Improving text categorization methods for event tracking

Yiming Yang and Tom Ault and Thomas Pierce and Charles W. Lattimer
Language Technologies Institute and Computer Science Department
Newell Simon Hall 3612D, Carnegie Mellon University
Pittsburgh, PA 15213-8213, USA; www.cs.cmu.edu/~yiming/

Abstract

Automated tracking of events from chronologically ordered document streams is a new challenge for statistical text classification. Existing learning techniques must be adapted or improved in order to effectively handle difficult situations where the number of positive training instances per event is extremely small, the majority of training documents are unlabelled, and most of the events have a short duration in time. We adapted several supervised text categorization methods, specifically several new variants of the k-Nearest Neighbor (kNN) algorithm and a Rocchio approach, to track events. All of these methods showed significant improvement (up to 71% reduction in weighted error rates) over the performance of the original kNN algorithm on TDT benchmark collections, making kNN among the top-performing systems in the recent TDT3 official evaluation. Furthermore, by combining these methods, we significantly reduced the variance in performance of our event tracking system over different data collections, suggesting a robust solution for parameter optimization.

Keywords:

- event detection and tracking
- machine learning and IR
- statistical/probabilistic models
- text categorization
- evaluation

1 Introduction

Topic Detection and Tracking (TDT) is a new line of research composed of three major sub-problems: segmenting speech-recognized TV/radio broadcasts into news stores, detecting novel events in segmented or unsegmented news streams, and tracking the development of an event based upon human-identified sample stories of that event[1, 24, 22, 3]. This last task, event tracking, is the focus of our research in this paper. The problem is defined as automatically assigning pre-defined event labels to documents presented to the system in a chronological order. An *event* in the TDT context is *something that occurs at specific place and time associated with some specific actions.* It contrasts with a *topic* in the traditional text categorization sense in that events are localized in space and time. For example, the *EgyptAir-990 crash* is an event, but not a topic, and "airplane accidents" is a topic but not an event. Systems in the TDT domain must be able to distinguish between events, regardless of whether they are part of the same topic or not. Events are typically short in duration and thus only a small portion of any corpus will be about any particular event.

Event tracking can be considered a text categorization problem subject to the following constraints:

- Each event of interest is defined by a set of positive instances (documents) that are manually identified before tracking starts; no other knowledge is available.
- As soon as a new document arrives, a binary (YES/NO) decision must made by the tracking system with respect to each defined event.
- Any document preceding the document being evaluated may be used as training data. However, only the previously-identified positive instances are labelled; the rest of the documents are not, although some of these unlabelled documents may actually be positive instances.
- When training on an event, relevance judgements for other events are assumed to be unknown. In actual use, the user of a tracking system would only be willing to identify a small number of documents relevant to the event of interest and none for events he or she is not interested in, nor could users be expected to share their relevance judgements.

From a statistical learning point-of-view, it is more difficult to identify documents as instances of small, *living* events than of large, *stable* topics. For example, the prior probabilities of subject categories in Reuters newswire stories are relatively stable and can be accurately estimated using a retrospective collection of

news stories. Estimating prior probabilities of recent or future events is more difficult, since most of the new events have no instances in retrospective data collections and events can evolve in unpredictable ways over time. Moreover, since the majority of training documents are unlabeled, learning an exact decision surface is much more difficult for event tracking than for the traditional text categorization problems when all the training documents are labelled with a complete category set. Classification methods which rely on accurate prior probability estimates or large amounts of labeled training data will perform poorly in event tracking and many existing text categorization methods would need to be modified to perform well under TDT conditions.

The small size of an event also presents special difficulties in tuning the parameters of the tracking system to produce optimal results. In text categorization, a standard approach to learning the optimal parameters is to use one or more *validation sets* of documents which are not used for training or evaluation but believed to be sufficiently representative of the unseen test documents. In event tracking, however, the small number of labelled positive examples available for training, and the online decision required per test documents upon it arrival make it difficult to construct a validation set that is representative of the unseen documents. For example, the typical number of positive training examples available in TDT benchmark evaluations are one to four, but the size of an event in the evaluation set is much larger (about 350 documents/event on average in the TDT3 corpus). Holding off any of these positive training examples for validation would reduce the effectiveness of training, and the small number of positive examples held for validation may not offer sufficient statistics for parameter tuning. Furthermore, the early-time positive training examples of an event may not be representative of later positive examples of that event.

