Threshold Calibration in CLARIT Adaptive Filtering

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

CLARITECH Corporation
A Justsystem Group Company

Abstract In this paper, we describe the system and methods used for the CLARITECH entries in the TREC–7 Filtering Track. Our main aim was to study algorithms, designs, and parameters for Adaptive Filtering, as this comes closest to actual applications. For efficiency's sake, however, we adapted a system largely geared towards retrieval and introduced a few critical new components. The first of these components, the *delivery ratio* mechanism, is used to obtain a profile threshold when no feedback has been received. A second method, which we call *beta–gamma regulation*, is used for threshold updating. It takes into account the number of judged documents processed by the system as well as an expected bias in optimal threshold calculation. Several parameters were determined empirically: apart from the parameters pertaining to the new components, we also experimented with different choices for the reference corpus, and different "chunk" sizes for processing news stories. Gradually increasing chunk sizes over "time" appears to help profile learning. Finally, we examined the effect of terminating underperforming queries over the AP90 corpus and found that the utility metric over AP88–AP89 was a good predictor. All of the above innovations contributed to the success of the CLARITECH system in the adaptive filtering track.

1 Introduction

Filtering in general, and adaptive filtering in particular, is one of the most challenging problems in information retrieval.

This year's TREC Filtering track was redesigned to accommodate a more realistic evaluation of practical filtering systems. One major difference was the absence of initial training information in the Adaptive Filtering task. Another was the more realistic nature of feedback: judgments were provided only for documents accepted by the system, and restrictions were placed on information available at the time of that decision.

These changes necessitated important modifications to our Filtering Evaluation system from previous years. Our goal was to evaluate the basic CLARIT adaptive filtering approach, which is based on standard CLARIT retrieval and routing techniques. [1,2,3,4] While the CLARIT system can be (and has been) extended to support real–time filtering—processing each incoming document in real time—we could not afford the time to adapt such a real–time filtering system for TREC evaluation. Therefore we used our standard CLARIT retrieval and profile training mechanism to make batch filtering decisions and perform batch updating on a succession of "chunks" of source documents. This improved efficiency but at the cost of some precision and sensitivity, as any feedback information can only be applied from the next chunk on.

Our basic approach to filtering still involves a two–step procedure similar to the one used in many other systems. For each document–profile pair, we compute a relevance score and then apply a score threshold to make the (binary) decision to accept or reject the document. Therefore, in this paradigm the two most important technical procedures to be worked out are scoring and threshold setting.

For our TREC–7 Filtering Track experiments, we decided to focus primarily on the problem of threshold setting, in large measure because (1) we did not understand it as well as the problem of scoring, and (2) it may have the greater impact on perceived performance (utility). The threshold-setting problem can be subdivided into two parts: (a) initial threshold setting, before there are any relevance judgments from the user; and (b) threshold updating, at any point when relevance judgments are fed back to the system. We used different techniques to set an initial threshold and to update the threshold during filtering.

Although we participated in both the adaptive filtering task and the batch filtering task, our focus was on adaptive filtering. We made two submissions for each utility measure for adaptive filtering. The first submission for each utility represented an optimal threshold parameter configuration as determined in our preliminary experiments. The second submission differed from the first only in that we adopted the rather user–unfriendly strategy of refusing any documents from AP90 for those topics that have an accumulated negative utility over AP88 and AP89. A comparison of the two submissions allows us to see how well a negative training utility can identify "difficult" topics.

In the following section, we describe our general procedure for adaptive filtering experiments. In Section 3, we discuss our main new algorithms, for initial threshold setting and for threshold updating. Our parameter space and the parameter settings we found to be best in pre-TREC runs are described in Section 4. Section 5 reviews our results and findings based on the experiments. Batch filtering, though not our main focus, nevertheless led to interesting insights and is discussed in Section 6. Finally, in Section 7, we summarize the main points and our plans for further work.

2 Adaptive Filtering Experimental Procedure

Conceptually, a profile (i.e., a binary document classifier) consists of three elements: a term vector, IDF statistics, and a score threshold. The first two are used to assign a score to any document, and the third is used to make the binary decision whether to accept the document.

