kNN, Rocchio and Metrics for Information Filtering at TREC-10[1]

Tom Ault TOMAULT@CS.CMU.EDU
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Yiming Yang YIMING.YANG@CS.CMU.EDU
Language Technologies Institute & Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

We compared a multi-class k-nearest neighbor (kNN) approach and a standard Rocchio method in the filtering tasks of TREC-10. Empirically, we found kNN more effective in batch filtering, and Rocchio better in adaptive filtering. For threshold adjustment based on relevance feedback, we developed a new strategy that updates a local regression over time based on a sliding window over positive examples and a sliding window over negative examples in the history. Applying this strategy to Rocchio and comparing its results to those by the same method with a fixed threshold, the recall was improved by 37-39% while the precision was improved by as much as 9%. Motivated by the extremely low performance of all systems on the $T10S$ metric, we also analyzed this metric, and found that it favors more frequently occurring categories over rare ones and is somewhat inconsistent with its most straightforward interpretation. We propose a change to this metric which fixes these problems and brings it closer to the C_{trk} metric used to evaluate the TDT tracking task.

1. Introduction

We participated in the TREC-10 information filtering track, submitting results of a standard Rocchio method and one variant of our k-nearest neighbor (kNN) algorithms[13] for the batch and adaptive filtering tasks. Our goals for this year's TREC were twofold:

1. To establish a performance baseline for text categorization on the new Reuters corpus.
2. To develop an effective adaptive thresholding technique for the adaptive filtering subtask.
3. To investigate the properties of the $T10S$ metric using the isocurve analysis in precision-recall space we developed for the TREC-9 metrics[1].

[1] Authors' names are in alphabetical order.

Item (1) was motivated by the success of kNN on previous versions of the Reuters corpus[10, 9] and is addressed by our batch filtering results, for which we applied the Avg2 variant of kNN. Item (2) was motivated by our difficulties in developing an effective adaptive thresholding method for TREC-9, and is addressed by our new *margin-based local regression* technique of adaptive thresholding. We were successful in both endeavors, ranking third of eighteen runs in the batch-filtering task and second of thirty runs in the adaptive filtering task for both the F_β and $T10S$ metrics[2]

Item (3) was motivated by the over 50% drop in performance for the $T10S$ metric when moving from the validation to the test set, compared to a drop of only 27% for the F_β metric. This observation combined with an analysis of the $T10S$ metric in terms of its isocurves in precision-recall space lead to the discovery of some inconsistencies between the user behavior that the $T10S$ metric appears to model and what it actually models. We propose a fix for $T10S$, and compare the modified metric to the weighted tracking cost (C_{trk}) metric used for the TDT tracking task.

This paper has five sections past the introduction. Section 2 reports the experiments with our kNN and Rocchio systems in batch filtering. Section 3 compares kNN and Rocchio in adaptive filtering, and introduces our novel approach for adaptive thresholding. Section 4 analyzes the potential problems inherent in the $T10S$ metric, suggests a minor alteration that resolves these problems, and discusses the relationships between the modified and unmodified $T10S$ and the F_β and C_{trk} metrics. Section 5 presents our conclusions and future research goals for information filtering.

2. Batch Filtering

We applied our kNN system and our implementation[13] of a standard Rocchio method to this task to compare these two methods.

[2] Rankings were computed by the authors across all runs submitted to the TREC-10 filtering tasks from official per-category, per-run performance data supplied by the filtering track coordinators; these rankings are not official.

2.1 K-Nearest Neighbor (kNN)

kNN, an instance-based classification method, has been an effective approach to a broad range of pattern recognition and text classification problems [2, 8, 10, 13]. In contrast to "eager learning" algorithms (including Rocchio, Naive Bayes, Decision Trees, etc.) which have an explicit training phase before seeing any test document, kNN uses the training documents "local" to each test document to make its classification decision on that document. Our kNN uses the conventional vector space model, which represents each document as a vector of term weights, and the distance between two documents is measured using the cosine value of the angle between the corresponding vectors. We compute the weight vectors for each document using one of the conventional TF-IDF schemes [4], defined as:

$$w_d(t) = (1 + \log_2 n(t, d)) \times \log_2(|\mathcal{D}|/n(t)) \tag{1}$$

where $n(t, d)$ is the within-document frequency of term t in document d, $n(t)$ is the total document frequency of term t in document set D.

Given an arbitrary test document d, the kNN classifier assigns a *relevance* score to each candidate category (c_j) using the following formula:

$$s(c_j, d) = \sum_{d' \in R_k(d) \cap \mathcal{D}_j} \cos(w_d, w'_d) \tag{2}$$

where set $R_k(d)$ are the k nearest neighbors (training documents) of document d. By sorting the scores of all candidate categories, we obtain a ranked list of categories for each test document; by further thresholding on the ranks or the scores, we obtain binary decisions, i.e. the categories above the threshold will be assigned to the document. There are advantages and disadvantages to different thresholding strategies [12].

2.2 Rocchio

Rocchio is an effective method using relevance judgments for query expansion in information retrieval[3, 5], and the most common (and simplest) approach to the filtering tasks in TREC[14].

The standard Rocchio formula computes a vector as the *prototype* or *centroid* of a class of a documents. Given a training set of documents with class labels, the *prototype* vector is defined to be:

$$\vec{c}(\gamma, k) = \frac{1}{|R_c|} \sum_{u_i \in R_c} \vec{u}_i - \gamma \frac{1}{\bar{R_{c,k}}} \sum_{v_i \in \bar{R_{c,k}}} \vec{v}_i \tag{3}$$

where R_c is the set of positive training examples, $\bar{R_{c,k}}$ is the "query zone"[6], that is, the k top-ranking documents retrieved from the negative training examples when using the centroid of the positive training examples as the query. To increase the efficiency of computation, we retain only the top p_{max} components of the prototype vector. The values of γ, k (the size of the local zone) and p_{max} are the prespecified parameters for the Rocchio method.