An alternative approach to parameter optimization would be to tune parameters on a retrospective corpus in which a larger number of positive examples have been identified for some events. This works as long as the optimal parameter settings for older events are optimal for newer ones also, or if the events being evaluated are large and long-lasting so that they have a sufficient number of positive examples in the retrospective validation corpus. Empirical results in the TDT evaluations (by our group and other research sites) have shown that often this is not the case; optimal parameter settings for early and later documents are often very different, and it is overly optimistic to expect optimal parameter settings to generalize to new events.

These and other open challenges in event tracking have been partly addressed in the recent TDT research and the benchmark evaluations[1, 6]. An increasing number of information retrieval and machine learning techniques have been applied, including k-Nearest Neighbor (kNN) classification, Decision Tree induction, a variety of Language Modeling (LM) approaches and relevance-based filtering methods[2, 24, 22, 18, 15, 16], and systematic analysis of their behavior on the event tracking task have just begun. In this paper, we report our new research findings with kNN and Rocchio in event tracking. Our KNN methods were among the two top-performing tracking systems in the official TDT1 and TDT3 evaluations, respectively. Rocchio approaches have not been thoroughly investigated in TDT so far, although several relevance-based filtering systems applied to TDT are more or less similar to Rocchio[2, 15].

2 Methods

2.1 Document Representation

Each document is represented using a vector of weighted terms which can be either words or phrases. We use a common version of the TF-IDF scheme[13] for term weighting [1], defined to be

$$w(t,\vec{d}) = \frac{(1+\log_2 tf(t,\vec{d})) \times \log_2(N/n_t)}{\|\vec{d}\|}$$

where

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

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

$\log_2(N/n_t)$ is the Inverted Document Frequency (IDF);

N is the number of documents in the training set;

n_t is the number of training documents in which t occurs;

$\|\vec{d}\| = \sqrt{\sum_{t\in\vec{d}} w(t,\vec{d})^2}$ is the 2-norm of vector $\vec{d}$.

2.2 Rocchio

Rocchio is a classic retrieval method for query expansion using relevance judgments on documents[12, 13]. The Rocchio formula is defined to be

$$\vec{q'}(\vec{q}, D, \alpha, \beta, \gamma) = \alpha\vec{q} + \beta\frac{1}{|R|}\sum_{\vec{y}\in R}\vec{y} + \gamma\frac{1}{|S|}\sum_{\vec{z}\in S}\vec{z}.$$

where $\vec{q}$ is the original query, $\vec{q'}$ is the extended query, D is a training set, $R \in D$ consists of the training documents relevant to the query, $S = D - R$ consists of the training documents irrelevant to the query, and α, β and γ are the weights for the three components in the summation. Rocchio has been applied to text categorization (TC) in a modified form:

$$\vec{c}(D,\gamma) = \frac{1}{|R|}\sum_{\vec{y}\in R}\vec{y} + \gamma\frac{1}{|S_n|}\sum_{\vec{z}\in S_n}\vec{z} \qquad (1)$$

where $\vec{c}(D,\gamma)$ is the *prototype* or the *centroid* of a category and is Rocchio's representation of the event. D and R are the same as defined before; $S_n \in D - R$ consists of the n most-similar (as measured by cosine similarity) negative instances to the positive centroid (e.g. $\sum_{\vec{y}\in R}\vec{y}$). This formula allows us to selectively use the negative examples that lie in the neighborhood (called

[1] TF-IDF based term weighting has been intensively studied in the IR literature. The implementation of the standard versions (more than a dozen) are provided in the SMART benchmark retrieval system (developed at Cornell). We tested a few common options and found the *ltc* option yielded the best results in our limited experiments. This does not mean that *ltc* is the best possible term weighting scheme for document collections. Finding the best term weighting scheme is an open research question.

the "query zone" in the literature[14]) of the positive centroid. Using the category prototype, a test document can be scored by computing the cosine similarity

$$r(\vec{x}, \vec{c}(D, \gamma)) = cos(\vec{x}, \vec{c}(D, \gamma)).$$

A binary decision is obtained by thresholding on this score.

One should note that classification in Rocchio does not depend on estimation of the prior probabilities of categories and that its one-centroid-per-category assumption is not necessarily a deficiency in event tracking, where news stories collected over a short time period are often similar in content and topic, and hence the small number of positive examples available for training are likely to form a tight cluster. For these reasons, we believe Rocchio may perform well in event tracking[2] even though it is one of the weaker methods for text categorization in cross-method comparisons with kNN, Support Vector Machines, boosting methods, Sleeping Experts, inductive rule learning and Naive Bayes classifiers [7, 14, 4, 10].