The initial profile term vector for each topic was created automatically by parsing the original topic descriptions. We used all the fields (except the definition field, if any) in the topic description. The initial IDF statistics were derived from an unrelated reference corpus (*Wall Street Journal* 1987[1]). The initial profile threshold is set using the delivery ratio method described in the next section.

Source documents (i.e., the AP data) are segmented into a number of chunks, possibly of different sizes. Each chunk is indexed on noun phrases and individual words using the standard CLARIT phrase indexing technique. [1,2,4] Chunks are processed sequentially. At each chunk, we iterate over all the profiles, and run each profile as a query over the current chunk corpus. The matching function is a vector space dot product over terms in the profile space. To avoid using statistics based on "future" information, including the current chunk, we only used IDF statistics based on "earlier" chunks for matching (or WSJ87 data for the first two chunks — see Table 1). All the documents scored above the profile threshold are accepted.

Relevance judgments for the accepted documents are then obtained and the current chunk is used as a training corpus to update each profile independently.

Updating consists of two stages. The first stage involves term vector updating (i.e., expansion). We used the same general procedure that we employed in our TREC-6 Routing experiments. [4] Rocchio feedback, on relevant documents only, is used to expand the current term vector. [5,6] Specifically, the centroid vector of the relevant document vectors is computed and the terms are ranked by their centroid weight.[2] The *K* best–ranked terms are selected. Unlike standard Rocchio feedback, we assign a uniform weight to the selected terms before merging them into the current term vector. *K* grows heuristically with the number of relevant documents (*N*) available for training, according to the function $K = 10 + 10 * \log(N + 1)$.

The second stage involves threshold updating (i.e., re-estimating a threshold for the new term vector to be used when processing the next chunk). We use a method we call "beta-gamma regulation" to set a threshold for the new vector based on the current chunk (as an approximation of the next chunk), the future matching IDF statistics (from the "seen" documents up to the current chunk), and the partial relevance judgment on the current chunk. The details of this method are described in the next section.

The new, updated profile is then used to process the next chunk, and the above process is repeated until the last chunk (i.e., AP90) has been handled. Finally, the accepted documents for all the chunks are combined as the results for evaluation.

Note that a drawback of retrieval over chunks is that relevance information cannot be used immediately (according to the needs of each profile independently). If the threshold has been set inappropriately, there is no way for the system to correct this until the end of the chunk, at which time considerable damage may have been done to the performance.

3 Threshold Setting and Threshold Updating

To estimate an *initial profile threshold,* we used a new method, which we call the "delivery ratio" method. The rationale behind this method is that, in the absence of evidence pertaining to document relevance scores and stream topic density, a plausible utility metric may be the number of documents delivered to a user. A threshold can be set to best approximate the desirable number of documents to deliver. For a given time period, a desirable amount of delivery can be projected to a delivery ratio based on an estimate of the stream volume. A small reference corpus can be used to estimate an approximate threshold score at which the desirable ratio would be achieved.

Specifically, assume that the user wants to have a certain fraction (*r*), say 10% of the news delivered, we can run the profile vector as a query on the reference corpus using the same IDF statistics as would be used for matching future documents. The delivery ratio threshold is set to the score of the *K*-th document in the ranked list of retrieved documents, where $K = r*N$ and *N* is the number of documents in the reference corpus. In special cases when $K < 1$ or *K* is larger than the size of the list of all matched documents, heuristic extrapolation is applied.

For *threshold updating* we used beta-gamma adaptive threshold regulation. This technique selects a threshold, θ, by interpolating between an "optimal" threshold, θ_{op}, and "zero" threshold, θ_{zero},.

The *optimal threshold* is the threshold that yields the highest utility, given the newly updated term vector, over the accumulated training data. 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.

There are two reasons for believing that the "optimal" threshold our training procedure derives from the training data serves as an upper bound for the threshold, and is biased towards higher values. First, only an incomplete set of documents has been judged. As all un-judged documents are assumed to be non-relevant, the true optimal threshold, assuming complete knowledge of relevance judgments,

[1] Note that we avoided using any data from the time period covered by the test data, as these data might have had some overlap and would not have been available in a real application.

