kNN at TREC-9

Tom Ault and Yiming Yang {TOMAULT,YIMING}@CS.CMU.EDU
Language Technologies Institute and Computer Science Department
Newell Simon Hall 3612C, Carnegie Mellon University
Pittsburgh, PA 15213-8213, USA

Abstract

We applied a multi-class k-nearest-neighbor based text classification algorithm to the adaptive and batch filtering problems in the TREC-9 filtering track. While our systems performed well in the batch filtering tasks, they did not perform as well in the adaptive filtering tasks, in part because we did not have an adequate mechanism for taking advantage of the relevance feedback information provided by the filtering tasks. Since TREC-9, we have made considerable improvements in our batch filtering results and discovered some serious problems with both the T9P and T9U metrics. In this paper, we discuss these issues and their impact on our filtering results.

1. Introduction

We participated in the TREC-9 information filtering track, submitting results for the OHSU and full MeSH topic sets for the batch and adaptive filtering tasks. We used a filtering engine based on the multi-class kNN algorithm successfully applied to other text categorization problems reported in the literature[6, 5]. In adapting kNN to the TREC-9 filtering tasks, we faced the following challenges:

1. Find the optimal per-category decision thresholds, given that these thresholds vary with the quantity and quality of the training data.
2. Take advantage of the relevance feedback information provided by the TREC filtering tasks.

In our official submissions, we were somewhat successful in meeting the first challenge, and not at all successful in meeting the second. As a result, we did well in the batch filtering tasks, but not as well in the adaptive filtering ones.

Since TREC-9, we have accomplished the following:

1. Increased the performance of our system for the batch filtering tasks by improving our threshold calibration methods and exploring alternative scoring mechanisms.
2. Discovered problems with the T9U and T9P metrics used for official evaluation in TREC-9.
3. Developed an effective mechanism for taking advantage of relevance feedback information.

We discuss the first two accomplishments in this paper. Although our new relevance feedback mechanism is promising, we have just begun to experiment with it, and so discussion of it is deferred to a future work.

This paper has five sections past the introduction. Section 2 describes our filtering system, including the multi-class kNN classifier and our threshold calibration mechanisms; section 3 summarizes our official TREC-9 submissions, while section 4 discusses improvements to our batch filtering results, and in section 5, we analyze the problems inherent in the T9P and T9U metrics and suggest an alternative metric for future evaluations. Section 6 presents our conclusions and future research goals for information filtering.

2. System Description

2.1 Multi-class kNN

We used the multi-class kNN algorithm previously applied by Yang et. al. to the OHSUMED[5] and Reuters collections[6] for our document filtering experiments. We chose this version of the algorithm over the single-class algorithm used in our TDT work because of the large number of categories in the MeSH topic set. Unlike the single-class variants, the multi-class kNN algorithm effectively considers all categories "simultaneously" and is much more efficient for large topic sets.

Documents are represented using the conventional vector space model in which each element is a weighted term corresponding to a token (word) appearing in the T (title), W (abstract), A (author), and S (source, e.g. journal) sections of the document. A document is parsed into a vector of term-weights by breaking the content of the T, W, A and S sections into tokens[1], eliminating stop

[1] A token is the longest occuring sequence of alphanumeric

words taken from a conventional list, stemming with the Porter stemmer, and computing term weights using a variation of the Okapi term-weighting formula[3, 1]:

$$w(t,\vec{d}) = \frac{tf(t,\vec{d})}{0.5 + 1.5 * \frac{len(\vec{d})}{avg\_len} + tf(t,\vec{d})} \times \frac{log(0.5 + N - n(t))}{0.5 + n(t)} \quad (1)$$

where

$w(t,\vec{d})$ is the weight of term t in document $\vec{d}$;

$tf(t,\vec{d})$ is the within-document frequency of term t;

N is the number of documents in the training set;

$n(t)$ is the number of training documents in which t occurs;

$len(\vec{d})$ is the number of tokens in document $\vec{d}$ after stemming and stop-word removal, e.g. $\sum_{t \in \vec{d}} tf(t,\vec{d})$

avg_len is the average number of tokens per document in the training set, e.g $\frac{1}{N}\sum_{i=1}^{N} len(\vec{d}_i)$

The values of N, $n(t)$, and avg_len were computed from the entire training set, but were not updated as the test set was processed, because dynamic updating of training set parameters slows down our current, soon-to-be-improved, document indexing system.