2.3 New Variants of kNN

kNN is an instance-based classification method commonly used in pattern recognition and machine learning. We have found kNN a robust approach to text categorization, ranking among the top-performing classifiers in cross-method evaluations on benchmark collections [20, 21, 23]; other methods with comparable performance include the Decision Trees with boosting by Weiss et al., Support Vector Machines by Joachims, the Boosting method by Schapire et al., the Linear Least Squares Fit mapping by Yang and the neural networks by Wiener et al. [21, 23, 14, 19].

We adapted conventional M-way classification kNN to the 2-way classification problem of event tracking by constructing a 2-way kNN for each event. The kNN computes a relevance score for each test document by the formula we call kNN.sum:

$$r(\vec{x}, k, D) = \sum_{\vec{y} \in P_k} cos(\vec{x}, \vec{y}) - \sum_{\vec{z} \in Q_k} cos(\vec{x}, \vec{z}) \qquad (2)$$

where

- $\vec{x}$ is the test document;
- $\vec{y}$ ($\vec{z}$) is a positive (negative) training document;
- D is the training set of documents;
- k is the number of nearest neighbors ("local zone") of $\vec{x}$ in D, which the system use to compute the score;
- P_k (Q_k) is the set of the positive (negative) instances among the k nearest neighbors of $\vec{x}$ in D.

A binary decision (YES or NO) on $\vec{x}$ with respect to the event is obtained by thresholding on kNN.sum. Like Rocchio, kNN does not depend upon an estimate of the prior probability of an event. Unlike Rocchio, kNN allows multiple centroids per category, and uses information from the neighborhood local to each test document.

The performance of kNN depends on the choice of the value of k. In text categorization, empirical evaluations have shown that a stable or optimal performance of kNN typically occur with using a relatively large value of k (between 30 to 200, for example), and that using a retrospective validation set to choose the optimal or nearly optimal value of k on test documents is not difficult[20, 21]. Optimizing k in event tracking, however, is a problem if we use the kNN.sum formula because the very small number of positive examples in the training set. Using a large value of k will retrieve many negative examples, whose sum could easily exceed the sum of the positive examples, even if each negative example is very dissimilar to the test document and thus has a small similarity score. Using a small value of k will cause the system to retrieve only negative examples for a test document unless it is very close to a positive training example. This will causes the system to lose discriminatory power for most test documents.

To resolve this dilemma, we modified the scoring function in Formula 2 using two alternatives we call kNN.avg1 and kNN.avg2 respectively:

$$r'(\vec{x}, k, D) = \frac{1}{|P_k|} \sum_{\vec{y} \in P_k} cos(\vec{x}, \vec{y}) - \frac{1}{|Q_k|} \sum_{\vec{z} \in Q_k} cos(\vec{x}, \vec{z}) \qquad (3)$$

$$r''(\vec{x}, kp, kn, D) = \frac{1}{|U_{kp}|} \sum_{\vec{y} \in U_{kp}} cos(\vec{x}, \vec{y}) - \frac{1}{|V_{kn}|} \sum_{\vec{z} \in V_{kn}} cos(\vec{x}, \vec{z}) \qquad (4)$$

where U_{kp} consists of the kp nearest neighbors of $\vec{x}$ among the positive documents in the training set; and V_{kn} consists of the kn nearest neighbors of $\vec{x}$ among the negative documents in the training set.

kNN.avg1 is almost the same as kNN.sum except using the score averages instead of the score sums. This modification allows k to be set to any large number without allowing negative examples to dominate. kNN.avg2 differs from the other two versions by using two local zones (kp positive nearest neighbors and and kn negative nearest neighbors) instead of one (k nearest neighbors), allowing the sizes of the zones to be as small as needed while guaranteeing that the system uses both positive and negative instances to score a test document. Thus the system will not lose any discriminatory power when using small local zones. Our empirical results (Section 4) show that both of the new variants of kNN can significantly improve the tracking performance of the original kNN.

2.4 Parameter Tuning

As mentioned in the introduction, parameter tuning is a tricky issue for event tracking. The kNN methods need the values of k, kp and kn to be optimized, Rocchio needs the values of γ and n to be optimized, and all methods need the threshold for binary decisions to be tuned for optimal results. To address the unavailability of adequate validation sets for the TDT tracking

[2] Rocchio may not perform well when dealing with large, long-lasting and evolving events. However, we cannot empirically verify this without a corpus with a sufficient number of manually-identified large, long-lasting events. To our knowledge, no such corpus is available.

task, we propose a novel solution: combining the output of a diverse set of classifiers and tuning parameters for the combined system on a retrospective corpus in which a larger number of positive examples are identified for some old events. The idea comes from the well-known strategy of investing in a variety of funds in the stock market to offset the volatility of any one of them, yielding better returns on average over the long run, and from then well-known practice in information retrieval and speech recognition of combining the output of a large number of systems to yield a better result on average[17, 8, 9, 5]. In both cases, entities with large individual performance variances are averaged together to produce a single entity whose variance is much reduced. We leverage this idea in a novel application: effective parameter tuning for event tracking. By combining the output of multiple classifiers whose errors tend to be uncorrelated, we hypothesize that the cross-collection performance variance of the combined system will be much less than those of the individual classifiers. In other words, the combined system is likely to yield better results on a new test collection after optimization on a given validation set than the individual systems would do after being optimized on the same validation set.

