Optimization in CLARIT TREC-8 Adaptive Filtering

Chengxiang Zhai, Peter Jansen, Norbert Roma, Emilia Stoica, David A. Evans

CLARITECH Corporation

Abstract In this paper, we describe the system and methods used for the CLARITECH entries in the TREC-8 Filtering Track. Our focus of participation was on the adaptive filtering task, as this comes closest to actual applications. In TREC-7, we proposed, evaluated, and proved effective two algorithms for threshold setting and updating—the delivery ratio mechanism, which is used to obtain a profile threshold when no feedback has been received, and beta–gamma regulation, which is used for threshold updating. This year, we explored two ways of improving filtering performance given these our threshold-setting algorithms as a basis by (1) allowing profile-specific anytime updating and (2) optimizing the other filtering system components, in particular, the retrieval/scoring mechanism and the profile vector learning. Our results show that profile-specific frequent updating indeed improves filtering performance. In addition, they suggest that optimizing the scoring function and the term vector learning component independently leads to even further improvement, providing another indication of the effectiveness and robustness of our threshold updating mechanism.

1 Introduction

Our basic approach to filtering is a vector-space-based two-step procedure similar to the one used in many other systems: first we compute a relevance score for each document–profile pair, and then we make a binary decision to accept or reject the document based on comparison with a score threshold. When, over time, feedback information becomes available for the accepted documents for any profile, the profile term vector and threshold can be updated periodically.

A filtering system capable of performing these tasks consists of the following major components:

1. An initial profile term vector creation component
2. An initial threshold setting module
3. A scoring mechanism that computes a relevance score based on a document vector and a profile vector
4. A profile term vector updating module
5. A threshold updating module

For TREC-7, we built our filtering system by adapting a regular retrieval system. This approach has the advantage that several of the main required components are already implemented in some form. For example, the creation of an initial term vector (1) and the scoring mechanism (3) are naturally supported by any retrieval system, whereas term vector updating (4) can be achieved by means of any relevance feedback mechanism. That leaves two components to be designed and implemented, namely the two threshold-related components. One component (2) sets the initial threshold at the beginning, i.e., without any feedback information, the other (5) updates the threshold based on the collected user feedback. In TREC-7, we proposed and evaluated an algorithm for each of these: the delivery ratio algorithm for initial threshold setting and the beta-gamma regulation algorithm for threshold updating [Zhai et al. 1999]. They proved to be very effective as reported in the evaluation of the TREC-7 filtering track [Hull 1999].

This year, we examined methods for improving the overall filtering performance given these basic threshold-setting algorithms. We considered two directions: (1) allowing profile-specific "anytime" updating; and, (2) optimizing the other two components in a filtering system: the retrieval/scoring mechanism and the profile vector learning mechanism.

One special challenge in adaptive filtering is the problem of extremely sparse data. Without reasonably frequent feedback—at least some of which must be positive—no effective learning is possible. In our TREC-8 work, we explored the possibility of improving filtering performance by performing frequent profile-specific event-driven (anytime) updating. More specifically, we wanted to allow each profile to update after some small number of new documents were accepted by that profile (2 or 4 in the official experiments). It was here, however, that the limitations imposed on us by the use of a retrieval engine became significant, and this eventually prompted us to a complete redesign of the system.

The second hypothesis we tested in TREC-8 is that we can improve the filtering performance by improving individual filtering system components independently. Somewhat simplifying the classification given above, we can view the filtering problem as consisting of three facets: (a) scoring and ranking, (b) term vector learning, and (c) threshold regulation. We were interested in seeing how techniques for optimizing the scoring function and the term vector learning module would interact with techniques for optimizing threshold learning. To test the hypothesis that the positive effects of the optimization of each individual module would outweigh and supersede any negative effects or interference, we held the beta-gamma variables constant while varying other components of the system. In particular, we varied (1) the TF formula (maximum-frequency TF normalization vs. the BM25 TF formula [Robertson et al. 1994; Robertson & Walker 1994]), which mainly affects the scoring function, and (2) the coefficients in Rocchio vector learning [Rocchio 1971] (static, fixed coefficients vs. dynamic coefficients that depend on the (growing) number of positive examples).