The basic multi-class kNN algorithm has three steps:

1. Index the training set

2. For each document $\vec{x}$ to be classified, retrieve its k most-similar documents from the training set (where k is a parameter of the algorithm). Call this set $\mathbb{R}_k(\vec{x})$.

3. For each category C, compute its relevance to $\vec{x}$ as:

$$s(C,\vec{x}) = \sum_{\vec{d} \in \mathbb{R}_k(\vec{x},C)} sim(\vec{d},\vec{x}) \quad (2)$$

where $\mathbb{R}_k(\vec{x},C)$ is the subset of documents in $\mathbb{R}_k(\vec{x})$ that are relevant to C.

We use the standard cosine-similarity metric to compute similarity between the training and test documents (e.g. $sim(\vec{d},\vec{x}) = cos(\vec{d},\vec{x}) = \frac{\vec{d}\cdot\vec{x}}{\|\vec{d}\|\|\vec{x}\|}$). If we let

N_{train} be the number of documents in the training set

N_c be the number of categories being evaluated over

$|v|$ be the size of the training set vocabulary

$\bar{v}$ be the average number of words per document

$\bar{c}$ be the average number of categories per document

then step (1) takes $O(N_{train}\bar{v})$ time and space, step (2) takes $O(\frac{N_{train}\bar{v}^2}{|v|})$ time and no additional space, and step (3) takes $O(N_{train}logk) + O(k\bar{c})$ time and $O(k + N_c)$ space. Note that these complexities do not include the time it takes to convert a document to its vector space representation.

There are many ways to transform the scores $s(C,\vec{x})$ for a particular category-document pair into a YES/NO decision on whether to assign that document to that category. In this paper, we consider three methods which have been widely reported in the text categorization literature, which we call SCut, RCut and PCut respectively[6]:

- *SCut*: Assign to each category a threshold $t(C)$. Assign a document $\vec{x}$ to category C if $s(C,\vec{x}) \geq t(C)$. How the category-specific thresholds $t(C)$ are set for the multi-class kNN algorithm is discussed in section 2.2.

- *RCut*: For each document $\vec{x}$ in the evaluation set, sort its scores $s(C,\vec{x})$ in descending order and assign the top R-ranking categories in this list to $\vec{x}$, where $R > 0$ is an integer parameter of the RCut scoring method.

- *PCut*: For each category C, sort the scores $s(C,\vec{x})$ for it in descending order. Assign to C the top $N_P(C)$-ranked documents $\vec{x}$ for that category, where $N_P(C) = N_{doc} \times K \times P(C)$, N_{doc} is the number of documents in the evaluation set, K is a user-specified parameter, and $P(C)$ is the estimated prior probability of category C. In this paper, we tested our system on two different ways to compute $P(C)$:
  - *Uniform*: $P(C) = 1/N_c$
  - *Training Set Relative Frequency*: $P(C) = N_{train}(C)/N_{train}$, where $N_{train}(C)$ is the number of documents in the training set assigned to category C.

  When necessary to distinguish the two variants, we call the former "Uniform PCut" and the latter "Proportional PCut."

The relative strengths and weaknesses of these three scoring methods are investigated in a separate paper[7]. Because the PCut method requires scores to be assigned to all documents in the evaluation set before decisions about category assignments are made, this method is not suitable for use in the TREC-9 filtering track, given its constraint that category assignment decisions must be made on-line. We present results for this scoring method only for comparison purposes with the other two methods, RCut and SCut, both of which can make category assignment decisions in real-time.