Ideally, we would like to test our hypothesis with at least a few dozen systems. However, such a large number of systems is not currently available, so we limit our empirical validation to the two new variants of kNN and the Rocchio method, with more careful analysis on the behavior of these methods and the conditions we combine them. We designed the following experiments to verify the effectiveness of our approach:

1. Have two sets of data in disjoint time windows, namely, Corpus A and Corpus B. Each corpus contains a set of events with labelled positive documents for each of those events; the two sets of events do not overlap. Use Corpus A for parameter tuning and Corpus B for evaluation, and vice versa.

2. For each event in each of the corpora, split the corpus right after the N_t labelled positive examples; use the first half of the corpus for training, and the second half for testing.

3. Run kNN.avg1, kNN.avg2 and Rocchio on Corpus A with different parameter settings. Select the runs with the optimal DET curves, globally or locally, including those optimal for low false alarm (high precision), low miss (high recall), and/or for balanced errors (optimal C_{trk}). Allow more than one run to be selected for a classifier, if needed.

4. Normalize the scores for each of the selected runs in step 3 by $x' = \frac{x-\mu}{s.d}$, where x is the original score, μ is the mean of the scores for the run, and $s.d.$ is their standard deviation. Sum together the normalized scores for each run and renormalize the result in the same way, using the resulting scores as the output of the combined system, which we have named BORG.AA (Best Overall Results Generator tuned on Corpus A and evaluated on Corpus A). Identify the threshold that yields the minimum C_{trk} for the BORG.AA results.

5. For each selected run from step 3, create a parallel run on Corpus B using the same parameters. Combine the parallel runs to create parallel BORG results (called BORG.AB) for Corpus B.

6. Evaluate each parallel run on Corpus B using the optimal parameters from Corpus A and compare the results of BORG.AB with the parallel runs of the individual classifiers.

7. Swap the role of Corpus A and Corpus B, and repeat Steps 3 to 6.

Results of the above experiments are describe in Section 4.

3 Data and Evaluation Measures

3.1 Corpora

We chose the TDT1 corpus of news stories covering July 1, 1994 to June 30, 1995 and the TDT3 dry-run corpus covering January 1, 1998 to June 30, 1998 as our temporally-disjoint corpora. We use both corpora as they are and set the evaluation conditions as close as possible to those used in the TDT1 and TDT3 benchmark evaluations to make our results comparable to the published results on these evaluations.

The TDT1 corpus, developed by the researchers in the TDT Pilot Research Project, was the first benchmark evaluation corpus for TDT research.[3] It consists of 15,863 chronologically-ordered news stories; roughly half of these stories are randomly sampled Reuters articles, and the other half are CNN broadcasts which were manually transcribed by the Journal Graphics Institute (JGI). Twenty five events were manually identified from the memories of the participants for the period covered by this corpus or by scanning through the collection, and then exhaustive relevance judgements were made for each of those events for all stories in the corpus. There are about 43 documents per event on average. Each story was assigned a label of YES (article focusses on event), NO (article does not mention event) or BRIEF[4] (article mentions event in passing) for each of the 25 events. Note that this process resulted in only a subset of the existing events in the corpus; however, for each of those labelled events, complete relevance judgments over all documents are available. For each event and a particular N_t value, the TDT1 corpus was split at the point right after the N_t-th positive example of that event; the stories before that split point were allowed to be used for training, and the remaining stories were used for testing. Fifteen of the 25 events have more than 16 YES stories, the maximum allowable N_t in TDT1, and thus were the ones used for event tracking evaluation. In this paper, we fix N_t at 4 to make the setting consistent with the TDT3 evaluation.

The TDT3 dry-run corpus, developed at LDC, is a larger and richer collection, consisting of 82,084 documents with 100 manually identified events. Each event consists of 350 documents on average and comprises about 0.4% of the corpus. Not all the events have exhaustive relevance judgments over the entire corpus.