In the filtering process, Rocchio computes the cosine similarity between each test document and the prototype of every category, where the prototype is updated over time using the past documents whose category labels are available through relevance feedback. Thresholding on these scores yields binary decisions on each document with respect to every category.

2.3 Batch Filtering Results

In our experiments with Rocchio and kNN, we defined a token to be the longest possible sequence of letters and digits, followed by an optional "'s" or "'nt". Tokens which were purely numbers were discarded, as were common English words found on a stop word list. Tokens were stemmed with the Porter stemmer and assigned weights according to equation 1 above. Per-category thresholds for binary decision making were set by five-fold cross-validation on the training data.

We submitted two sets of results, labelled "CMUCATa2f5" and "CMUCATa210", for batch filtering; the former is optimized for the F_β metric and the latter the $T10S$ metric. Both runs used the kNN.avg2 method with $kp = 200$ (number of nearest neighbors which are positive examples of the category) and $kn = 500$ (number of nearest neighbors which are negative examples of the category), since this method and parameter settings had the best performance during cross-validation for both metrics. Table 1 summarizes the results.

Based on previous experience with the Reuters-21578 and OHSUMED corpora[9, 11], we applied a variety of feature selection methods, including document frequency, mutual information, information gain, and chi-square. None of them produced any significant improvement in the performance of our system on the Reuters 2001 corpus. Why we should see no improvement on this corpus while we see considrable improvement on the other corpora requires further investigation.

3. Adaptive Filtering

Our research strategy consists of two parts:

- analyzing the scores generated by kNN and Rocchio over time, to see which method produces more discriminatory scores for separating positive and negative examples of a category; and
- using *margin-based local regression* (our new approach) to track the potential shift of the optimal threshold for each category over time.

RUN ID	RECALL	PREC	$T10S$	F_β	RANK-$T10S$	RANK-F_β
CMUCATa2f5	0.322	0.719	0.287	0.511	7/18	3/18
CMUCATa210	0.358	0.618	0.324	0.489	3/18	7 or 8/18

Table 1. Results by CMU-CAT for Batch filtering

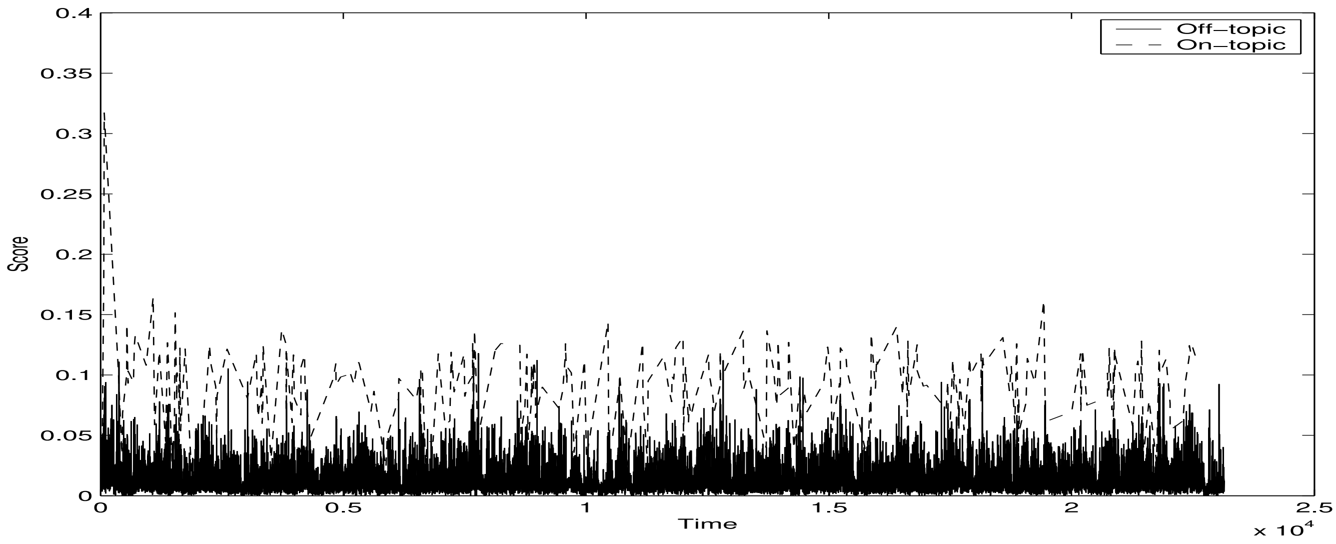

Figure 1. Scores vs. time for on- and off-topic documents by Rocchio for category R83

3.1 Score analysis for Rocchio and kNN

Figures 1 and 2 show the scores for category R83 ("METALS TRADING") generated by kNN and Rocchio during the adaptive filtering process where the relevance judgment for each test document was made available to the system after that document is scored, regardless what decision (Yes or No) the system made for that document; the category prototype was updated accordingly per test document. We used the TREC-10 training corpus for this experiment by splitting the data in to the training and test halves and then running the systems on these data. The scores generated in such a process, obviously, are better than those kNN and Rocchio would generate under the condition required in TREC for relevance feedback, i.e. relevance judgments are available for a system only for the documents which the system make a Yes decision upon. Nevertheless, those figures allow us to get a rough idea about a major difference between the scores generated by our two systems. For Rocchio, there is clear separation on average between the scores for the two classes (Yes and No) over time, while for kNN, the scores for the two classes are will blended. This means that finding the optimal thresholding function over time using the scores generated by Rocchio would be a much easier task than thresholding over the scores generated by kNN. Also, the average scores for each class by Rocchio seem to be constant in different time intervals; however, there is a visible trend (increasing in value over time) in the average scores by kNN, at least for the particular category being observed. In fact, we compared pairwise figures for kNN and Rocchio over all the 84 categories selected from Reuters 2001 for TREC-10, and observed similar patterns with most categories in the TREC-10 filtering training corpus.

