 1 shows average scaled utility for the LF1 function and the adaptive filtering task. This plot demonstrates that the LF1 function is a very challenging standard, as none of the systems manage to beat the baseline. Most of the lines do not cross, indicating that relative system performance changes very little as a function of the lower bound. The convergence of lines as the lower bound increases is a natural phenomenon caused by the fact that the possible range of scores is much larger, therefore the actual scaled scores will cluster in a smaller part of the space. This pattern will be observed in all plots of this type. The two most successful runs (IOWA-2 and TNO-1p) owe their success to their extremely conservative document selection strategies. Figure 2 shows average scaled utility for the LF2 function and the adaptive filtering task. The task here is simpler, and five systems consistently beat the baseline.

Figures 3 and 4 show average scaled utility for the LF1 function and adaptive filtering broken down by year. The difference between the two figures is the scale of the vertical axis. In addition, the TNO and CLARITECH runs were averaged in Figure 3 to avoid crowding the display at the larger scale. All systems start below the baseline in 1992, due to the need to calibrate thresholds by retrieving a few documents with little prior information about the odds that they are relevant. It is encouraging to see that by 1993 six different runs are above the baseline and this pattern continues into 1994. However, there are striking differences in system behavior in these plots. We can distinguish three different patterns in this plot. CL99afL1(b,c) and uttno8lf1 improve gradually and consistently over time. CL99afL1(a,d) and uttno8lf1(f,p) improve significantly in 1993 and then fall back slightly in 1994. Microsoft (ok8f), and to a lesser extent UMass (INQ) and UIowa, do more poorly in 1993 but recover nicely in 1994. This indicates that learning behavior differs substantially from system to system over time.

Figure 5 shows average scaled utility for the LF2 function and adaptive filtering broken down by year. We see similar patterns to Figure 4, except that most systems are well above the baseline in 1993 and 1994. Note that the utility scores were scaled independently by year. This means that the scores are not additive, i.e. you can't add up the scores for a system in Figure 5 and get the score in Figure 4 at the same lower bound. This is because the upper bound is different for each plot. In hindsight, it might have made more sense to scale once and then divide up the score by year.

Figures 6 and 7 show average scaled utility for the LF1 and LF2 functions respectively and batch filtering. Almost all systems score well above the baseline. Figure 8 shows the average uninterpolated precision and average scaled rank for routing systems.

3.3 Utility vs. Other Measures

The evaluation results for adaptive and batch filtering presented in the previous section are all based on scaled utility. In order to give a broader view of system performance, we look at a number of alternative measures in Table 2 and Table 3 below. Scores are tabulated for the best run from each of the five groups with the highest average scaled utility ($U(s)$ = -100). The measures include average size of the retrieved set, micro-averaged precision (precision of retrieved set as a whole), total number of relevant documents retrieved, and the number of topics with utility score greater than zero. We use micro-averaging for precision to avoid the problem of empty retrieved sets. It is clear from the results that scaled utility hides a lot of important differences in system behavior.

The most striking pattern for LF1 is that scaled utility is inversely correlated with average set size and the total number of relevant documents. In other words, retrieving more documents and more relevant documents leads to worse performance. Even more surprising, uttnolf1p and CL99afL1d have a higher precision and a higher recall than IowaF992 but receive a lower utility score! This is because the precision threshold for LF1 is 0.4, and all systems retrieve with a precision below this level. When a system fails to meet this threshold, each new document has a negative expected utility, and retrieving more documents lowers the overall utility score. Therefore, higher precision *and* higher recall does not always translate into better performance when evaluation is based on utility. This particular property runs counter to most intuitive notions in information

Run Name	u^*_{50} LF1	Average Set Size	Micro Avg Precision	Total Rel.	# Topics LF1 > 0
Baseline	0.0	0.0	–	0	0
IowaF992	-0.012	3.5	0.32	57	10
uttnolf1p	-0.015	5.7	0.36	102	11
CL99afL1d	-0.025	10.0	0.36	180	11
INQ610	-0.071	17.4	0.31	266	12
ok8f311	-0.153	23.3	0.25	295	14

Table 2: Comparison of LF1 to other measures.

Run Name	u^*_{100} LF2	Average Set Size	Micro Avg Precision	Total Rel.	# Topics LF2 > 0
uttnolf2f	0.034	23.2	0.32	372	20
CL99afL2	0.025	11.3	0.35	197	17
INQ612	0.013	17.4	0.31	266	18
IowaF991	0.007	4.3	0.34	74	13
Baseline	0.0	0.0	–	0	0
ok8f321	-0.011	29.2	0.27	398	23

Table 3: Comparison of LF2 to other measures.

retrieval and may help explain why many people are uncomfortable with the utility measure.