[3] TDT data collections are made available via the Linguistic Data Consortium (LDC) – see www.ldc.upenn.edu/TDT

[4] In the official TDT evaluations, the stories judged as BRIEF were allowed to be used for training but were excluded from testing. We did the same in our evaluations to make our results comparable with the results by others in the TDT benchmark evaluations.

The documents were collected from 3 newswires, 3 radio programs and 4 television programs; some of these documents are in both English and Mandarin, with a machine-translated (via SYSTRAN at the LDC) version of the Mandarin-language documents also available.[5] The audio sources (TV or radio) were transcribed either manually or by automatic speech recognition. Twenty out of the 100 events were selected for the TDT3 dry-run evaluation on the basis that each event was bi-lingually parallel in the corpus and had more than 4 positive instances in both English and Mandarin. Each tracking system could use up to 4 positive training instances per event as training data for TDT3 dry runs. In the evaluation, the corpus was split immediately after the fourth positive example in the English sources or the fourth positive example in the Mandarin sources, whichever occurred later. The stories before that split point were used for training, and those after for testing, implying that, at the splitting point, there may be more than 4 positive training examples in one of the two languages but only 4 are explicitly labelled.

3.2 Measures

To evaluate the effectiveness, a two-by-two contingency table is used for each event:

Table 1. Contingency table

	YES is true	NO is true
System-predicted YES	a	b
System-predicted NO	c	d

Performance measures are defined as:

- Miss $m = \frac{c}{a+c}$ if $a + c > 0$, otherwise undefined;
- False Alarm $f = \frac{b}{b+d}$ if $b + d > 0$, otherwise undefined;
- Recall $r = \frac{a}{a+c}$ if $a + c > 0$, otherwise undefined;
- Precision $p = \frac{a}{a+b}$ if $a+b > 0$, otherwise undefined;
- Tracking cost $C_{trk} = \alpha_1 \frac{b}{n} + \alpha_2 \frac{c}{n}$ where $n = a + b + c + d$.

Global performance over all events is evaluated using two methods: the *micro-average*, obtained by first summing the corresponding cells in the contingency tables of the individual events and computing the global performance scores from the combined table, and the *macro-average*, obtained by computing per-event performance measures first and averaging them. The micro-average introduces a scoring bias towards frequently-reported events, and the macro-average towards less-reported events; both are informative. Macro-averaged C_{trk} have been used as the primary measure (with $\alpha_1 = 0.1$ and $\alpha_2 = 1$) in benchmark TDT evaluations. We therefore present our results using this measure and refer it as C_{trk} unless otherwise specified.

In addition to the above measures defined for classification decisions, we also use the Detection-Error Trade-off (DET) curve[11] to show how varying the threshold would affect the trade-off between the miss and false-alarm rates. The DET curve is obtained by sweeping the decision threshold over the range of scores generated by the tracking system, evaluating its performance at small, consecutive intervals, and plotting the normal deviates of the miss versus false alarm rate at each interval.

[5]Including Mandarin data is not important for our study; however, we used this corpus as it is to make our results comparable to other TDT benchmark evaluation results.

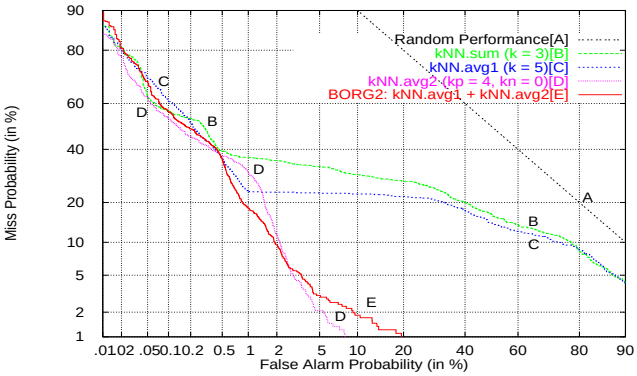

Figure 1: kNN variants on TDT1

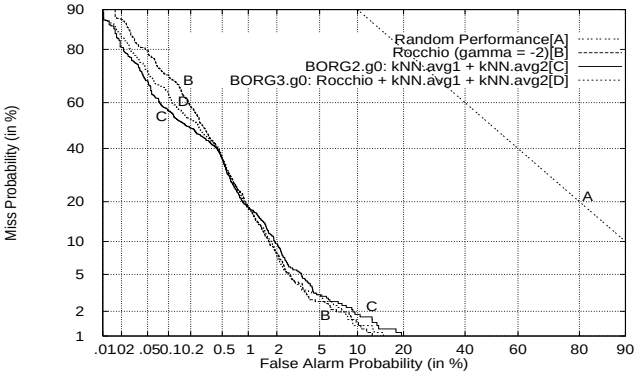

Figure 2: BORG combining Rocchio and 2 kNN variants

4 Results

4.1 Primary results

We tested Rocchio and the variants of kNN on TDT1 and TDT3, with various parameter settings for γ and n in Rocchio, and for k, kp or kn in the kNN variants, yielding a DET curve for each method and parameter setting. Figure 1 show the "best" DET curve for each method on TDT1, where "best" means that minimum C_{trk} of that method lies on the curve. A reference line for a system with random performance is also added as a poor baseline. Because the DET curve plots normal deviates of the miss and false alarm rates, rather than the rates themselves, random performance generates a straight line passing through the 50%-50% error point with slope -1.0.