These empirical findings were rather suppressing to us, because we have found kNN to perform better than Rocchio in batch filtering and conventional text categorization[9, 10]. On the other hand, we have also found Rocchio works surprising well (comparable or just slightly worse than kNN) for the event tracking task in the domain of Topic Detection and Tracking (TDT)[12], which is similar (but not identical) to adaptive filtering, in the sense that both processes start with a small number of training examples per class. Why does Rocchio perform worse than kNN in batch filtering but better in adaptive filtering? We do not have a satisfactory interpretation for this question at this point; deeper understanding about this invites future research.

3.2 Margin-based local regression

Here we propose a novel approach, namely *margin-based local regression*, for predicting optimal thresholds over time. The intuition is rather simple: if we have two streams of scores (one for previously-classified positive examples and the other for previously-classified negative examples for a particular class), and if the two streams are separable in value (Figure 1) in any particular time interval, then we would choose some values inside of the *margin* between the two streams as the thresholds, where by margin we mean the difference between the minimal score for positive examples

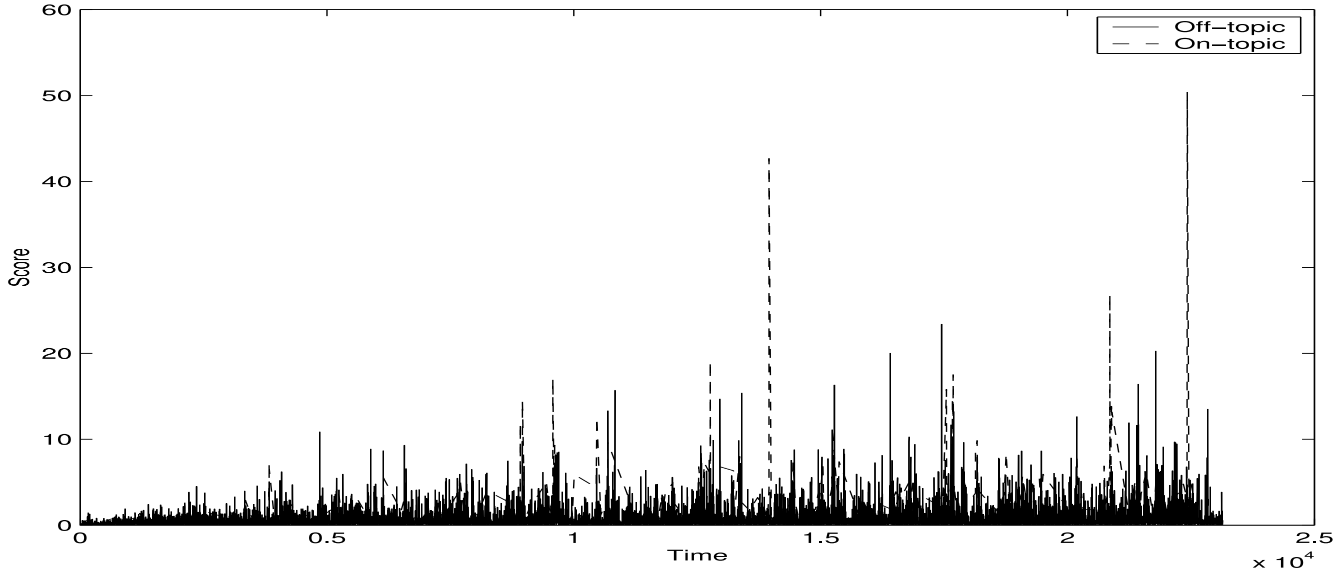

Figure 2. Scores vs. time for on- and off-topic documents by unnormalized kNN for category R83

and the maximal score for negative examples in a particular time interval. One can think of a waving band with both its center location and width changing over time in a two dimensional space of magnitude verses time. We want to use local regression to track the shift of the "optimal" threshold over time inside of the band.

This simple description is sufficient to gain intuition into the method, but is not sufficient for an accurate definition. First, we do not assume that the positive and negative streams are always separable, but artificially make those streams separable by excluding outliers. That is, for the positive stream, our system only includes the truly positive examples for which the system predicted YES. All other examples, including the false-alarms, misses and documents for which the system correctly predicted NO, form the negative stream; our system unfortunately cannot exclude the misses because the relevance judgments for those examples are not available during the relevance feedback. Second, we need to relax the definition of margin, to avoid making our method overly sensitive to outliers. Third, we need to be precise about what is "optimal", discussing the concern about risk minimizing. Fourth, we need specify the local regression for tracking the shift of optimal thresholds over time.