In the following sections, we first describe the new evaluation system we used for our TREC-8 experiments. We then describe our adaptive experiments and analyze the adaptive filtering results. In Section 4, we briefly discuss our batch filtering runs.

2 The CLARIT Adaptive Filtering Evaluation System (CAFES)

Our system for TREC-7 was essentially based on the retrieval-oriented CLARIT toolkit, which formed the core of all our past TREC efforts [Evans & Lefferts 1994; Milic-

Frayling et al. 1998; Zhai et al. 1999]. Though powerful, this toolkit did not allow for easy implementation of profile-specific updating, and offered only limited experimental flexibility in other respects.

Hence, to support the experiments we intended for the current TREC evaluation, we built an entirely new evaluation system only minimally dependent on the CLARIT toolkit—the CLARIT Adaptive Filtering Evaluation System (CAFES). Like most of CLARIT, CAFES is based on the standard vector space model, and written in C++. It consists of many independent modules that interact with each other, allowing maximum flexibility and efficiency in experiments. The algorithms that are implemented include the standard vector-space retrieval algorithms [Salton 1988], supporting various kinds of TF formulas, dot-product similarity measure, the cosine measure, and Rocchio term vector learning [Rocchio 1971], as well as our new delivery ratio and beta-gamma threshold regulation methods [Zhai et al. 1999]. On the downside, the system contains as of yet no support for the use of CLARIT constraints and subdocuments (used for example in our TREC-8 Ad Hoc submission).

The Rocchio learning algorithm is implemented with the flexibility of using different TF weighting methods. Clearly, the choice of TF may affect the centroid vector. Although it is not a-priori clear whether Rocchio learning is compatible with the BM25 TF formula, which was derived based on a probabilistic retrieval model, in practice they appear to work together very well. The Rocchio learning method makes it very easy to update a vector at any time: since we maintain a vector accumulator over time, we simply need to compute the average of the accumulator and then merge the new vector with the original profile vector. Note that, as we allow the IDF to be updated over time, the incremental Rocchio accumulator could contain the sum of document vectors weighted using different IDFs.

The system offers the option to normalize each example document vector (i.e., scale the vector to unit length) before adding it to the accumulator. This gives each example an equal weight in its contribution to the angle / direction of the centroid vector. Without the normalization long vectors dominate, which may be undesirable in some situations, such as when a document treats multiple topics.

When the selected terms are merged into the existing profile vector, the truncated vector formed by the selected terms and the existing profile vector are both first normalized to unit length, after which the new term vector can be rescaled by a parametrized coefficient (the merged weight will be the sum of the original weight (or zero, if the term does not exist in the original profile) and the new term weight multiplied by this coefficient). We provided two ways to assign this coefficient: (1) constant and uniform for all profiles or (2) dynamic, i.e., depending on the number of examples from which the terms are extracted. The dynamic formula is: $C = C_{\min} * a + (1-a) * C_{\max}$, where $a = e^{-b*N}$. The dynamic coefficient is essentially a weighted average of a minimum coefficient and a maximum coefficient in which the weight is determined by the number, N, of examples used and a parameter β that controls the "sensitivity" to *N*. When β is zero, *a = 1*, so $C = C_{\min}$. When β is non-zero, it determines how quickly the coefficient approaches $C = C_{\max}$ as the number of examples increases.

To estimate an initial profile threshold, we implemented the "delivery ratio" method, which was described in [Zhai et al. 1999]. The idea is to set the initial threshold in such a way that the filter can be expected to accept a given ratio or proportion of documents from the stream. The acceptance ratio, or delivery ratio, essentially serves as a substitute for a utility function in the absence of relevance judgments. The actual threshold is estimated by using a small reference corpus.