The DET curves of kNN.avg1 and kNN.avg2 are "complementary": one is better in the low false alarm (high precision) area and the low miss (high recall) area, while the other is better for more balanced errors (high C_{trk}); both curves are significantly better than kNN.sum (optimized at $k = 3$). BORG2 combining these two new variants of kNN has an even better curve in the

balanced-error region (center of the graph), but not in the high-precision or high-recall ends (recall larger than 90% or precision higher than 99.5%). Adding Rocchio to this combination, shown in Figure 2, only yields a marginal improvement. We interpret this to mean that Rocchio (using one centroid for a large number of the negative examples) and kNN.avg2 with $kn = 0$ (ignoring negative training examples entirely in the kNN scoring) tend to make similar classification decisions, emphasizing the influence of positive training examples while discounting (or ignoring) the negative training examples. Combining such highly correlated methods is not expected to produce very different results from those by the individual systems.

The optimal C_{trk} scores, obtained by thresholding on the DET curves of these systems, are shown in Tables 2 and 3. When tuning these systems on TDT1 and evaluating them on the same corpus (the "AA" column in the table), Rocchio with $\gamma = -2$ and $n = 200$ had the lowest score among the individual methods. The new variants of kNN had worse scores than Rocchio, but significantly better than the original kNN.sum did. KNN.avg1 reduced the C_{trk} score from .0056 of kNN.sum to .0033 (a 41% reduction), while kNN.avg2 reduced the score to .0030 (a 46% reduction). When tuning these systems on TDT3 and evaluating them on the same corpus, the improvement by the new kNN variants was event larger, i.e., from .0080 to .0023 (a 71% reduction). BORG3 combining Rocchio and the two new kNN variants yielded an identical or somewhat worse scores than the best scores found for the individual methods on these two corpora.

The results of tuning the systems on the TDT1 corpus and evaluating them on the TDT3 corpus are shown in the "AB" column in Table 3. Conversely, the results of tuning the systems on the TDT3 corpus and evaluating them on the TDT1 corpus are shown in the "BA" column in Table 2. The threshold for optimal decision-making in the evaluation runs was selected by one of two strategies:

- use the exact threshold optimal for the system on the corpus for tuning; or
- select the threshold that generates the same proportion of YES decisions in both the tuning corpus and the evaluation corpus.

The first strategy ("T1") allows a online decision for each document upon it arrival, while the second strategy ("T2") sometimes yields better results. Both are provided in these tables for a comparison.

4.2 Performance variance reduction

While the BORG combination of kNN and Rocchio had performance scores as good or slightly worse than the individual methods, this is not important, since our point is not to use BORG as a fine tuning technique for marginal improvement on a fixed corpus (i.e., the validation corpus). Our goal is to improve the performance consistency of our tracking system between different collections and on new events for which few tracking examples are available. We hypothesize that we can achieve this goal by combining classifiers with different learning strategies and have shown the BORG combination of Rocchio and the kNN variants as a convincing example. We expect that multiple learning strategies will fail more gracefully on new data or events than a single-strategy classifier optimized on a validation corpus because multiple learning strategies usually will not produce the same kinds of errors simultaneously. As illustrated in Tables 2 and 3, the best-performing single method on TDT1 was Rocchio; however, it performed poorly on TDT3 data with its TDT1-optimal parameters. BORG tuned on TDT1, on the other hand, had a much more consistent performance of TDT3, yielding a 58% reduction from C_{trk} of Rocchio when applying the optimal threshold of BORG on the TDT1 corpus to its scores on the TDT3 documents ("AB.T1" in Table 3). Figures 3-6 illustrate the DET curves for each method evaluated on TDT3 with TDT1-optimal and TDT3-optimal parameters. Clearly, BORG has a much smaller performance variance over any region in the error trade-off space compared to the individual methods. This explains why optimizing the decision threshold through cross-validation is easier for BORG than for the individual methods. These observations strongly support our assertions in Sections 1 and 2.4 about the difficulty of parameter tuning across different collections and events, and likewise supports our rationale for combining a diversity of classifiers to reduce performance variance.