The precise definition of our approach uses the following notation:

- $X = \{x_1, x_2 \dots x_{k_+}\}$ is the sliding window of scores for the k_+ (at most) most recent positive examples for which the system predicted YES;
- $Y = \{y_1, y_2 \dots y_{k_-}\}$ is the sliding window of scores for the k_- (at most) false-alarms, misses, and documents for which the system correctly predicted NO;
- $\mu_x(t) = a_1 t + b_1$ is the local regression obtained by fitting a line over the data points in sample X, where a_1 and b_1 are the regression coefficients;
- $\mu_y(t) = a_2 t + b_2$ is the local regression obtained by fitting a line over the data points in sample Y, where a_2 and b_2 are the regression coefficients;
- $\delta(t) = \mu_x(t) - \mu_y(t)$ is the local *margin* (the Mean-Mean version; see the next section);
- $\theta(t) = \mu_y(t) + \eta\delta(t) = at + b$ is the local regression for optimal thresholding, where

$$a = \eta a_1 + (1 - \eta) a_2$$

$$b = \eta b_1 + (1 - \eta) b_2.$$

This method has five parameters, k_+, k_-, η, Δ_+ and Δ_- which are empirically chosen (through cross validation). We purposely allow k_+ and k_- to take different values (instead of a single parameter k) so that we can empirically tune the window sizes to be sufficient sensitivity to the local trends for both positive examples and negative examples. As for parameter η, it allows us to adjust the position (instead a fixed position, such as the middle) of the thresholding function between the margin, in order to overcome the inductive bias of the system (if any) and to optimize the performance with respect to different evaluation metrics ($T10S$, F_β, or the like) through cross validation. Δ_+ and Δ_- are the number of documents the window must slide through before the positive and negative margins are updated; in our TREC-10 results, we updated both margins with every document ($\Delta_+ = \Delta_- = 1$).

For initialization, we set $a_0 = a_1 = 0$ and set b_0 and b_1 to return the top 1% of documents in the validation set for each category. Early on, when there is not sufficient data to reliably compute the margins (defined in terms of number of documents within the window and parameterized by min_- and min_+ for the positive and negative windows respectively), we apply the following hueristic to set the threshold to a reasonable value without drawing too many false- alarms or misses: if both the positive window and the negative window have less than min_- and min_+ documents, the threshold is set to just above the score of the last false-alarm observed.

3.3 Variants of margins

We propose several versions of margin as the variants of our approach:

1. *Min-Max* margin:

$$x = arg\min\{x_1, x_2 \ldots x_{k_+}\}$$

$$y = arg\max\{y_1, y_2 \ldots y_{k_-}\}$$

$$\delta(t) = x - y$$

$$\theta(t) = \eta x + (1-\eta)y.$$

2. *Mean-Mean* margin:

$$\mu_x(t) = a_1 t + b_1$$

$$\mu_y(t) = a_2 t + b_1$$

$$\delta(t) = \mu_x(t) - \mu_y(t)$$

$$\theta(t) = \mu_y(t) + \eta\delta(t)$$

.

3. *MinK-MaxK* margin:
   X' is the bottom n_+ data points in $X = \{x_1 \ldots x_{k_+}\}$,
   Y' is the top n_- data points in $Y = \{y_1 \ldots y_{k_-}\}$,
   $\mu'_x(t) = a_1 t + b_1$ is the linear fit to X';
   $\mu'_y(t) = a_2 t + b_2$ is the linear fit to Y';

$$\delta'(t) = \mu'_x(t) - \mu'_y(t)$$

$$\theta(t) = \mu'_y(t) + \eta\delta'(t)$$

   where n_+ and n_- are pre-specified parameters.

4. *MeanVar-MeanVar* margin:

$$\delta(t) = (\mu_x(t) + \alpha\sigma_x) - (\mu_y(t) + \alpha\sigma_y)))$$

$$\theta(t) = \mu_y(t) + \eta\delta(t)$$

   where σ_x is the standard deviation of X, σ_y is the standard deviation of Y, and α is a pre-specified parameter.

5. Other combinations, e.g., *Mean-MaxK*:

$$\delta''(t) = \mu_x(t) - \mu'_y(t)$$

$$\theta(t) = \mu'_y(t) + \eta\delta''(t)$$

The Min-Max margin is the simplest, but likely to be over-sensitive to outliers and under-sensitive to the trend of the margin within a window. The Mean-Mean margin is the one we introduced in the previous section, which is less sensitive to extreme values than Min-Max. The MinK-MaxK has an sensitivity between the previous two, with additional (ad-hoc) parameters; in fact, Min-Max is just a specific case of MinK-MaxK in which $n_+ = n_- = 1$. MeanVar-MeanVar take the densities of data points on both sides (the positive side and the negative side) in to consideration, which would be more powerful than Mean-Mean but assumes normal distribution of the scores for the positive and negative examples and requires more data for the estimation of the variances. There are other possible variants along this line; we do not intend to give an exhaustive list.

3.4 Adaptive filtering results

We chose Rocchio over kNN for adaptive filtering. Tables 2 and 3 describe our submitted runs and the results, including one run (CMUCATmrf5) using Mean-Mean margin (as our primary submission), one run (CMUCATmr10) using the Mean-MaxK margin, and two runs for the baseline Rocchio (CMUCATsrf5 and CMUCATsr10) in which the the prototypes were adaptive but the thresholds were fixed. The last two runs were generated as baselines for comparisons with the margin-based adaptive filtering methods.

In addition to the submitted runs, we also tested other versions of the margins (MinK-MaxK, for example). We found the Mean-Mean method with the best results in cross validation over the TREC-10 training corpus. The Mean-MaxK, however, performed better on the evaluation data, suggesting that that variant tends to have a large performance variance.

We were surprisingly pleased by the improvements by the margin-based regression over the baseline Rocchio with a fixed threshold. Under the same condition of optimizing F_β, the Mean-Mean method improved the performance over the baseline by 37.5% (from 24.8% to 34.1%) in recall and 0.5% (from 65.8% to 66.1%) in precision. Under the condition of optimizing $T10S$, the Mean-MaxK improved the performance over the baseline by 38.7% (from 24.8% to 34.4%) in recall and 9.2% (from 60.3% to 65.7%) in precision. We are also surprised that the Rocchio baseline with a fixed threshold worked very well, being ranked at the top four among 30 submissions in both $T10S$ and F_β measure.