2.2 Parameter Calibration

Before the multi-class kNN algorithm can be used, the value of k must be set. We used standard m-way cross-validation to set this value; the OHSUMED-87 training data was split into m partitions, with documents assigned randomly to each partition. For each cross-validation run, m_{tr} of these partitions formed the training subset and $m_{va}(= m - m_{tr})$ partitions the validation subset. Partitions were rotated between the training and validation subsets so that each partition was used m_{tr} times for training and m_{va} times for validation. Performance was averaged over all m runs to produce a final value used for comparison between different values of k. In our experiments, we considered $k = 10, 50, 100, 200$ and settled on $k = 200$.

characters, dashes ("-"), or underscores ("_") followed by an optional *'s* or *'t* digraph)

Setting the values of $t(C)$ for the SCut method is a little more tricky, since these values depend on the number and diversity of examples for each category in the training set, as well as many other factors. For our TREC-9 experiments, we explored four different methods for computing $t(C)$:

1. *Through Standard Cross-Validation*
   Use the same m-way cross-validation procedure used to set k, and average the per-category optimal thresholds obtained from each of the m cross-validation runs. If the ratio of m_{tr} to m_{va} is large enough, then the training subset used for each cross-validation run should be sufficiently representative of the complete training data that the averaged thresholds will be sufficiently close to the true optimal values.

2. *Through Linear Regression with Respect to Training Set Size*
   Perform m-way cross-validation with at least three different ratios of m_{tr} to m. For each category, fit a straight line using linear regression to the (ratio, optimal threshold) pairs for that category, and use this straight line to predict the optimal threshold for the full OHSUMED-87 training set. If the thresholds for a category are not sufficiently linear, use the threshold from the largest ratio as a fallback value.

3. *Through Linear Regression with Respect to Number of Examples*
   Perform m-way cross-validation with at least three different ratios of m_{tr} to m, and record the number of examples of each category in the training subset for each cross-validation run. Fit a straight line via linear regression to the $N_{ratios} \times m$ data points for each category, and use the straight line to predict the optimal threshold for a category as a function of the number of examples of that category in the complete training set.

4. *Through a Modified Leave-One-Out Cross-Validation Algorithm*
   Perform a variation of leave-one-out cross-validation on the training data. For each document $\vec{d}$ in the training set, use every other document in the training set to assign scores $s(C, \vec{d})$ via the multi-class kNN algorithm. Then set the values of $t(C)$ to be those which produce optimal performance over this set of scores. This method has the advantage of deriving the values of $t(C)$ from a data set that is as close as possible to the actual training data.

Only thresholding methods (1) and (2) had been developed when the official submissions for the TREC-9 filtering track were due, and hence our official submissions presented in section 3 reflect the performance of these two methods. Thresholding methods (3) and (4) were developed after the TREC-9 submission deadline to improve the performance of our batch filtering systems, and their impact is discussed in section 4.

The values of R for the RCut scoring method and K for the PCut scoring method can also be set using either method (1) or method (4). Because of time constraints, we use method (4) to set these values in this paper, and defer comparison of the two methods to a later work.

3. Official TREC-9 Submissions

We submitted six runs for the TREC-9 filtering track, four *baseline* and two *combination* runs. The baseline submissions (runs CMUCAT1-4 in Table 1) represent our best attempt at tuning the parameters of our basic system, while the combination runs (CMUCAT5 and CMUCAT6 in Table 1) are an attempt to improve performance on the OHSU query set by using a weighted linear combination of the output of the multi-class kNN algorithm on two different views of the documents in the training and evaluation sets, one view in which the abstracts were left unmodified, and another view in which the abstracts had been replaced with the definitions of the MeSH subject headings appearing in the .M section of the document. All runs used $k = 200$ and the SCut scoring method, but *none* of them, including those submitted for the adaptive filtering task, made use of relevance feedback information; only the initial training data was used to filter documents.

The following summarizes our official results

- *Baseline Batch Filtering*
  Thresholds for the baseline batch filtering runs (CMUCAT1 and CMUCAT4) were set using method (2). The poor performance of the baseline run on the OHSU queries (CMUCAT4) can be explained by the lack of linearity with training set size of the thresholds for these categories. Of all 63 categories in this set, only 11 were sufficiently linear with training set size to predict a good threshold for the full training set. In contrast, 3790 (77%) of the 4904 MeSH topics were sufficiently linear with training set size to predict a good threshold. If we use method (1) with a m_{tr}:m_{va} ratio of 19:1, T9P for CMUCAT4 leaps to 0.241, which is comparable to the scores for the other runs.