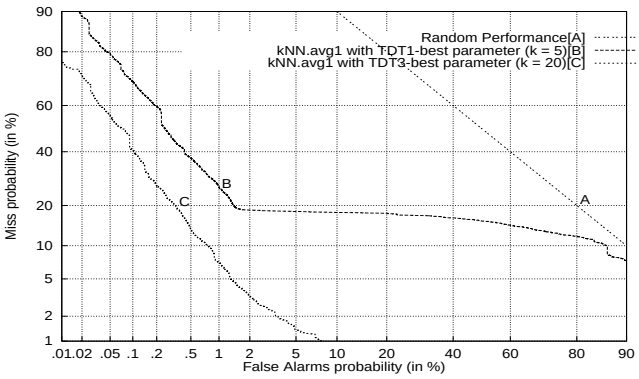

Figure 3: Variance of kNN.avg1 between optimal parameters for TDT1 and TDT3 on the TDT3 dry-run2 corpus

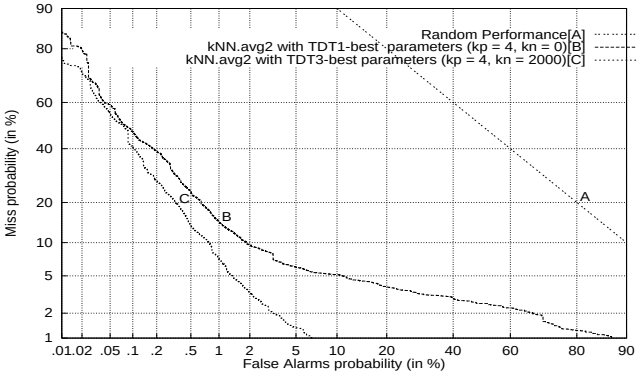

Figure 4: Variance of kNN.avg2 between optimal parameters for TDT1 and TDT3 on the TDt3 dry-run2 corpus

Table 2. Results summary in C_{trk} of classifiers evaluated on TDT1 Corpus

System	AA	δ	BA.T1	δ	BA.T2	δ
kNN.sum ($k = 3$)	.0056	+50%	.0080	+64%	.0081	+63%
kNN.avg1 ($k = 5$)	.0033	+15%	.0030	+3%	.0027	-11%
kNN.avg2 ($kp = 4, kn = 0$)	.0030	+7%	.0032	+9%	.0028	-7%
Rocchio ($\gamma = -2, n = 200$)	.0022	-36%	.0033	+12%	.0028	-7%
BORG3 (Rocchio + kNN.avg1 + kNN.avg2)	.0028	+0%	.0029	+0%	.0030	+0%

Table 3. Results summary in C_{trk} of classifiers evaluated on TDT3 Corpus

System	BB	δ	AB.T1	δ	AB.T2	δ
kNN.sum ($k = 3$)	.0080	+71%	.0085	+68%	.0116	+78%
kNN.avg1 ($k = 2000$)	.0023	+0%	.0063	+57%	.0075	+67%
kNN.avg2 ($kp = 4, kn = 2000$)	.0023	+0%	.0076	+67%	.0090	+72%
Rocchio ($\gamma = -.25, n = 200$)	.0026	+12%	.0060	+58%	.0047	+47%
BORG3 (Rocchio + kNN.avg1 + kNN.avg2)	.0023	+0%	.0025	+0%	.0025	+0%

AA: tuned on TDT1 and tested on TDT1; BA: tuned on TDT3 and tested on TDT1;
AB: tuned on TDT1 and tested on TDT3; BB: tuned on TDT3 and tested on TDT3;
δ: C_{trk} reduction by using BORG instead of the individual classifier.

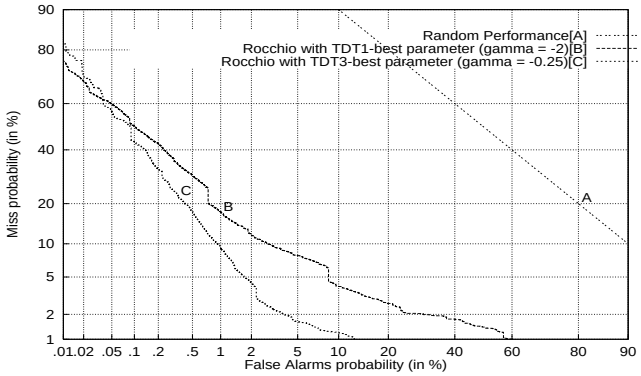

Figure 5: Variance of Rocchio between optimal parameters for TDT1 and TDT3 on the TDT3 dry-run2 corpus