It is worth mentioning that the margin-based local regression approach is not a part of the Rocchio method. Instead, it can be applied to the output of any system as long as the average of scores for positive examples by that system are higher than the average of the scores for negative examples, and as long as there is some continuous trends over time in the margins. An interesting point is, when we designed this method and until our submission to TREC-10, we only tested Rocchio under the condition of complete relevance feedback (and did not have the time to run it under more re-

RUN ID	DESCRIPTION
CMUCATsrf5	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using fixed threshold, optimized for F_β
CMUCATsr10	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using fixed threshold, optimized for $T10S$
CMUCATmrf5	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using margin-based thresholding with means for both margins, optimized for F_β
CMUCATmr10	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using margin-based threshold with lower margin computed from median of top 20 negative examples and higher margin computed from mean of positive margin, optimized for $T10S$.

Table 2. Official submissions by CMU-CAT for the Adaptive filtering task

RUN ID	RECALL	PREC	$T10S$	F_β	RANK by $T10S$	RANK by F_β
CMUCATsrf5	0.248	0.658	0.211	0.467	7/30	4/30
CMUCATsr10	0.248	0.603	0.228	0.415	4/30	6/30
CMUCATmrf5	0.341	0.661	0.251	0.489	3/30	3/30
CMUCATmr10	0.344	0.657	0.263	0.499	2/30	2/30

Table 3. Results by CMU-CAT for Adaptive Filtering

alistic settings of relevance feedback); under that condition we did not found dynamic trends in the margins among the scores by Rocchio. We took this approach anyway because it was rational. The strong results, 37.5-38.7% improvement in recall while precision improved in the same direction over Rocchio baseline, suggest that, perhaps, there were indeed dynamic trends in the margins that worth tracking.

4. Metrics

	$T10S$	F_β
Batch filtering		
Minimum	0.081	0.154
Mean	0.239	0.429
Maximum	0.414	0.606
Adaptive filtering		
Minimum	0.015	0.046
Mean	0.134	0.266
Maximum	0.291	0.519

Table 4. Macro-average performance summary for the TREC-10 filtering task

The adaptive and batch filtering tasks for this year used two metrics: $T10S$ and van Rijsbergen's F_β[7], which are defined as with respect to a category C as:

$$T10S(C) = \frac{max(2A - B, minU) - minU}{2 * N_+ - minU} \quad (4)$$

$$F_\beta(C) = \frac{(\beta^2 + 1)A}{A + B + \beta^2 N_+} \quad (5)$$

where A is the number of documents correctly assigned to C, B is the number of documents incorrectly assigned to C (aka false-alarms), N_+ is the number of documents relevant to C, $minU$ is the lower bound on the unscaled utility $(2A - B)$, and β is a constant that specifies the relative weight between recall and precision for F_β. Both $T10S$ and F_β are scaled to fall between 0 and 1, and for TREC-10, $minU$ was fixed at -100 and β at 0.5 for all categories. The overall performance of the system for the task was obtained by computing the unweighted average across all categories (called the *macro-average* in the information retrieval literature).

The most straightforward interpretation of the $T10S$ metric is that it computes the return the user receives in terms of information gained vs. effort expended in reading the documents assigned by the filtering system to a particular category, scaled relative to the range of possible returns, where 1.0 represents maximum information gain with minimum effort, and 0.0 represents the point at which the effort required in reading the documents for the category in question so exceeds the information gained that the user regards any information contained in those documents as worthless. The F_β metric does not have such a straighforward interpretation in terms of the preferences of a particular user, but is instead the weighted harmonic average of recall [3] and precision[4] over the set of documents assigned to a category.

An examination of table 4 shows that systems participating in the batch and adaptive filtering tasks performed much worse on $T10S$ than on F_β. Does the $T10S$ measure, in fact, describe a harder task than the F_β measure or are there

[3] *Recall is defined for a category as the ratio of documents correctly assigned to that category to the total number of documents relevant to that category, e.g.* $r = \frac{A}{N_+}$.

[4] *Precision is defined for a category as the ratio of documents correctly assigned to the category to the total number of documents assigned to that category, e.g.* $p = \frac{A}{A+B}$.

other factors at work which would cause this performance gap? In the following section, we analyze the properties of the $T10S$ metric and find that, while $T10S$ is a definite improvement over $T9U$, it still has an undesirable characteristic that biases it against frequently-occuring categories. We propose a minor modification to $T10S$ which fixes the undesirable properties and brings it closer to the tracking cost (C_{trk}) metric used in the TDT evaluations.