- *Baseline Adaptive Filtering Runs*
  Thresholds for the adaptive filtering runs (CMUCAT2, CMUCAT3, and CMUCAT5) were set using a version method (1) modified to account for the limited available training data. This variation always keeps the adaptive filtering training documents in the training subset and rotates the remainder of the OHSUMED-87 data set through the validation subset.

- *Comparison of Baseline Runs*
  Multi-class kNN performed better on the batch filtering task than on the adaptive filtering one, which is expected since the former has more training data than the latter. The algorithm also performed much better on the MeSH topics than on the OHSU queries. This may also be because of the larger number of training documents on average per category for the MeSH topics than for the OHSU queries for both tasks (an average of 236.82 documents/topic for MeSH vs. 50.87 documents/topic for OHSU for the batch filtering task, and 4 documents/topic for MeSH vs. 2 documents/topic for OHSU for the adaptive filtering task).

- *Combination Runs* Combining scores from classifying different views of a document seems to improve performance by 1-2% in T9P over the corresponding baseline run (when appropriate thresholding is used).

4. Improved Batch Filtering Performance

4.1 Improved Threshold Calibration For SCut

Increases in the performance of our batch filtering methods come from the development of improved threshold calibration methods for the SCut method and the application of alternative transformations (RCut and PCut) from

Table 1: Official Submissions by CMU-CAT for the TREC-9 Filtering Track

		Topic	Use .M	Single or	
Run ID	Task	Set	Field?	Combination?	T9P
CMUCAT1	Batch	MeSH	No	Single	0.436
CMUCAT1*	Batch	MeSH-SMP	No	Single	0.443
CMUCAT2	Adaptive	MeSH	No	Single	0.303
CMUCAT2*	Adaptive	MeSH-SMP	No	Single	0.304
CMUCAT3	Adaptive	OHSU	No	Single	0.213
CMUCAT4	Batch	OHSU	No	Single	0.100
CMUCAT5	Adaptive	OHSU	Yes	Combination	0.224
CMUCAT6	Batch	OHSU	Yes	Combination	0.261

*The CMUCAT1 and CMUCAT2 runs were used for both the full and "sampled" MeSH topic sets; the CMU-CAT group did not have separate submissions for these two topic sets.

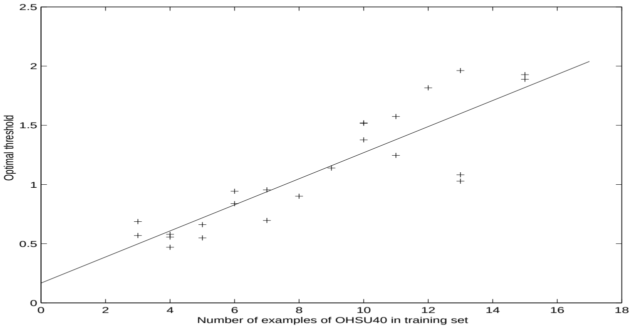

Figure 1. Optimal threshold vs. number of examples in training set for OHSU40

document-category pairs scores to assignment decisions for that category. Figure 1 shows the motivation for threshold calibration method (3). Most, but not all, categories from both the OHSU and MeSH topic sets have optimal thresholds which are linear with the number of examples of that category in the training set. We can use the (number of examples, optimal threshold) pairs gathered from different cross-validation runs to build a linear predictor of the optimal threshold for a category given the number of examples of it currently in the training set.

Threshold calibration method (4) was motivated by the observation that the larger the ratio of m_{tr} to m_{va} used for threshold calibration method (1), the better the thresholds predicted by this method. If we want to classify a document in the training set, the most representative subset of the training data we can use consists of every document in the training set except the one being classified. After scoring every document in the training set in this fashion, we can set the per-category thresholds to be those which yield optimal performance on these scores.