For threshold updating we implemented the beta-gamma adaptive threshold regulation method proposed in [Zhai et al. 1999]. This method selects a threshold, θ, by interpolating between an "optimal" threshold, θ_{opt} and "zero" threshold, θ_{zero}. The optimal threshold is the threshold that yields the highest utility over the accumulated training data, given the current profile term vector. The zero–threshold is the highest threshold below the optimal threshold that gives a non-positive utility over the training data under the assumption that all documents that were rejected are non-relevant. The interpolation factor is sensitive to the number of judged examples used for computing the threshold. The actual formula is

$\theta = \alpha\theta_{zero} + (1-\alpha)\theta_{opt}$, where $a = \beta + (1-\beta)e^{-\gamma N}$.

The parameter β corresponds to a correction factor for the training score bias, and γ expresses our confidence in the optimal utility threshold based on the number of judged examples, *N*, to compute it.

The beta-gamma threshold regulation method requires the scoring of the training examples. Doing this for all of the examples would be very expensive, but fortunately this is not necessary. All we need are the documents with scores either above or closely below the threshold point. Thus, in CAFES, we maintain a "sample set" of training documents for precisely this purpose. Specifically, we keep track of only the documents above a reference threshold, usually a certain fraction of the regular threshold (e.g., one half). Naturally, when a threshold is updated, the reference threshold will also be updated. The effect is that the sample for threshold training consists of all the documents above an adaptive reference threshold. To control the use of sample documents, we introduced a few additional parameters. For example, we can restrict the actual sample used for threshold training to those that are not "K-documents older" than the current document, unless there are too few samples, where "too few" is defined by other parameters. This gives us the flexibility to emphasize recent documents in certain ways.

One of the most significant features of CAFES is that it allows profile-specific anytime updating. The updating time is an important parameter in profile-specific updating. In our system, a profile (both the vector and the threshold) will be updated when one of the following conditions is satisfied:

1. It has accepted enough new documents, where enough is defined by a parameter.

2. It has not been updated for a long time, where the length of time elapsed is specified by a parameter. This time elapse will be referred to as the maximum update delay (MUD).

The idea behind the first condition is to allow us to control the update timing based on the amount of feedback information, which can be measured by the number of

documents accepted for the profile. Alternatively it can be measured by the number of *relevant* documents accepted, which we will explore in the future. The idea behind the second condition is to address the problem of under-delivery (e.g., when the initial threshold is set too high), by allowing a profile to update even if it has not accepted any documents. These two conditions also allow us to move smoothly between the uniform/chunk-based updating strategy and the profile-specific strategy. If the "enough" parameter is very big, then the update timing is similar to chunk-based or uniform updating, as every profile is updated after the same interval, determined by the elapsed parameter. If the lapse is very long, on the other hand, then the updating time is determined solely by the amount of delivery for each profile, which provides profile-specific timing for updating.

Of course, the threshold must be updated whenever the term vector is updated, because the old threshold (trained using the old vector) would no longer be compatible with the new vector.

3 Adaptive Filtering Experiments

3.1 Experiment design and official submissions

Preprocessing. All documents, including the testing documents and the reference documents, and all topic descriptions are pre-indexed by all the single words occurring in noun phrases as well as all the two-word noun phrases, as recognized by the CLARIT Parser [Milic-Frayling et al. 1998]. Preprocessing only eliminated the "<PROFILE>", "<DATE>", "<PUB>", and "<PAGE>" fields, so the controlled-language fields remained in the corpus and were used for indexing.

IDF Statistics. The IDF statistics are needed for term weighting, which affects both scoring accuracy and Rocchio term vector learning. Because we cannot use IDF statistics collected from the testing documents, we constructed our initial IDF statistics from all the 1991 Financial Times documents (FT91) (~14MB, 5368 documents), which are not part of the testing set. Ideally, we would like to update the IDF statistics by gradually mixing the initial statistics with the term counts in the actual documents seen over the stream, as the system sees more and more documents. However, our preliminary experiments on AP and FBIS data indicate that doing so does not seem to have as much impact on the overall filtering performance as other factors. Hence, we decided to use the initial IDF statistics throughout.