4.1 T10S

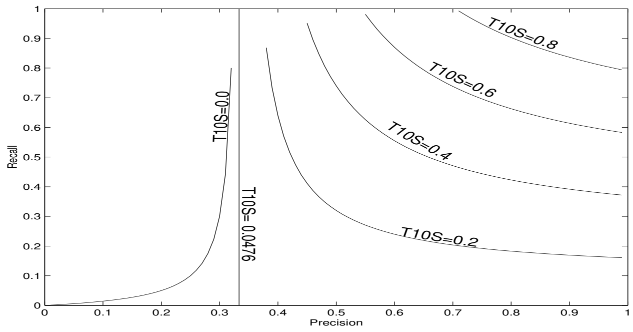

Figure 3. Isocurves of T10S for $\alpha = -0.1$

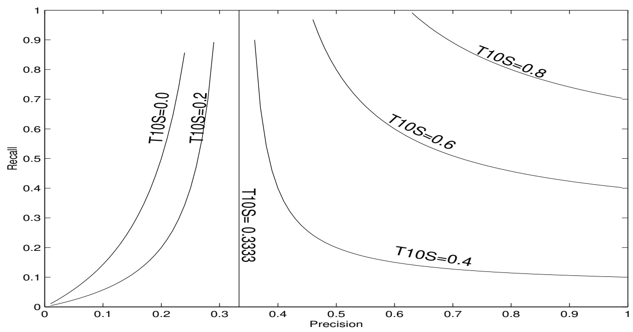

Figure 4. Isocurves of T10S for $\alpha = -1$

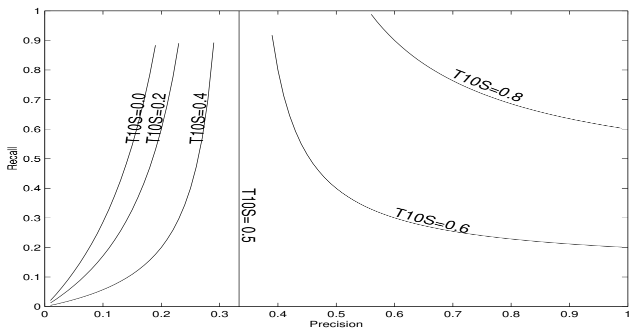

Figure 5. Isocurves of T10S for $\alpha = -2$

$T10S$ is a scaled version of the linear utility metric used in TREC-9 ($T9U$). This scaling addressed some of the problems of $T9U$ (which are discussed in detail in [1]), specifically, the wide variation in the maximum value of $T9U$ with the number of relevant documents for the category, which makes the macro-average of $T9U$ difficult to interpret and causes performance on common categories to dominate the average. However, the scaling introduced another problem, which can be seen if $T10S$ is written in terms of recall(r) and precision(p):

$$T10S = \begin{cases} \frac{max(r*\frac{3p-1}{p},\alpha)-\alpha}{2-\alpha} & \text{if } p > 0, \alpha < 2 \\ \text{Some value in } [0, \frac{\alpha}{2-\alpha}] & \text{if } p = 0, \alpha < 2 \end{cases} \qquad (6)$$
$$\alpha = \frac{minU}{N_+}$$

The isocurves of $T10S$ for $\alpha = -0.1$, $\alpha = -1$ and $\alpha = -2$ are shown in figures 3 through 5. The biggest problem with $T10S$ is that the locations of its isocurves are dependent on the number of examples of the category in question, giving two different categories with the same relative performance different $T10S$ values. In particular, the larger the number of relevant documents for a category, the lower it's $T10S$ score for the same relative performance compared to a category with fewer documents. Given the large number of categories with 5,000 or more relevant documents in the test set used for TREC-10, it's not surprising that the scores for this metric were significantly lower than those for F_β.

Another problem with $T10S$ is that it's lower bound of $minU$ is applied before the metric is scaled. This means that the user's tolerance for poor-performing profiles (as modeled by $T10S$) also varies with the number of relevant documents for the category. In particular, the user is far more tolerant of poor performance (in relative terms) for categories with fewer documents. Furthermore, the value of $minU$ controls not only the user's tolerance for poor performance but also how sensitive the locations of the isocurves are to changes in the value of N_+; the larger the magnitude of $minU$, the less sensitive $T10S$ is to variations in N_+. This linkage between the minimum return the user is willing to accept on his or her investment in reading documents before he or she gives up and the preference of the user for categories with fewer relevant documents is counterintuitive and not obvious from the form of the $T10S$ metric itself.

One could dismiss these objections to $T10S$ by claiming that it is not necessary for $T10S$ to have consistent properties across categories with respect to recall and precision, since $T10S$ is not based on those metrics. However, having inconsistent properties with respect to recall and precision leads to inconsistencies within the $T10S$ metric itself if we take the most straightforward interpretation described in section 4. Under this interpretation, each document on average requires equal effort to read and provides the same amount of information, which are reasonable assumptions given that the TREC filtering tasks only supply binary relevance judgements and no judgements about the effort required to read a document. Furthermore, the TREC filtering tasks make no distinctions among categories as to which might be more or less important to the user. These conditions imply that a reasonable user should expect to spend more effort reading documents for categories that occur more frequently than for those that occur less frequently, and that the metric used to model the user should take this into account. Moreover, two sets of documents assigned to two different categories with different occurence frequencies which have the same relative amount of information (e.g. same recall) and require the same relative effort to extract that information (e.g. same precision), should be regarded as equally useful to the user, since all documents require the same effort, all relevant documents have the same information content, and the user expects to spend more effort on the more frequently-occuring category. In violating this latter principle that equal recall and precision should yield equal utility, $T10S$ implicitly assumes that the user favors rare categories over more common ones, that the utility of relevant documents decreases as more of them are found while effort to read them remains the same, or that the effort required to read a document decreases in proportion to the number of relevant documents for a category while the utility of relevant documents remains the same. None of these latter assumptions are consistent with the straightforward interpretation of $T10S$ described above, or with the fact that the TREC filtering tasks make no explicit distinctions between categories.