Sometimes this method computes very low thresholds that perform very poorly on both the training and test data. This typically occurs for categories that have few representative examples in the training set, and thus those examples are assigned low scores during the training set self-evaluation. However, since any non-zero T9P, no matter how small, is better than than a zero T9P, method (4) will set the "optimal" threshold for any such category to the score of its highest-ranked positive example, even though that threshold will recall far too many false-alarms when applied to the test data.

To compensate for this behavior, we have added fallback values to threshold calibration method (4). If the performance of the "optimal" threshold computed by method (4) for a category over its set of scores falls below a specified value, then we use the score of the N_{fb}-th ranked document for that category as its threshold instead. How N_{fb} is computed depends on the fallback method chosen. We examined three fallback methods in our post-TREC-9 filtering work: *FBR*, *FBP* and *FBPcut*.

1. *FBR*: $N_{fb}(C) = y$, where y is a constant rank specified by the user for all categories. If y exceeds the number of documents with scores for a category, the score of the lowest-ranked document is used. This method is also called "fallback to constant rank."

2. *FBP*: $N_{fb}(C) = p * N_{doc}(C)$, where $N_{doc}(C)$ is the number of documents in the training set with scores for category C and p is a proportion between 0 and 1 assigned by the user. This method is also called "fallback to proportional rank."

3. *FBPCut*: $N_{fb}(C) = K * N_{doc} * P(C)$ (e.g. the same forumula used for the PCut scoring method), where N_{doc} is the number of documents in the training set, K is a nonnegative user-specified constant, and $P(C)$ is one of the two distributions (uniform or proportional) used by the PCut method in section 2.1. This method is also called "PCut fallback."

Table 2 shows the potential gains possible from adding fallbacks to method (4). For both topic sets, FBPcut and FBR made similar gains in performance while FBP showed no improvement over the no fallback condition. Note that the parameters are the ones that produce optimal performance on the *test* data rather than the training data, and thus represent potential rather than actual gains. The parameters for FBP, FBR and FBPCut are sensitive to overfitting, and we are exploring effective ways to set them from the training data.

4.2 Alternative scoring methods RCut and PCut

Until recently, we believed that SCut was the top-performing scoring method regardless of the corpus or evaluation conditions. Recent work by Yang[7] has shown that this is not

Table 2: Comparison of Different Fallback Methods

	Topic	Optimal	
Method	Set	Parameter(s)	T9P
FBP	OHSU	p= 0.05	0.278
FBR	OHSU	y= 10	0.313
FBPCut	OHSU	P= 1.1	0.305
FBP	MeSH	p= 0.05	0.462
FBR	MeSH	y= 10	0.475
FBPCut	MeSH	P= 4,TrainRF	0.474

true. Of the three common scoring methods (PCut, RCut and SCut), which one is better varies with the corpus and the desired ability to make trade-offs between recall and precision. Consequently, we applied the RCut and PCut scoring methods to our TREC-9 batch filtering results to see if they would produce better results than SCut. The results are shown in Table 3.

4.3 Analysis

Except for RCut, all of the methods we explored in this section outperformed SCut with threshold calibration method (2), which we used for our official batch filtering submissions. RCut performed poorly because of its inability fine-tune the number of assignments made by the system, forcing it to draw too many false-alarms or not enough correct documents. PCut performed extrordinarily well, outperforming our best SCut thresholding strategy (method 4) by 7% for the OHSU queries![2] Unfortunately, as mentioned in section 2.1, PCut cannot be used to make real-time assignments.

For the OHSU queries, threshold calibration methods (3) and (4) give equal performance, even though they use very different means to compute the set of optimal thresholds. This suggests that we have found the best possible thresholds from the training data for categories with good performance, and we need to concentrate our efforts on low-performing categories. The improvements in T9P from using fallback thresholds with method (4) support this claim. On the other hand, there is slight but significant improvement between methods (3) and (4) for the MeSH categories, suggesting that further improvement in threshold optimization is possible even for well-performing categories in this topic set.

