On-line New Event Detection and Tracking

James Allan, Ron Papka, and Victor Lavrenko

Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003

Abstract We define and describe the related problems of *new event detection* and *event tracking* within a stream of broadcast news stories. We focus on a strict on-line setting—i.e., the system must make decisions about one story before looking at any subsequent stories. Our approach to detection uses a single pass clustering algorithm and a novel thresholding model that incorporates the properties of events as a major component. Our approach to tracking is similar to typical information filtering methods. We discuss the value of "surprising" features that have unusual occurrence characteristics, and briefly explore on-line adaptive filtering to handle evolving events in the news.

New event detection and event tracking are part of the Topic Detection and Tracking (TDT) initiative.

1 Introduction

The problems discussed in this study are *new event detection* and *event tracking*. The goal of those tasks is to monitor a stream of broadcast news stories so as to determine the relationships between the stories based on the real-world events that they describe. New event detection requires identifying those news stories that discuss an event that has not already been reported in earlier stories. Event tracking means starting from a few sample stories and finding all subsequent stories that discuss the same event.

Typical Information Retrieval problems rely upon a user-defined query to specify what is "interesting." A retrieval system finds documents that help the user satisfy an information request. A filtering system uses a long-lived profile of a user's request to identify relevant material in a stream of arriving documents. In contrast, new event detection has no knowledge of what events will happen in the news, so must operate without a prespecified query. A detection algorithm might look for clues in the news reporting and/or maintain a "memory" of past events so that it can determine that a new event is being mentioned. The event tracking task does include an implicit query in the sample stories, but the "query" is given by example and is meant to capture the underlying event, slightly different from the typical IR concern with "aboutness."

In Section 2 we discuss the TDT initiative, its basic ideas, and some related work. Section 3 discusses the corpus and evaluation measures that are used in this study. Section 4 covers our work on new event detection and is followed by Section 5 which discusses our tracking approaches and results. We conclude by briefly discussing directions of this research in Section 6.

We stress that this work was carried out as part of a pilot study with a relatively small corpus (by the standards of Information Retrieval). The results should therefore be taken as suggestive and not conclusive.

2 History and Definition

The Event Detection and Tracking problems are part of a broader initiative called Topic Detection and Tracking (TDT). The domain of TDT's interest is all broadcast news—i.e., written and spoken news stories in multiple languages. As such, the problem is substantially broader than the work reported in this study, encompassing automatic speech-to-text efforts, finding the boundaries between news stories for archival and presentation purposes, locating new events within the stream, tracking located events, and doing all of that in a multi-lingual environment with degraded information. As its name implies, TDT is also ultimately concerned with ways of organizing information that are broader than "events," but the initial work has focused on the more limited setting.

The TDT tasks and evaluation approaches were developed by a joint effort between DARPA, the University of Massachusetts, Carnegie Mellon, and Dragon Systems. A year-long pilot study was undertaken to define the problem clearly, develop a test bed for research, and evaluate the ability of current technologies to address the problem. The groups involved in the tasks found that the "state of the art" is capable of providing adequate performance for detection and tracking of events, but that there is a high enough failure rate to warrant significant research into how algorithms can be advanced. As the research is broadened to the larger TDT scope, the unresolved questions become more troublesome. Detailed results of that study are reported elsewhere[3]; this paper presents advances in our understanding of the problem after the end of the pilot study.

We wish to make it clear that the corpus and evaluation methodology that were devised in the TDT study were a joint effort by four groups. The research results reported in this study are our own; the framework for the work is only partly ours.

Permission to make digital/hard copy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or fee. SIGIR'98, Melbourne, Australia © 1998 ACM 1-58113-015-5 8/98 $5.00.

2.1 What is an event?

A possible definition of *event* is something that happens at a particular time and place. A goal of the TDT pilot study was to test that definition for reasonableness. The specific location and time of an event differentiate it from broader classes of events: for example, "The Eruption of Mt. Pinatubo on June 15th, 1991" is an event whereas "volcanic eruptions" is the more general class of events containing it. There are problems with this definition, however: many would consider "the O.J. Simpson saga" an event, though it occurs over several years and in many places. One hope of the TDT initiative is that an iterative process of partial definition, evaluation, better definition, evaluation, and so on, will eventually result in a satisfactory understanding of "event."