Initial profile term vector. An initial profile vector is built from the original topic description (using all fields). The profile terms are assigned TF-IDF weights. There are two different TF formulas that we tried. One is the standard maximum frequency normalized TF score (MaxNorm TF) used in the CLARIT system (i.e., $0.1 + 0.9 \times TF(t)/MaxTF(d)$)[1]. The other is the BM25 proposed by Robertson and colleagues [Robertson et al. 1994; Robertson & Walker 1994]. For the BM25 TF formula, since we cannot use the average length as computed over the testing documents, we simply set it as a parameter. In all the BM25 TF experiments, the average document length is set to 1,000 (tokens). The parameter k and b in BM25 are set to 1 and 0.5, respectively. The length for the query BM25 formula is set to 20.

Scoring algorithm. Two document profile matching formulas are considered: one is the cosine measure, and the other is simply the dot product. Since only the dot product makes sense for the BM25 TF, we always use dot product when the TF formula is BM25. When the TF formula is MaxNorm, we always use the standard cosine measure.

Initial profile threshold. We use the same initial threshold setting algorithm and the same parameter value as we used in TREC-7: the delivery ratio threshold method with a delivery ratio of 0.0005. That is, our initial threshold corresponds to accepting one out of every 2,000 documents. The initial threshold is estimated based on the FT91 data.

Threshold updating. We use the same threshold updating algorithm as we did in TREC-7: the beta-gamma threshold regulation algorithm. Beta was always set to 0.1. We chose a value of 0.1 for gamma (instead of the 0.05 we used for TREC-7), because, in our preliminary experiments with AP and FBIS data, this was found to be a more robust setting. The reason for this difference might be that, instead of using examples only from the previous chunk, we now used all the past examples for each profile for threshold updating. In other words, the meaning of *N* in our beta-gamma method has changed slightly. Threshold updating involves the use of all past examples up to 1,000 samples. If more than 1,000 samples are used, we forget any sample older than 30,000 documents. The sample is collected by using a second reference threshold, which is set to half of the regular threshold. The profile-specific updating time is either of two or four new accepted documents. A profile also must be updated if it has not been updated for more than 3,000 documents in the testing stream.

Term vector updating. We use Rocchio term vector learning, but only positive examples are used to expand the profile. Specifically, a centroid vector accumulator is updated whenever a profile accepts a relevant document. The terms in an example document are weighted by TF-IDF weighting. The TF formula is the same as the TF formula used in scoring. So, if BM25 is the TF formula used for scoring, then it is also used to weight the terms for the purpose of Rocchio learning. When the profile term vector is to be updated, this centroid accumulator is used to select the top 20 highest scored terms to add to the original profile. We tried two different methods for setting the coefficient: (1) fixed coefficient (0.1) and (2) dynamic coefficient (Cmin=0.1, Cmax=2, b=0.1) [see Section 2].

Official submissions. We submitted six runs for the adaptive filtering task, including four runs optimized for LF1 and two other runs optimized for LF2 and NF1 respectively. The parameters used in each run are shown in Table 1.

Run	TF formula	Rocchio coeff	Updating Interval
CL99afL1a	MaxNorm	Dynamic	4
CL99afL1b	BM25	Dynamic	4
CL99afL1c	BM25	Dynamic	2
CL99afL1d	BM25	Static	2
CL99afL2	BM25	Static	2
CL99afN1	BM25	Dynamic	2

Table 1. Official runs

[1] We changed the constant coefficients for this TF formula with respect to the ones used in earlier CLARIT experiments ($0.5 + 0.5 \times TF(t)/MaxTF(d)$) because we found this to work slightly better in the context of the CAFES system.