5. Problems with T9P and T9U

Systems participating in the TREC-9 filtering track were evaluated by one or both of two measures, T9P or T9U, which are defined for a category as:

$$T9P = \frac{A}{max(A+B, \alpha)} \tag{3}$$

$$T9U = max(2A - B, minU) \tag{4}$$

where

A is the number of relevant documents assigned to that category

B is the number of false-alarms for that category

α is a constant parameter indicating the desired number of documents to be retrieved for that category. Systems which retrieve less than α documents for the category are penalized by having the remaining $\alpha - (A+B)$ documents considered to be false-alarms.

$minU$ is a constant parameter representing the minimum value of the T9U measure. This is to keep large negative utility scores from dominating the system-wide average of T9U.

For TREC-9, α was fixed at 50 for all categories, and $minU$ was fixed at -100 for all categories in the OHSU query set and at -400 for all categories in the MeSH topic set. The overall value of T9P or T9U for a filtering system is the unweighted average of its T9P or T9U values for the individual categories (also known as the *macro-average* of T9P or T9U in the information retrieval literature).

Both T9P and T9U can be rewritten in terms of the number of relevant documents for the category (N_+), and the widely-known information retrieval metrics recall[3] (r) and precision[4] (p):

$$T9P = \begin{cases} p & \text{if } r >= \frac{\alpha}{N_+}p \\ \frac{N_+}{\alpha}r & \text{if } r < \frac{\alpha}{N_+}p \end{cases} \tag{5}$$

$$T9U = \begin{cases} min(N_+ r(3 - \frac{1}{p}), minU) & \text{if } p > 0 \\ \text{Some value in } [minU, 0) & \text{if } p = 0 \end{cases} \tag{6}$$

(Note that if $p = 0$, $A = 0$ and the value of B becomes unrecoverable; hence, in this case, T9U will have some negative value not directly computable from r, p and N_+)

5.1 T9P

Several problems with T9P are immediately visible from an examination of equation 5 and the isocurves of T9P plotted in figures 2 through 4 for $\alpha < N_+$, $\alpha = N_+$ and $\alpha > N_+$ respectively. Specifically:

- In spite of its name, T9P actually measures *recall* if $r < \frac{\alpha}{N_+}p$, and thus the macro-average of T9P is actually a mix of recall and precision values, depending

[2] A program bug discovered at the last minute prevented us from evaluating the MeSH topic set using the PCut method.

[3] Recall is defined for a category as the ratio of relevant documents assigned to the category to the number of relevant documents for that category, e.g. A/N_+.

[4] Precision is defined for a category as the ratio of relevant documents assigned to the category to the total number of documents assigned to that category, e.g. $A/(A+B)$

Table 3: Improved Batch Filtering Results for TREC-9

	Offical	Method (3)	Method (4)		PCut	
Topic Set	T9P	T9P	T9P	RCut	Priors	T9P
OHSU	0.100	0.277	0.278	0.013	Training	0.348
MeSH	0.436	0.441	0.463	0.292	Training	*

*An unfortunate program bug that kept us from evaluating the MeSH topics was discovered the night the paper was due.

on where the operating point of each category lies in recall-precision space!

- The isocurves of T9P impose a harsh tradeoff between recall and precision. When the operating point lies above the line $r = \frac{\alpha}{N_+}p$, only improvements in precision will be of any benefit. When below this line, only improvements in recall will have any effect.

- From figure 2, if $\alpha < N_+$, then T9P imposes an effective maximum recall of $\frac{N_+}{\alpha}$. Increasing recall past this point will be of no benefit to the system.

- Likewise, from figure 4, if $\alpha > N_+$, then T9P imposes an effective maximum precision of $\frac{N_+}{\alpha}$. Furthermore, this value is also the maximum value of T9P for the category, and so the macro-average T9P of a perfect filtering system over a set that includes categories with this property will not be 1.0, but some value possibly much less. For example, the full MeSH topic set has a maximum macro-average T9P of 0.99, but the OHSU query set has a maximum of 0.73! This has strong implications for comparisons of macro-average T9P across topic sets.

- The properties of T9P are heavily dependent on the number of relevant documents for a category and thus are not consistent from category to category. This makes the macro-average of T9P an especially confusing and counter-intuitive metric.