[2] While normalized TF is often used in the CLARIT system, we used the raw within-document frequency for Rocchio feedback here. Our goal in doing this was to emphasize TF over IDF in the presence of very few relevant training examples.

could only be lower and never higher than the estimated optimal threshold. Second, the scores of the positive training examples tend to be higher than the expected score of any randomly selected relevant document, since the term vector as trained with training examples favors the terms in the training documents. In other words, using the same training data for vector updating and threshold setting may lead to over–fitting. In addition, for learning and experimentation (especially in the beginning), we want to use a threshold somewhat lower than the true optimal threshold, even should we be able to estimate its value accurately.

At the lower end of the range, preliminary experiments indicated that using a zero utility threshold as a lower bound is a safe procedure, even though it is theoretically possible that the actual optimal threshold is lower still.

To obtain an actual threshold to use for a profile, we interpolate between θ_{opt} and θ_{zero}. Our pre-TREC experiments were geared towards finding an appropriate interpolation scheme. We first experimented with simple linear interpolation, using a constant parameter α, and called this method "alpha regulation" where α plays the role indicated in the following formula.

$$\theta = \alpha * \theta_{zero} + (1-\alpha) * \theta_{opt}$$

After several experiments, and some study of the method's behavior, we decided to express α as a function of two further parameters, β and γ, related to the two factors in the threshold bias identified above. We postulated the following formula, in which M is the number of judged training documents.

$$\alpha = \beta + (1-\beta) * e^{-\gamma * M}$$

In writing α in terms of β and γ, we attempt to capture both aspects of the bias present in the optimal threshold calculation: (1) β is a score bias correction factor that compensates for the relatively higher scores of relevant documents in the training corpus, and (2) γ expresses our belief that the estimated optimal threshold approximates the true optimal threshold more closely when more training examples are available. Note that γ is the inverse of the number of documents at which we place the threshold at approximately the midpoint of our range. If fewer than $1/\gamma$ training examples are available, the threshold will be somewhat lower; if more, somewhat higher.

Figure 1 illustrates the idea behind the formulas graphically.

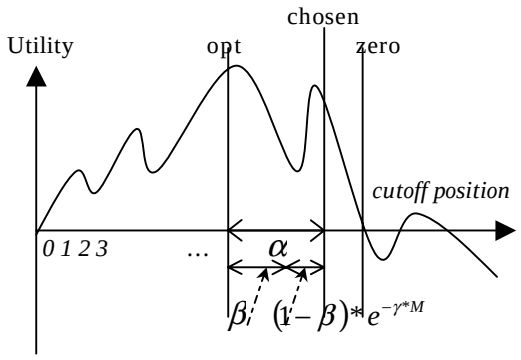


Figure 1. Beta-gamma regulation parameters.

Given a ranked list of all the documents in the training database sorted by their scores, their relevance, and a specific utility criterion, we can plot the utility value at each different cutoff position. Each cutoff position corresponds to a score threshold. Figure 1 shows how a choice of alpha determines a cutoff point between the optimal and the zero utility points, and how β and γ help us to adjust alpha dynamically according to the number of judged examples in the training database.

4 Configuring Experimental Parameters

In this section, we describe the values of several parameters of our system led to the best performance in our preliminary experiments. The parameters with the largest impact were the initial profile threshold, the document chunk sizes, subdocument size, and the β and γ factors used in threshold updating.

Delivery Ratio. The threshold for the initial profiles was set using our *delivery ratio* method, with a ratio of 0.0005, i.e., 1 out of 2,000 documents. A collection of all available 1987 Wall Street Journal articles was used as a reference corpus, approximating somewhat a possible earlier news stream.

Chunk Size. Another main parameter (though a direct consequence of our approximate method to simulate real–time filtering) was the *size of successive chunks* of news articles (roughly corresponding to periods of time over which news is accumulated). Using smaller chunks tends to be more "robust" as this limits the damage from a bad threshold in the overall utility. It also provides more flexibility in the presence of changes. But as smaller chunks contain fewer examples, they may provide less reliable profile learning and be overly sensitive to random fluctuations (we only used the examples in the previous chunk for updating).

In the first stages of learning, both the term vector and threshold are less reliable, and smaller–sized chunks are preferable. In addition, it is sometimes useful to lower the threshold to boost the number of judged examples presented to the system to speed up initial learning. This introduces the risk that many non–relevant documents might be accepted. In later stages, the profile can be assumed to be more stable, and the threshold more reliable. Hence larger chunks are to be preferred for better training.