3.2 Results Analysis

3.2.1 Effect of frequent profile-specific updating

The updating time can be an important parameter in adaptive filtering. Intuitively, we want to update as frequently as possible to take advantage of any feedback information as early as possible. However, with little training information, overly frequent updates may mislead the system, as the training algorithm may show divergent behavior. In our preliminary experiments on AP and FBIS data, we compared uniform updating at every N documents with (in addition) profile-specific updating at every new accepted document. We observed that profile-specific frequent updating is most beneficial for avoiding over-delivery, and usually does not hurt the performance of good-performing topics. Thus, we adopted profile-specific updating for all our official runs, but we varied the updating time. Specifically, CL99afL1b and CL99afL1c are essentially the same, except that CL99afL1b updated profiles every four new accepted documents, while CL99afL1c updated profiles every two new accepted documents. Since the only difference is the use of different updating intervals, the comparison of these two runs will show the effect of frequent profile-specific updating. The results are shown in Table 2.

Time	LF1(Int2)	#Int2>Int	#Int2<Int4	#Int2=Int4
92	1.12	23	2	25
93	0.44	3	0	47
94	0.42	4	1	45
92-94	1.98	23	2	25

Table 2. Effect of frequent profile-specific updating

It is clear that more frequent updating consistently improves performance. We also see that the greatest difference comes from the difference at the time period 92. For periods 93 to 94, there is no difference for most topics. We suspect that this is largely the effect of stopping over-delivery.

Figure 1 shows the effect of different updating intervals (1, 2, & 4) when the MUD constraint is not used.

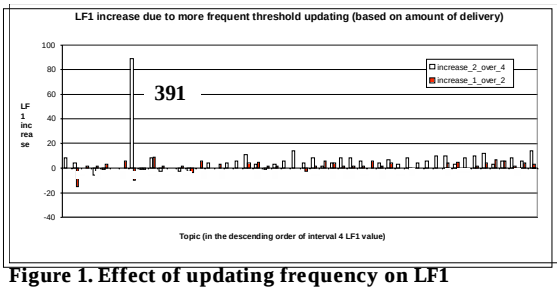

Figure 1. Effect of updating frequency on LF1

We see that a high updating frequency is generally better. Even so, the spectacular increase for topic 391 by 89 points for LF1 from an updating interval of 4 to one of 2 is unexpected. After examining the experiment trace, we found that this is because when the interval is 4, the threshold was raised to a high value quite early in the stream leading essentially to a shut-off of this profile. Specifically, at the second updating point (i.e., after accepting 8 documents), the training set contained two very high-scoring non-relevant documents, and this boosted the threshold up to a value at which it rejected almost all of the subsequent documents. With an interval of 2, the updating points were different, the slightly different threshold accepted a few more documents, and though the threshold was still high, it was low enough for the beta-gamma algorithm to be able to regulate the threshold within an appropriate range. This example shows that with frequent updating the performance is more sensitive to the initial training examples, and that unreliable initial feedback information may cause long-term damage to the profile. It also warns us to be careful about interpreting the results, i.c., the large 'improvement' for topic 391: indeed, had we updated the profile with a much lower frequency, we would not have suffered from the accidental bias in the training sample at all!

We also looked at the effect of the Minimum Update Delay. We observed that while using the MUD seems to have a positive effect when the interval is 2 or 4, it generally has a negative effect when the interval is 1. Our original intent of using the MUD for coping with under-delivery did not work very well.

3.2.2 Effect of TF weighting

The use of different TF formulas affects the filtering performance in several ways: (1) through the scoring function, which affects score range and ranking accuracy and (2) through vector updating, as different terms might be selected or a term may receive a different weight. The difference in score range may also affect the effectiveness of threshold setting, since the parameters in our threshold method might be sensitive to the actual score range. Since we found the BM25 TF formula to be generally more effective for retrieval, we were curious whether it would also improve filtering performance. Some of our official submissions were designed to measure this effect. Specifically, CL99afL1a (MaxNorm TF) and CL99afL1b (BM25) differ only in the TF formula they use. (Note that this also implies the use of different similarity measures, as the BM25 TF formula is best used with the dot product [Robertson & Walker 94], whereas the MaxNormTF formula is used with the cosine similarity measure.)