Most of the problems with T9P come from the harsh, discontinuous trade-off between recall and precision that occurs at the line $r = \frac{\alpha}{N_+}p$. A metric with a smoother trade-off would not have these problems. While the other metric used for TREC-9, T9U does have a smooth trade-off between recall and precision, it has other problems, which we discuss in the next section.

5.2 T9U

Figure 5 shows the isocurves for the T9U metric. Almost immediately, one can see that T9U is a better metric than T9P because its isocurves have a smooth continuous trade-off between recall and precision. Nor does T9U become insenstive to changes in recall or precision except in two regions:

- When p= 1/3, T9U becomes insensitive to changes in recall because for every correct document assigned, two false-alarms are also assigned.

- In the region on or above the curve $r = (\frac{minU}{N_+})(\frac{p}{3p-1})$ (if $minU < 0$) or in the region on or below that curve (if $minU > 0$), $T9U = minU$ regardless of the value of recall or precision.

However, T9U is not without its problems, most of which are discussed in the final report for the TREC-8 filtering track[2] and summarized here:

- The minimum value of T9U ($minU$) is arbitrary value reflecting a particular user's tolerance of poor-performance by the system.

- The maximum value of T9U depends on N_+, the number of relevant documents for a topic, which presents a problem for macro-averaging across topics, since topics with many relevant documents will dominate the average. While T9U could be scaled to fall within the same range for all topics, such scaling has problems of its own, which are discussed in [2].

- In the region where T9U is negative ($p < 1/3$), an increase in recall actually results in a *decrease* in T9U, making it possible for one system to have a higher T9U than a second system which has higher recall and precision.

- A system which recalls *no* relevant documents (or any documents at all) may have a higher T9U than a system that recalls a few correct documents and many false-alarms. However, if we assume that the user ignores any category for which too many false-alarms are returned, then a system which returns a few relevant documents among many irrelevant documents, a system which returns no relevant documents, and a system which returns no documents at all are equally "useful" to the user, and T9U should reflect this with an equal value for for all three systems.

While one might attempt to avoid the problems with T9U by setting $minU = 0$ and thus avoiding the entire region of negative utility where most of the problems occur, this would have the undesirable side-effect of obscuring large performance differences between systems. What is needed is a metric which has the same smooth trade-off between recall and precision in its isocurves that T9U has, but without the undesirable behavior exhibited by T9U when it becomes negative. We examine such a metric in the next section.

5.3 A Proposed Alternative To T9P and T9U

Figure 6 shows the isocurves for van Rijsbergen's F_β-measure [4]

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r}$$

with $\beta = 1$. Like T9U, F_β's isocurves have a smooth, continuous trade-off between recall and precision. However, F_β has several nice properties not found in T9P or T9U, specifically:

- F_β exhibits its smooth, continuous trade off throughout the entire recall-precision space, never becoming completely insensitive to changes in recall or precision.