4.2 Normalized Filtering Utility

Given that outside of its variable properties in precision-recall space, $T10S$ is otherwise a good metric with an understandable user model, one wonders if it might be possible to correct for these problems while still preserving it's understandable user model and hyperbolic isocurves in precision-recall space. We can gain insight in how to do so if we consider another linear metric: unnormalized tracking cost (C_{trk}) which is used for the TDT tracking task (which is similar in many ways to the TREC filtering tasks). The value of C_{trk} is defined for category C as:

$$\begin{aligned} C_{trk}(C) &= C_{miss} * P_{on} * P_{miss} + C_{fa} * P_{off} * P_{fa} \\ &= C_{miss} * P_{on} * (\frac{N_+ - A}{N_+}) + C_{fa} * P_{off} * (\frac{B}{N - N_+}) \end{aligned} \tag{7}$$

where:

- C_{miss} and C_{fa} are the relative costs of a miss (relevant document not assigned to C) and a false-alarm respectively
- P_{miss} and P_{fa} are the conditional probabilities of a miss or a false-alarm occuring, given that the document is relevant or not relevant to C respectively
- P_{on} and P_{off} are the prior probabilities that a document is relevant or not-relevant to C. $P_{on} + P_{off} = 1$ naturally.
- A, B, and N_+ are the number of correct assignments, false-alarms, and documents relevant to C respectively.
- N is the total number of documents in the corpus.

In TDT, the values of P_{on} and P_{off} are fixed to their prior probabilities of 0.02 and 0.98 respectively for all categories in the tracking task. However, if we replace these values with their posterior probabilities (e.g. $P_{on} = N_+/N$ and $P_{off} = (N - N+)/N$, then equation 7 becomes:

$$C_{trk}(C) = C_{miss} * \frac{N_+ - A}{N} + C_{fa} * \frac{B}{N} \tag{8}$$

Written in terms of recall and precision, this form of C_{trk} is

$$C_{trk}(C) = \frac{N_+}{N}(C_{miss} * (1 - r) + C_{fa} * r(\frac{1-p}{p})), p \neq 0 \tag{9}$$

This immediately suggests that by normalizing C_{trk} by P_{on}, we can obtain a version of C_{trk}, designated C'_{trk} which is stable in precision-recall space. Written in terms of A, B, and N_+, C'_{trk} is:

$$\begin{aligned} C'_{trk}(C) &= \frac{N}{N_+}(C_{miss} * \frac{N_+ - A}{N} + C_{fa} * \frac{B}{N}) \\ &= C_{miss} + \frac{C_{fa} * B}{N_+} - \frac{C_{miss} * A}{N_+} \end{aligned} \tag{10}$$

If we subtract C'_{trk} from C_{miss} (which is equivalent to flipping the scale and moving the zero point), scale by $1/C_{miss}$ so that the upper bound becomes 1, and rename C_{miss} to C_{corr}, we get the following normalized linear utility metric, which we call *normalized filtering utility* and designate U_f:

$$\begin{aligned} U_f(C) &= \frac{C_{corr} * A - C_{fa} * B}{C_{corr} * N_+} \\ &= \frac{C_{corr} * P_{on} * P_{corr} - C_{fa} * P_{off} * P_{fa}}{C_{corr} * P_{on}} \\ &= r(1 - (\frac{C_{fa}}{C_{corr}})(\frac{1-p}{p})) \end{aligned} \tag{11}$$

U_f is essentially an unbounded $T10S$. We derrive it in this fashion to emphasize both it's connection to the C_{trk} metric used in TDT and its theoretical justification in terms of being a weighted combination of the conditional probabilities of correctly and incorrectly identifying relevant documents. Unlike $T10S$, U_f has consistent isocurves in precision-recall space, and thus it's straighforward interpretation as measuring the trade-off between effort expended and information gained is consistent with what it actually measures.

As an unbounded metric, U_f suffers from the weakness that poor-performing categories can dominate the macro-average. We address this by limiting the limiting the lower value of U_f to $U_{f,min}$. Like $minU$ for $T10S$, $U_{f,min}$ represents the lowest return on reading the set of documents assigned a category the user is willing to accept before he or she regards that set as worthless, but since $U_{f,min}$ is applied after normalization, the tolerance of the user for poor performance by the filtering system remains consistent from category to category. We can now scale U_f to fall between 0 and 1 by:

$$U'_f = \frac{max(U_f, U_{f,min}) - U_{f,min}}{1 - U_{f,min}} \quad (12)$$

where 1 represents maximum information gain with minimum effort, and 0 represents the point where the documents become worthless.

4.3 Comparison of Metrics

As an example, figure 6 plots F_β and $T10S$ vs. precision across all runs submitted to the adaptive and batch filtering subtasks for category R15. As we expect from our analysis in section 4.1, F_β and $T10S$ are correlated when precision is greater than or equal to 1/3, since the isocurves of $T10S$ in this region have a similar shape to the isocurves for F_β and thus a strategy that maximizes $T10s$ is also likely to maximize F_β and vice-versa, but are uncorrelated when precision is less than 1/3. Note that because most runs have a precision above 1/3 for most categories, the macro-average F_β and $T10S$ for each run will appear to track each other, even though the metrics are not necessarily correlated.

	Batch		Adaptive	
Metric	Validation	Test	Validation	Test
$T10S$	0.681	0.324	0.387	0.263
$F\beta$	0.703	0.511	0.343	0.499
U'_f	0.671	0.548	0.362	0.463

Table 5. Performance of our systems on for $T10S$, F_β and U'_f metrics

Table 5 shows the performance of our batch and adaptive filtering systems on both the validation and test sets for all three metrics. For U'_f, we set $C_{corr} = 2$, $C_{fa} = 1$ and $U_f, min = -0.5$, which corresponds to $T10S$ with an α of -1.0. Note that F_β and U'_f have much more stable performance when going from validation to evaluation conditions, than $T10S$ for which performance decreases by more than half for the batch filtering tasks. In the adaptive filtering task, the performance drop experienced by the increase of N_+ in going from validation to test data hides an important observation: that the margin-based algorithm actually performs signifcantly *better* (significantly improved recall) on the test data than on the validation data! This again illustrates the effect of the variation of the isocurves of $T10S$ with number of relevant documents for a category; a system tuned to an optimal region on the validation data may find itself in a very suboptimal region when evaluated on the test data and the isocurves of $T10S$ shift with the change in category frequency, even though its relative performance on both validation and test data remains approximately that same.