In fact, our preliminary experiments with *Wall Street Journal* data bore out these hypotheses: chunks of increasing sizes generally led to better performance than chunks of equal sizes. Our post–TREC experiments confirm that using increasing–size chunks helps learning on AP as well.

In practice, we segmented AP88 and AP89 into 15 chunks with increasing sizes starting at 3,000 articles and going up to over 20,000 articles. Our hope was that both the term vector and the threshold would become stable enough to handle the AP90 collection as one chunk.

To simulate the accumulation of information about the news stream over time, we pre–built the reference corpus for each chunk (used for matching IDF statistics) so as to provide a compromise between availability, recency, size, and convenience. This resulted in the following arrangement.

Current Chunk	**Reference corpus (IDF) used**
Chunk 1 (3,000)	WSJ87
Chunk 2 (3,000)	WSJ87
Chunk 3 (4,000)	Chunk 1 + Chunk 2
…	…
Chunk 8 (9,000)	Chunk 1 + … + Chunk 7
Chunk 9 (10,000)	Chunk 1 + … + Chunk 8
Chunk 10 (12,000)	Chunk 1 + … + Chunk 9
Chunk 11 (14,000)	Chunk 1 + … + Chunk 10
Chunk 12 (17,000)	Chunk 1 + … + Chunk 11
Chunk 13 (20,000)	Chunk 1 + … + Chunk 11
Chunk 14 (24,000)	Chunk 1 + … + Chunk 11
Chunk 15 (22,597)	Chunk 1 + … + Chunk 11
Chunk 16 = AP90	AP89

Table 1. Size of source and reference chunks

Subdocument Size. Another parameter we varied in our experiments is the *subdocument size* used for indexing. Although intuitively subdocument indexing is appealing, preliminary experiments indicated that indexing on whole documents performed better, though only slightly.[3]

Threshold Regulation. The beta–gamma threshold regulation method was used to set a new threshold using the formula described in Section 3. In our preliminary experiments with *Wall Street Journal* data, we explored a large space of beta and gamma values and found that the best performance was fairly consistently reached, for both F1 and F3, at a setting of $\beta = 0.1$ and $\gamma = 0.05$. We used this setting in all our official runs.

5 Analysis of Adaptive Filtering Results

As a general trend, participating systems did relatively poorly for AP88, much better for AP89, and again somewhat worse for AP90. This effect can generally be attributed to an inherent instability in experimenting over part of the AP88 corpus (to train the profile and set the threshold), attained stability in performance for the rest of AP88 and AP89, and perhaps deteriorating stability (and the use of more defensive strategies) for AP90. This would indicate that most systems did indeed learn. For F1 (and a fortiori for F3) it turned out to be possible to obtain an overall positive average utility.

CLARITECH submitted four runs for adaptive filtering, two per utility. Except for F1 for AP88, our runs ended up at the top of participating systems. In this section we try to identify the factors that contributed to this result.

Apart from an indication that our system works satisfactorily, that Rocchio is a dependable term selection method, and that we did not make major errors, we can offer the following observations, the first three of which we discuss further in separate subsections:

1. Eliminating "bad" topics from consideration for AP90 yields a significant benefit for F1, and does not harm F3.
2. The more complicated beta-gamma regulation algorithm is better than a simple alpha regulation.
3. The use of increasing chunk sizes helps learning.
4. The delivery ratio method, with a conservative initial parameter setting, is a good method for initial threshold setting in the absence of training data.

5.1 Topic Elimination

For certain topics, for example those with only a few relevant documents in the news corpus, it is not possible to obtain a good profile (i.e., a profile for which the precision is > 0.4 for F1, or > 0.2 for F3).[4] For such topics, the highest utility (viz., 0) is achieved by not accepting any documents at all.

The simplest criterion we could think of was whether the total training utility over AP88 and AP89 was positive or not, and though this gives somewhat conservative results, we were not able to find better predictors.