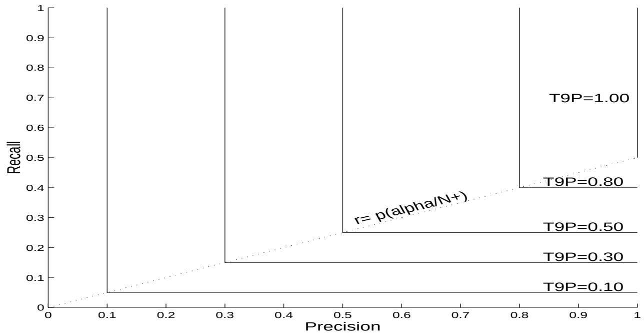

Figure 2. Isocurves of T9P for $alpha/N_+ < 1$

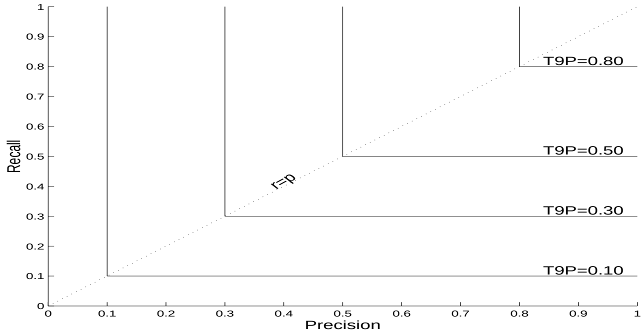

Figure 3. Isocurves of T9P for $alpha/N_+ = 1$

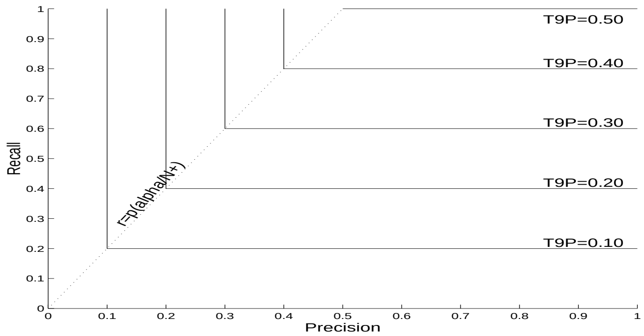

Figure 4. Isocurves of T9P for $alpha/N_+ > 1$

- By adjusting the value of β, one can adjust the recall-precision trade-off of the isocurves to favor recall ($\beta > 1$) or precision ($\beta < 1$). Moreover, this trade- off will be the same for all categories, regardless of the number of relevant documents or other category-dependent properties.
- Expanding on the above item, the range and interpretation of F_β have no category-dependent properties, and thus there are no problems in interpreting its macro-average value.
- In contrast to T9U, systems which return no documents and systems which return only irrelevant documents both have the minimum F_β of zero.
- F_β with $\beta = 1$ is a commonly used metric in information retrieval research, and thus the use of the F_β measure would make the results of the TREC filtering tracks more comparable to other published results in the information retrieval literature.

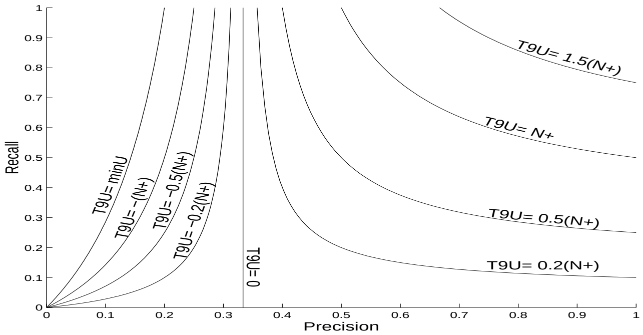

Figure 5. Isocurves of T9U

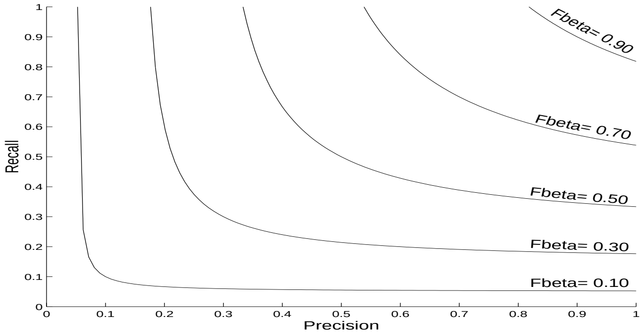

Figure 6. Isocurves of F_β with $\beta = 1$

6. Conclusions and Future Work

We have made considerable progress towards applying the multi-class kNN algorithm to the TREC batch and adaptive filtering tasks. We have developed improved thresholding methods and a promising relevance feedback mechanism. We have also discovered serious problems with the T9P and T9U metrics, and proposed the use of the F_{beta} metric instead. Our efforts in the coming year will focus on exploring the properties of our new relevance feedback mechanism and discovering new ways to tune paramters for our fallback thresholds.

References

[1] M. Franz and S. Rouikos. Trec-6 ad-hoc retrieval. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, 1994.

[2] David A. Hull and Stephen Robertson. The trec-8 filtering track final report. In D.K. Harmon, editor, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 1999.

[3] S. E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In D.K. Harmon, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, 1994.

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

[5] Y. Yang. An evaluation of statistical approach to text categorization. In *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*, 1997.

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

[7] 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, (submitted). The Association for Computing Machinery.