Note also that F_β and U'_f have similar values for both batch and adaptive filtering and validation and test conditions. This is to be expected, since for most categories, we are operating in the region ($p > 1/3$) where F_β and U'_f have similar isocurves.

5. Conclusions

In our TREC-10 experiments and analysis, we observed the following:

- Standard Rocchio using relevance feedback to update the profiles but not the threshold performed surprisingly well: ranking fourth of thirty runs for both the F_β and $T10S$ metrics.
- Rocchio using relevance feedback and margin-based local regression (our new approach to adaptive thresholding) significantly outperformed the baseline Rocchio using relevance feedback and constant thresholds.
- The isocurves of the $T10S$ metric vary their locations in precision-recall space with the number of documents relevant to a particular category, causing this metric to favor common categories over rare ones and potentially obscuring important observations. We propose a slight but important modification to $T10S$ which removes these undesirable properties.

For future research, we would like to consider the following open questions:

- Why Rocchio produced more separable scores than kNN remains an open question. More failure analysis with methods other than kNN and Rocchio would be helpful in understanding the nature of adaptive filtering.
- Are all of the current classifiers used for adaptive filtering only finding those relevant documents which surround the initial two positive examples for each category? How can a classifier obtain relevance feedback for positive examples in clusters other than the initial one?
- How can we measure redundant information and return the set of documents which best covers what the user needs to know? What sorts of metrics are best suited for measuring this task?
- Why did feature selection fail to produce any improvement for our batch filtering results, when it has produced considerable improvement in other text categorization tasks on other corpora?

References

[1] Thomas Ault and Yiming Yang. knn at trec-9. In E. M. Voorhees and D.K. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. Department of Commerce, National Institute of Standards and Technology, 2000.

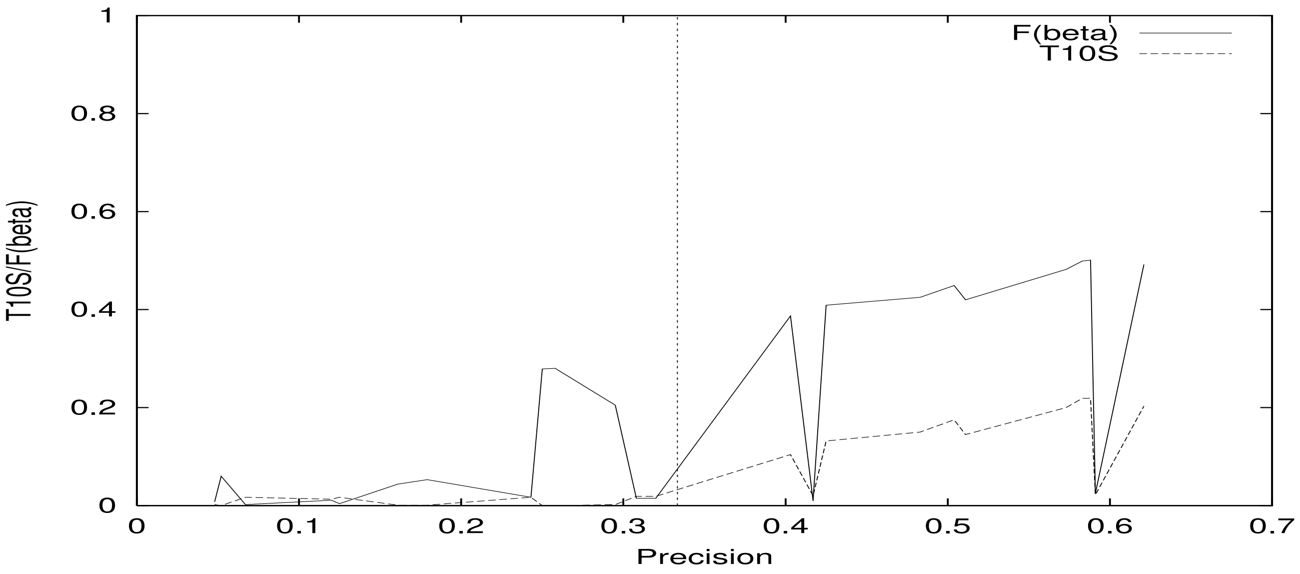

Figure 6. $T10S$ and F_β vs. precision for category R15

[2] Belur V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* McGraw-Hill Computer Science Series. IEEE Computer Society Press, Las Alamitos, California, 1991.

[3] J. J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton Ed., editor, *The SMART Retrieval System: Experiments in Automatic Document Retrieval*, pages 313–323, Englewood Cliffs, New Jersay, 1971. Prentice-Hall, Inc.

[4] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[5] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Sciences*, 41:288–297, 1990.

[6] Robert .E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In *21th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 215–223, 1998.

[7] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[8] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 13–22, 1994.

[9] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

[10] Y. Yang and X. Liu. A re-examination of text categorization methods. In *The 22th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, 1999.

[11] Y. Yang and J.P. Pedersen. A comparative study on feature selection in text categorization. In Jr. D. H. Fisher, editor, *The Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[12] Yiming Yang. A study on thresholding strategies for text categorization. In *The Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New York, 2001. The Association for Computing Machinery.

[13] Yiming Yang, Thomas Ault, and Thomas Pierce. Improving text categorization methods for event tracking. In *The 23th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, page (submitted), 2000.

[14] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2001)*, New York, 2001. The Association for Computing Machinery.