To assess the benefit of this technique, we submitted two runs that differed only in whether they eliminated topics for AP90 or not. For F1, this approach was invasive, but very beneficial: half of the topics (25) were eliminated from consideration, resulting in an approximately 40% reduction in accepted documents and a 266 point increase in the total utility score for all 50 topics (302 vs. 36, i.e., an eight-fold increase in average utility!). Of the 25 eliminated topics, 16 would indeed have accumulated a negative utility (average value of –19.5) over AP90, whereas the remaining 9 would have contributed positive utilities (average value of 5.11). In comparison with the other groups, the median was tied or exceeded 10 more times, and the (zero) maximum 13 more times.

For F3, the impact was less substantial. Sixteen topics were eliminated, resulting in a reduction in the number of accepted documents by approximately 10%, and a total utility increase by 15 points, i.e., by less than 1%. Only 8 of the 16 topics showed a benefit (11 points on average), whereas 7 topics would have accumulated an average positive utility of 10.4. Here both the median and the maximum were tied or exceeded for four more topics.

[3] In practice, whole document indexing is achieved by using a very large subdocument size as a parameter for the CLARIT indexing procedure.

[4] This can be because of many noisy non–relevant documents or ineffective training.

5.2 Beta–Gamma Threshold Regulation

Before attempting a finer control over the threshold adjustments, we used simple linear interpolation with a constant coefficient α (alpha regulation). In this section we compare the average utility of this method with the results from use of the beta–gamma algorithm.

A comparison of average total utility per topic over AP88 and AP89 for our best runs using alpha regulation and our best runs with beta–gamma regulation is given in Table 2.

Average utility over AP88–89	F1	F3
Alpha (best)	6.98	54.96
Beta–Gamma	10.46	72.34
Increase	3.48 (50%)	17.38 (32%)

Table 2. Comparison of alpha, and beta–gamma regulation

Another interesting observation was that the best setting for beta–gamma also was less sensitive to small changes than the best setting for alpha. Furthermore, we found the maximum for beta–gamma (for the settings of our submissions, viz, β = 0.1 and γ = 0.05) to be stable even across databases (*Wall Street Journal* data as well as AP data).

Figure 2, below, demonstrates another aspect of the superiority of beta-gamma regulation. Shown in the graphs are the average utilities for equal–size chunks for our best runs with the respective methods. From these graphs, we learn that the difference between the two methods is most pronounced for "bad" chunks. In other words, beta–gamma regulation appears to be more stable than alpha regulation for both F1 and F3.

5.3 Learning Factors and the Effect of Chunk Size

A difficult but not unimportant question is how to evaluate the extent to which the system has improved the topic profiles over time as a result of learning. Two aspects of learning can be considered: improvement in scoring and improvement in threshold setting.

One way to assess the learning effect is to compare the actual utility scores obtained for each chunk with the maximum possible utility given the current term vector. The maximum possible utility is a measure of the quality of scoring, which is related to term vector training. How well the actual utility approximates the optimal utility, on the other hand, indicates the quality of threshold setting.

The actual and maximum average utilities for each chunk are shown, together with their ratios, in the graphs given in Figures 3a and 3b. In each case the comparison is shown for both F1 and F3. In Figure 3a, equal–sized chunks were used, and Figure 3b corresponds to our official run with increasing–size chunks. In the latter case, utility values are normalized with respect to the size of each chunk.

We see that in all cases there seems to be a *gradual small decrease* (at least not an increase) in optimal utility value. Although this may indicate that we obtain little benefit from profile training, it may also mean, for example, that the relevant documents are not evenly distributed over the stream, or that confusing non-relevant documents start appearing later on.

On the other hand, whereas actual and optimal utility values remain far apart (ratio relatively constant or even decreasing) for equal–sized chunks, the ratio clearly increases for increasing–sized chunks, indicating a gradual improvement in threshold setting.

Though this confirms our intuition that it is better to use chunks of increasing size, we need to point out that several factors play confusing roles. It is important, for example, which reference corpus was being used for the IDF calculations. The decrease in scoring quality from chunk 2 to chunk 3 in our official run, for example, may in part be explained by the switch from a large reference corpus (WSJ87) to a reference corpus only 8% its size (first two chunks of AP88). Also, certain chunks contain strongly varying numbers of relevant documents for some topics, leading to increased variance of the average utility.