The results of the comparison of CL99afL1a and CL99afL1b are shown in Table 3. In the table, we have shown the number of topics for which one TF formula is better than, the same as, or worse than the other. We also show the difference of average utility.

Time Period	LF1(BM25)-LF1(MaxNorm)	#BM25> MaxNorm	#BM25< MaxNorm	#BM25= MaxNorm
92	-0.64	19	27	4
93	0.3	12	9	29
94	0.62	9	6	35
92-94	0.28	20	24	6

Table 3. Topic count comparison between MaxNorm TF and BM25 TF formulas

It appears that at the beginning (i.e., for 1992 documents), MaxNorm outperforms BM25 both by average utility and by topic count, but for the later time periods BM25 is better. Overall for all three years of documents, BM25 is better for the average LF1 utility, while MaxNorm is better for more topics. We can also see that most of the difference results in the beginning period. Indeed, for 93 and 94, there is no difference at all for a majority of topics. There are a few topics (e.g., 391) for which BM25 significantly outperforms MaxNorm in all three periods.

It would be interesting to see if the TF formula has contributed to the utility increase by producing a better ranking. In other words, we want to know if there is a correlation between the utility increase and the ranking accuracy improvement.

In Figure 2, we show, for each topic, the increase of LF1 utility value (i.e., LF1(BM25) – LF1(MaxNorm)) over the

whole 92—94 period, along with the scaled increase in the average precision, i.e.,

$$100 \times (\text{avg\_pr(BM25)} - \text{avg\_pr(MaxNorm)})$$

Some topics were excluded because their average precision was not very reliable (for those topics neither TF formula retrieved more than three relevant documents over 92—94). The average precision over all 50 topics is much better for BM25 than for MaxNorm (0.26 vs. 0.19).

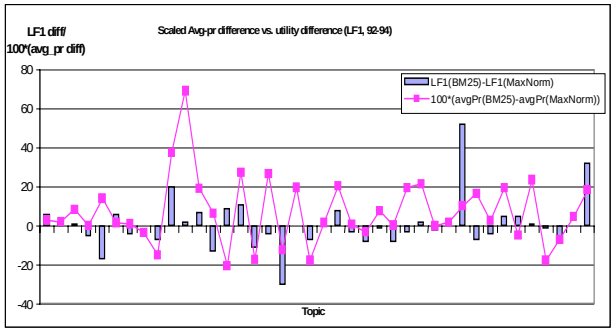

Figure 2. Ranking accuracy and utility increase

The correlation between the increase of average precision and the increase of LF1 utility is clear, but not perfect. In our system, the TF formula has a more complicated influence than just contributing to the scoring function, as it also affects the Rocchio term vector updating. Further study is needed to fully understand the effect of TF in filtering. However, it is clear that a certain minimum ranking accuracy is required if we are to expect positive utility values. If this minimum cannot be achieved it is better (in terms of utility) to return no documents at all.

3.2.3 Effect of vector updating

In our adaptive filtering model, updating the profile takes two steps. First, the term vector is updated to improve the ranking/scoring accuracy; second, the threshold is updated to optimize the decision cutoff point. Presumably, the (sparse) feedback information can benefit both vector updating and threshold updating. However, it is not a-priori clear that the score bias introduced by vector updating (because of the training documents receiving a higher score) as no negative effect on the effectiveness of threshold updating. Thus, it is interesting to evaluate whether the combination of vector and threshold updating improves the performance over either component separately.

We compared the official run CL99afL1a with an unofficial run that is the same as CL99afL1a, except that no vector updating was performed (i.e., only doing threshold updating), and found that using vector updating leads to occasionally significant changes in utility, which for some topics even turned a negative utility to a positive one. This suggests that the vector updating has indeed helped improve the ranking/scoring accuracy and thus made it possible to set a threshold to produce a positive utility.