For LF2, system behavior is more regular. Generally, the systems that retrieve more documents with a higher precision score better. In this case, all the top-ranked systems exceed the LF2 precision threshold of 0.25. It is sobering to note that even for the easier LF2 function, the best systems retrieve with positive utility on fewer than half the topics. For LF1, the best systems beat the baseline on fewer than 30% of the topics!

It is also interesting to note that with the exception of TNO, there is much more similarity (in terms of set size, precision, and number relevant) between runs from the same system for LF1 and LF2 than between different systems for the same utility function. This pattern is exemplified by the success of INQ610, which appears to be the same run submitted for both utility measures! One possible explanation is that many topics are highly insensitive to the choice of retrieval threshold and that differences in profile construction are more important for changing the size of the retrieved set. Most likely, this reflects the fact that best performance for LF1 and LF2 for most systems on most topics is to retrieve no documents, and a wide range of thresholds achieve this outcome.

4 General Commentary

Following the progression of system performance from TREC-7 to TREC-8 (or lack thereof!), it is becoming increasingly clear that the adaptive filtering task is too hard. Once again, no system performed better than the baseline over all three years for the LF1 utility function. In other words, it is better not to retrieve any documents according to this standard. Furthermore, systems with higher precision and higher recall get a lower score, because they are unable to retrieve with high enough precision. While systems are clearly ahead of the baseline for LF2, gains are concentrated in fewer than half the topics. Looking at Figures 3-5, we realize that the situation is not as bad as it first appears. Most systems suffer during the start-up phase but go on to beat the baseline quite substantially in 1993 and 1994. However, the current utility functions (particularly LF1) simply penalize too strongly against retrieving non-relevant documents early to allow systems to develop an overall winning strategy.

This suggests a clear road for improvement in TREC-9. Next year, we propose to supplement the topic statement with one or two positive training examples in order to give systems more of a head start. In addition, we will find a topic set with a larger number of relevant documents. The current TREC tasks favor highly conservative filtering strategies which creates an environment where there is little opportunity for adaptive learning. In addition, there is some interest in working with evaluation measures based on different user models, such as: find n relevant documents as fast as possible or return no more than k documents per unit time. We are strongly considering moving to a text categorization test collection for TREC-9, to give us more flexibility in topic selection and to reduce the assessment load for NIST. This will introduce new challenges, such as defining an acceptable topic statement from category labels that may not be well defined.

When looking over the system reports, several clear patterns emerge. As suggested in our commentary last year, it is important to take advantage of new training data as quickly as possible. For TREC-7, most systems used batch updating, learning new profiles and threshold simultaneously for all topics every k documents filtered. In TREC-8, the most successful systems are now updating their profiles independently with a much smaller batch ratio (usually 1 or 2 documents). Systems like Okapi and Pircs, which still use large batches, have much more trouble learning rapidly in the first year, a crucial period for determining overall performance. Differences in updating strategies may explain two of the dominant learning behaviors found in Figures 3 and 4. These systems gain more training data, which enables them to make more informed decisions later on, but not enough to overcome the cost of all the irrelevant material passed on to the user. Given that people tend to expect good results as soon as possible, this is not likely to be a winning strategy in a real filtering system.

As in TREC-7, most groups concentrate on optimizing their system's adaptive threshold setting ability, rather than changing the terms or the term weights in the profile. Once again, the TREC-8 task encourages highly conservative filtering strategies, which limits the ability of systems to adaptively learn better profiles. Nonetheless, groups which run comparisons with and without profile updating, such as CLARITECH and TNO/Twente, find that profile updating does improve performance. We hope to encourage more work in this area by revising the task for next year. Overall, the TREC filtering track continues to grow and prosper and we look forward to welcoming new participants next year.

Acknowledgements We give our thanks to all the people who have contributed to the development of the TREC filtering track over the years, in particular David Lewis, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, R.W. Hutchinson, Djoerd Hiemstra, the TREC program committee, and the team at NIST.

References

[1] I.J. Good. The Decision Theory Approach to the Analysis of Information Retrieval Systems. *Information Storage and Retrieval Systems*, 3:31–34, 1967.

[2] David A. Hull. The TREC-6 Filtering Track: Description and Analysis. In *The 6th Text Retrieval Conference (TREC-6), NIST SP 500-240*, pages 45–68, 1998.

[3] David A. Hull. The TREC-7 Filtering Track: Description and Analysis. In *The 7th Text Retrieval Conference (TREC-7), NIST SP 500-242*, pages 33–56, 1999.

[4] David Lewis. The TREC-4 Filtering Track. In *The 4th Text Retrieval Conference (TREC-4), NIST SP 500-236*, pages 165–180, 1996.

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