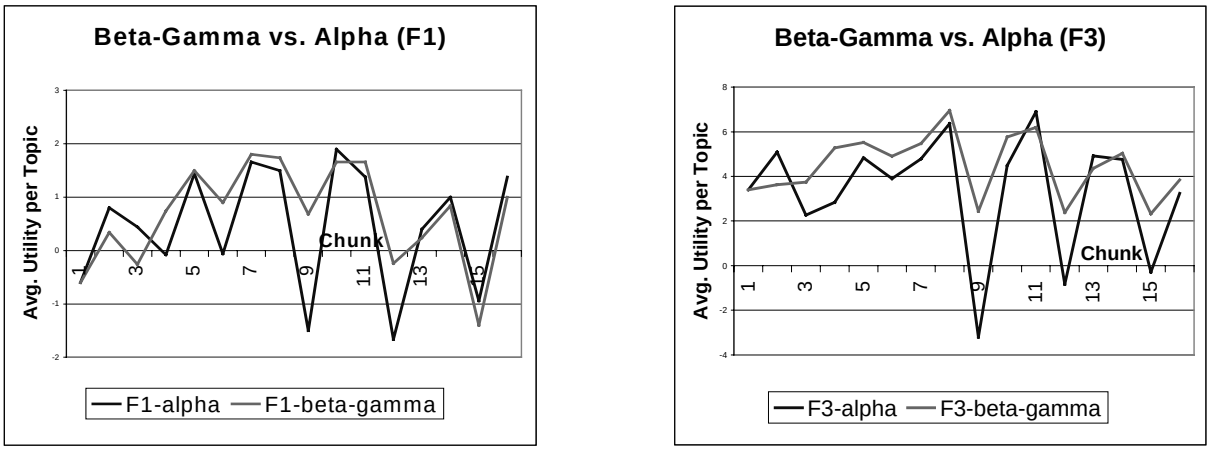


Figure 2. Comparison of chunk utility of beta–gamma regulation and alpha regulation

6 Batch Filtering

In this section we describe, more briefly, our experiments on batch filtering task.

6.1 Experimental Procedure

For Batch Filtering, we used essentially the same system that we used for Adaptive Filtering. The differences were as follows.

- As initial profiles, we took term vectors trained over AP88 instead of the vectors generated from the topic description.
- We used different chunk sizes.[5]
- AP88 was used as the initial reference corpus (for IDF, and delivery ratio).
- We used (standard) subdocument indexing.

As in adaptive filtering, we used the delivery ratio method to estimate an initial threshold. For this threshold estimation, the relevance judgments available for AP88 were not used.

6.2 Analysis of TREC-7 Batch Filtering Results

We submitted two runs (one each for F1 and F3) for the batch filtering task. Our runs were both clearly below median, though better for F3 than for F1.

By itself such a poor performance is not a surprise, as we did not exploit the complete set of relevance judgments on AP88 to establish a better initial threshold than the delivery ratio threshold. But a direct comparison showed that our adaptive runs over the same data would in fact have achieved a better performance, and instead of significantly below median, would have been very near median instead. There are then two questions:

1. Do the batch filtering runs perform worse as a result of lower profile quality (i.e., a problem with the training method)?
2. If not, what is the reason for the observed performance hit?

We tried to settle the first question by means of follow-up experiments.

6.3 Post–TREC Experiments

One possible hypothesis is that the profiles obtained by adaptive filtering at the end of AP88 are better than the profiles obtained from batch training over the same corpus. To test this hypothesis, we compared three different versions of initial profile vectors:

A. The original (untrained) profile vector ("NoTrain").
B. A trained profile vector based on adaptive filtering over judgments for accepted documents in AP88 ("AdaptTrain").
C. A trained profile vector using batch training over all judgments in AP88 ("BatchTrain").

Per topic, the initial threshold was kept constant. Evaluation was based on the average utility over all 50 topics over AP89 and AP90. We found that BatchTrain performed better than AdaptTrain, which was in turn better than NoTrain. This is as it should be, since BatchTrain uses more training examples than AdaptTrain, and NoTrain does not use any training at all.

Another way to evaluate relative performance is to compare the number of topics in which one method outperforms another. Again, AdaptTrain was clearly better than NoTrain, though now the difference between BatchTrain and AdaptTrain was less clear for F1.