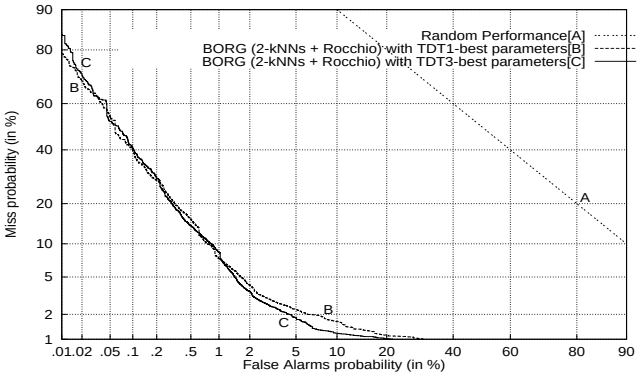

Figure 6: Variance of BORG3 between optimal parameters for TDT1 and TDT3 on the TDT3 dry-run2 corpus

5 Summary and Future Research

This paper studied the effectiveness of kNN and Rocchio in event tracking. We addressed the open questions of training classifiers on extremely small number of positive examples and cross-event parameter learning using a validation corpus. We developed new variants of kNN to overcome the specific problems of the original version when applied to event tracking; the new versions reduced the average cost (C_{trk}, the sum of weighted error rates) by 71% on the TDT3-dryrun corpus, mading kNN among the two top-ranking systems in the recent TDT3 official evaluation (December 1999).

We also found a robust solution for cross-event parameter learning: combining the output of classifiers with diverse learning strategies. Our cross-corpus evaluations on the TDT1 and TDT3 benchmark collections showed up to a 58% reduction in C_{trk} by combining kNN variants and Rocchio instead of using the best single method (kNN or Rocchio) on the validation corpus, strongly supporting the effectiveness of our approach. We believe this is also a powerful solution for problems beyond event tracking, such as categorization of documents (web pages) on the World Wide Web where the category spaces are typically fast growing and changing, or effective training for extremely rate categories in conventional text categorization problems. We would like to study this approach with a large number of diverse classifiers, including support vector machines, neural networks. etc.

References

[1] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, San Francisco, CA, 1998. Morgan Kaufmann Publishers, Inc.

[2] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45, New York, 1998. The Association for Computing Machinery.

[3] Jaime Carbonell, Yiming Yang, John Lafferty, Ralf D.Brown, Tom Pierce, and Xin Liu. Cmu report

on tdt-2: Segmentation, detection and tracking. In *Proceedings of the DARPA Broadcast News Workshop*, pages 117–120, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[4] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 1996. The Association for Computing Machinery. 307-315.

[5] Jon Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *IEEE Workshop on Automatic Speech Recognition and Understanding*, Piscataway, NJ, 1997. IEEE Signal Processing Society.

[6] Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. Nist's 1998 topic detection and tracking evaluation (tdt2). In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 19–26, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[7] W. Lam and C.Y. Ho. Using a generalized instance set for automatic text categorization. In *Proceedings of the 21th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 81–89, 1998.

[8] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–297, New York, 1998. The Association for Computing Machinery.

[9] Joon Ho Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, New York, 1995. The Association for Computing Machinery.

[10] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 1996. The Association for Computing Machinery. 298-306.

[11] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The det curve in assessment of detection task performance. In *EuroSpeech 1997 Proceedings*, volume 4, 1997.

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

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

[14] Robert .E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–223, New York, 1998. The Association for Computing Machinery.

[15] J. Michael Schultz and Mark Liberman. Topic detection and tracking using idf-weighted cosine coefficient. In *Proceedings of the DARPA Broadcast News Workshop*, pages 189–192, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[16] R. Schwartz, T. Imai, L. Nguyen, and J. Makhoul. A maximum likelihood model for topic classification of broadcast news. In *Proceedings of Eurospeech*, Rhodes, Greece, 1997.

[17] Joeseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *The Second Text REtrieval Conference*, pages 243–252, 1994.

[18] F. Walls, H. Jin, S. Sista, and R. Schwartz. Topic detection in broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, pages 193–198, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[19] S.M. Weiss, C. Apte, F. Damerau, D.E. Johnson, F.J. Oles, T. Goets, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[20] 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.

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

[22] Y. Yang, J.G. Carbonell, R. Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[23] Y. Yang and X. Liu. A re-examination of text categorization methods. In *The Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, New York, 1999. Association for Computing Machinery.

[24] Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In *Proceedings of the 21th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 28–36, 1998.