3.2.4. Effect of dynamic Rocchio coefficient

One parameter in the Rocchio algorithm is the coefficient applied to the positive centroid vector. This coefficient controls how much weight we put on the centroid as opposed to the original term vector constructed from the topic description. Intuitively, we want to trust the centroid more if the centroid is computed using more examples. Thus, in some of our official runs, we tried a heuristic dynamic coefficient that is greater when there are more positive examples to learn from. Our official runs CL99afL1c and CL99afL1d differ only in that CL99afL1d uses a fixed coefficient (0.1), while CL99afL1c uses a dynamic coefficient, ranging from 0.1 to 2.0. The two runs are compared in Figure 3 along with the corresponding difference in average. We can see a certain correlation between them, but there are also cases where a decrease in utility accompanies an increase in average precision, indicating a possibly harmful interaction between scoring and threshold updating.

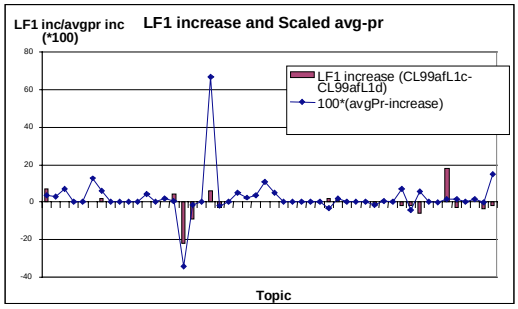

Figure 3. Utility difference vs. avg. precision difference

This effect may be due to the fact that a larger coefficient tends to increase the score bias, as, indeed, the recent training examples receive a higher weight in the centriod. As a result, the trained threshold is too high. Thus, a higher coefficient is less desirable from the point of view of threshold learning, as any benefits resulting from the use of an adaptive coefficient may be cancelled out. As evidence to support this, we observe that the average number of documents accepted for the dynamic coefficient is less than that for a fixed coefficient (7.4 vs. 10 documents/topic).

4 Batch Filtering Experiments

Our batch filtering experiments were designed mainly for validating our threshold updating mechanism. We did no preliminary experiments, but instead simply copied the parameters for threshold updating from our adaptive filtering runs. The initial threshold was trained using FT92, also with the same threshold setting parameters. To see how much value our adaptive threshold setting algorithm can add on top of initial training over FT92, we compared the batch filtering results with an alternative LF1 experiment in which no updating of either vector or threshold was performed over FT93-94. Updating turned out to be better for 16 topics and less good for 19 topics. For the remaining 15 topics there was no difference. The average LF1 utility value is better for the updating run (CL99bfL1) by 0.7.

This difference is disappointing, although, for some topics, adaptive updating does in fact significantly increase performance. For example, topic 389, exhibiting the largest increase, jumps from 83 to 122. The updating run accepted more documents for this topic than the run without updating (44 vs. 31) suggesting that our initial threshold may have been too high, and the adaptive updating helped to lower it over time.

Unfortunately, our experimental setup had a few mistakes with large consequences. For example, the runs were unable to make use of the FT92 training examples during the adaptive updating phase, and looked only at examples collected from the testing stream (FT93-94). This problem could explain the lack of benefit from adaptive filtering. In

addition, our run for LF2 used the wrong initial threshold (optimized for LF1).

Even so, some conclusions and suggestions appear warranted, based on an examination of those topics for which the system performed adequately. One observation is that our threshold setting may be generally too conservative (too high). If this is the result of score bias (artificially high score on the training documents), then increasing the value of beta in the beta-gamma algorithm should help here. Another possibility is to divide the training data in two parts, one of which is used for vector training, and the other for threshold setting. The penalty for this approach, of course, is a less effective use of training examples.

Many more analysis and experiments suggest themselves, and they are on our list for future work.