We can conclude that the hypothesis stated above is false (the answer to our first question is "no") and, therefore, we need to look elsewhere for explanations of observed performance.[6]

6.4 Topic–By–Topic Analysis

We have not as yet performed a thorough topic–by–topic analysis, but such an analysis might prove to be very interesting in general. A classification of topics according to different criteria, and an analysis of which filtering methods work better for which class of topics can only lead to more insight and better filtering systems.

In the batch filtering context, a cursory inspection of relative per–topic performance showed that a big hit in performance was due to under–delivery for one topic in particular (topic 22). This topic was, with over 800 relevant documents, the topic with the highest density in relevant documents (and hence the highest median utility). Smaller hits occurred for other high-density topics. This under–delivery points at some specific characteristics of the methods we used, and is the result of a combination of the delivery ratio method and the beta–gamma regulation.

6.5 Discussion: Defects and Remedies

To explain this, we need to look into the beta–gamma regulation method in some more detail. A critical observation is that, when the threshold is set considerably higher than optimal for the topic, many relevant documents score below threshold. These documents are considered non–relevant by the system when it computes the optimal and zero–threshold. In such cases, the zero–threshold may be well above the true optimal threshold. Although the beta-gamma algorithm will lower the threshold in small steps, it may take many updates before the threshold approaches the true optimal threshold for the profile.

This phenomenon occurs in situation where the initial threshold is too high, for example because the estimated ratio of delivery was underestimated (forcing a higher score). This occurs precisely in high–density queries.

[5] AP89 was broken up in 6 chunks consisting of 5,000, 8,000, 11,000, 16,000, 20,000, and approximately 25,000 documents, respectively.

[6] Incidentally, the data do suggest that a better initial profile leads to more effective learning in later updating stages. This hypothesis certainly warrants further analysis.

Both our batch–mode and adaptive mode runs suffered from this effect. But the situation was much worse for batch filtering because there were many more threshold updates for the adaptive runs (16) than for the batch mode runs (6). In addition, our batch filtering system made no use of relevance information over AP88 for initial threshold setting, whereas some other systems did.

Although there are potential risks associated with a less conservative initial threshold setting, we could try to improve our system in the following ways.

1. Estimate the individual topic density from the AP88 corpus and use this density to obtain a different delivery ratio for each topic.
2. Use smaller chunks, or a real document–based filtering system to allow more rapid detection of and adjustment to underdelivery.
3. Use the shape of the utility function over a sorted list of accepted documents to estimate density. Although this function tends to behave rather chaotically, it may still be possible get a rough estimate of the topic density and take action in extreme cases.

Each of these aspects of the process suggests interesting directions for further study.

7 Summary and Further Work

We evaluated the basic CLARIT adaptive filtering approach by participating in the TREC–7 Adaptive and Batch Filtering tasks. Our results show that using our standard retrieval and routing techniques in combination with heuristic threshold setting leads to reasonably good performance. Three major positive contributors to this performance were (1) a heuristic beta–gamma threshold regulation algorithm, (2) the use of increasing chunk sizes, and (3) the elimination of difficult topics. Our results also suggest that the delivery ratio method is an effective initial thresholding method. Less clear at this point is the benefit of a better initial term vector.

In the future, we intend to study in more detail the behavior of the beta–gamma threshold regulation algorithm, in particular, how its effectiveness varies with different topics. One example is the problem of slow learning for high density topics that may have damaged our performance. Another is the possibility of a combination with logistic regression for density estimation, which showed some promise in our approach to filtering in TREC 6. [4]

We also intend to investigate actual real–time filtering algorithms, as well as profile–specific updating. Other interesting aspects of filtering are related to an intelligent exploitation of historical training data based, for example, on recency and confidence. Finally, we believe that topics related to the learning effect and behavior over time, such as user interest drift and topic tracking, are important future research issues.

Acknowledgements

We are indebted to Dr. Alison Huettner for her comments on earlier drafts of this paper and to Ms. Lisa Stewart for help with the layout and formating of the final document.

References

1. Evans, David A., Kimberly Ginther–Webster, Mary Hart, Robert G. Lefferts, Ira A. Monarch, “Automatic Indexing Using Selective NLP and First–Order Thesauri”. In A. Lichnerowicz (Editor), *Intelligent Text and Image Handling. Proceedings of a Conference, RIAO ’91*. Amsterdam, NL: Elsevier, 1991, 624–644.

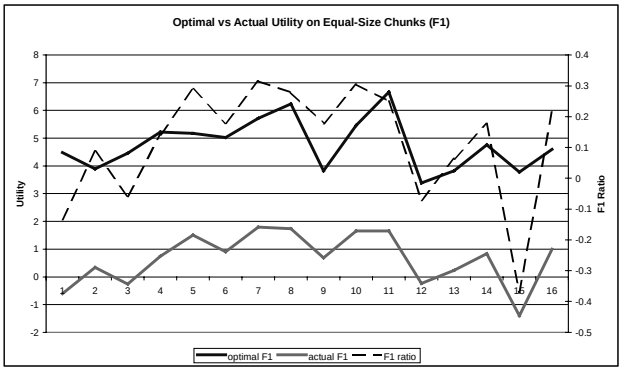
2. Evans, David A., and Robert G. Lefferts, “CLARIT–TREC Experiments”. *Information Processing and Management*, Vol. 31, No. 3, 1995, 385–395.

3. Evans, David A., Alison Huettner, Xiang Tong, Peter Jansen, and Jeff Bennett, "Effectivenss of Clustering in Ad-Hoc Retrieval," This Volume.

4. 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 D.K. Harman (Editors), *The Sixth Text REtrieval Conference (TREC–6)*. NIST Special Publication 500–240. Washington, DC: U.S. Government Printing Office, 1998, 415–454.

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

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



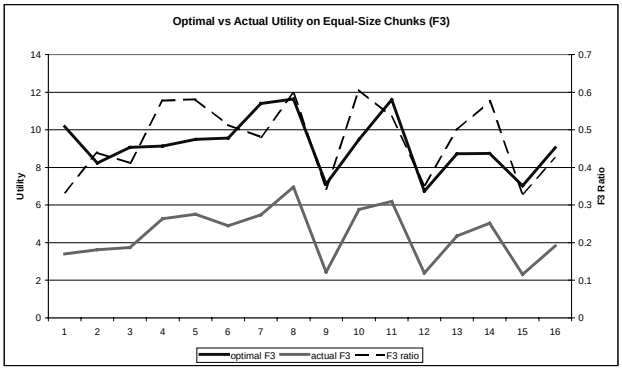
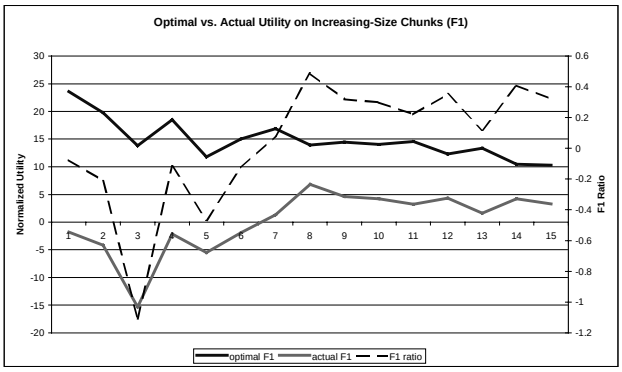


Figure 3a. Learning effect: optimal vs. actual utility for equal–size chunks (F1 and F3).



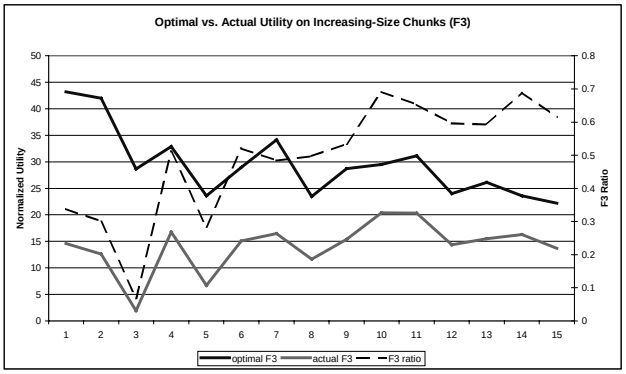


Figure 3b. Learning effect: optimal vs. actual utility for increasing–size chunks (normalized) (F1 and F3).