5 Summary and Further Work

We explored two approaches to improving adaptive filtering performance given the threshold setting algorithms we proposed in TREC-7. First, we tried to allow more frequent and profile-specific updating in the hope that this would result in a more efficient use of the sparse examples available in adaptive filtering. Second, we tried to improve the other two filtering system components—the scoring function and the term vector learning module—independently, with the additional aim of gaining understanding of how these two components interact with the threshold component.

Our results show that the more frequent profile updating does indeed consistently improve filtering performance. Our results also show that the improvement of scoring and term vector learning module generally leads to an improvement of filtering performance, but the relationship is more complex than can be understood from these experiments alone. One clear conclusion is that term vector updating increases the benefit of threshold updating and results in an often significant increase in utility. The effect of the TF formula is less clear: some topics are more strongly affected than others, and differences occur both in scoring and term vector learning. Finally, the term coefficients used in0 Rocchio learning do not seem to have a great impact on performance.

Based on these experiments and our experiments in TREC-7, we believe that our threshold setting and updating approach is effective, robust, and practically useful. In fact, our method of adaptive filtering is very general. Our approaches to establishing thresholds—both initial thresholds (using delivery-ratio threshold setting) and dynamically updated thresholds (using beta-gamma threshold learning)—could be applied in any vector-space filtering system and combined with any document-profile scoring function, as well as any term vector learning method. Moreover, our methods are also general enough to work with any utility function, including non-linear utility functions. An optimal choice of parameters may depend on the context, but we have reason to believe the general approach is robust.

In our future research, we intend to explore other possible threshold updating methods, e.g., based on logistic regression. We also hope to pay some more attention to the batch filtering task, and to study how to optimize the beta-gamma threshold regulation method in that setting.

References

[**Evans & Lefferts 1995**] Evans, David A., and Robert G. Lefferts, "CLARIT–TREC-Experiments". *Information Processing and Management*, Vol. 31, No. 3, 1995, 385–395.

[**Hull 1999**] Hull, David A., "The Trec-7 Filtering Track: Description and Analysis". In Voorhees, E.M., and Harman, D.K., (Editors), *The Seventh Text REtrieval Conference (TREC–7)*. NIST Special Publication 500–242. Washington, DC: U.S. Government Printing Office, 1999, 33–56.

[**Milic-Frayling et al. 1998**] Milic–Frayling, Natasa, Chengxiang Zhai, Xiang Tong, Peter Jansen, and David A. Evans, "Experiments in Query Optimization, the CLARIT System TREC–6 Report". In Voorhees, E.M., and Harman, D.K. (Editors), *The Sixth Text REtrieval Conference (TREC–6)*. NIST Special Publication 500–240. Washington, DC: U.S. Government Printing Office, 1998, 415–454.

[**Robertson et al. 1994**] Robertson, Steve E., et al. "Okapi at TREC3". In Harman, D.K., (Editors), Overview of *the Third Text REtrieval Conference (TREC–3)*. NIST Special Publication 500–226. Washington, DC: U.S. Government Printing Office, 1994, 109–126.

[**Robertson & Walker 1994**] Robertson, Steve E., and S. Walker, "Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval". In: Croft, W.B. and van-Rijsbergen, C.J., (eds), SIGIR'94, Dublin, 1994, 232–241.

[**Rocchio 1971**] Rocchio, J.J., "Relevance Feedback in Information Retrieval", In: Salton, Gerard (Editor), *The SMART Retrieval System*, Prentice–Hall, Englewood NJ. 1971, 313–323.

[**Salton 1988**] Salton, Gerard, *Automatic Text Processing*, Addison–Wesley, Reading, MA, 1988.

[**Zhai et al. 1999**] Zhai, Chengxiang, Peter Jansen, Emilia Stoica, Norbert Grot, and David A. Evans, "Threshold Calibration in CLARIT Adaptive Filtering", In Voorhees, E.M., and Harman, D.K., (Editors), *The Seventh Text REtrieval Conference (TREC–7)*. NIST Special Publication 500–242. Washington, DC: U.S. Government Printing Office, 1999, 149–156.