Though it is hard to define *event*, it is easier to define parts of *event identity*, the properties that make two events the same. A system that can represent properties of event identity is trivially capable of performing new event detection: the system determines for the current story whether it contains an event identical to one occurring in an already seen story. If so, the system does not detect a new event; otherwise it does. These properties are clearly important for an event detection and tracking system to model, so part of the TDT problem becomes deciding what properties of news stories can be used to detect event identity.

2.2 Previous work

The tasks defined within TDT appear to be new within the research community. Some efforts have been made to classify news stories or other documents into broad subject areas automatically using nearest neighbor matching [16], pattern matching[9], or other algorithms based on supervised training[4, 13, 17]. For the most part, those techniques are intended to match stories or documents against a set of category labels that are known *a priori*, a method that is helpful for event tracking but inappropriate for detection. Event detection (and to some extent tracking) requires finding stories that discuss an event that may not match any already known class of events.

One study that is close in spirit to the TDT work was done by DeJong using frame-based objects called "sketchy scripts"[8]. Frames associated with 50 general events were constructed by hand. The goal of his system was to predict which frame needed to be populated, and then to produce a short summary of the event. DeJong's system was primarily a natural language parser that detected when a story contained an event. It chose the correct script with a classification accuracy of 38% for the stories for which it had a sketchy script. We believe this frame-based approach will be difficult to use outside of specific domains: the number of frames and the details of their contents would quickly become difficult to maintain as new types of events emerge and existing events evolve in a news environment. We believe a better approach to new event detection is to use general word co-occurrence retrieval in a process that specifically models event-level features in addition to event-class-level features.

Some recent work that associates news photographs with stories about the picture[6] is very similar in spirit to Event Tracking. Their stated interest is in linking stories that discuss the same event, though they work solely on the problem of linking photographs and stories. As we will do in our work, they represented stories and photo captions by sets of features. Proper names turned out to be useful for their linking task.

The other research organizations involved in the TDT pilot study also worked on new event detection and event tracking. They used different approaches to address each of the issues discussed below, although there are some common themes that are the natural results of a collaborative study. For example, the notion of a time penalty (Section 4.2) and the use of forms of document clustering (Section 4) were threads picked up by all sites. The details of early work by Dragon Systems and Carnegie Mellon is published in the pilot study's final report.[3] Carnegie Mellon has also published additional work on detection, including the "retrospective" detection task that allows story clustering considering the entire corpus.[24]

3 Evaluation corpus

An important task of the TDT pilot study was the creation of an appropriate test corpus and a useful approach to evaluation of the problem. The goals of creating the corpus and evaluation methodology were two-fold: (1) to make strides toward a solid definition of "event" as outlined in Section 2.1, and (2) to evaluate how well "state of the art" approaches could address the TDT tasks.

To simplify the problem slightly for the pilot study, we generally ignored issues of degraded text coming from speech recordings, and used written newswire sources and human-transcribed stories from broadcast news. The resulting TDT corpus includes 15,863 news stories spanning July 1, 1994, through June 30, 1995. Half of the stories are randomly chosen Reuters news articles from that period; the other half are transcripts of several CNN broadcast news shows during the same period. The stories are assigned an ordering that represents the order that they appeared in the news. The average story contains 460 (210 unique) single-word features after stemming and removing stopwords.

The corpus also includes relevance judgments for a set of 25 events. Some events (e.g., the Oklahoma City bombing or the earthquake in Kobe, Japan) were disasters or crimes that occurred in the news and were unexpected. Others are stories that build up to an anticipated event (e.g., the collision of a comet into Jupiter, the appointment of U. S. Supreme Court Justice Breyer). The events were chosen to represent a range of events that seemed "interesting," to ensure that there would be a fair number of stories on each event in the corpus, and also to cover a range of event classes that are generally recognized as "events."

To provide a high-quality evaluation setting, *every* story in the corpus was judged with respect to *every* event. The judgments were made by two sets of assessors and any conflicts were reconciled by a third. For each of the 25 events, each of the stories was assigned a judgment on a ternary scale: about the event, not about the event, or mentioning the event but only briefly in a story that is generally not about the event. The exhaustive judgments of this corpus are in contrast to more common pooled strategies.[22] An unfortunate side-effect of requiring exhaustive judgments is that the cost of creating them limits the size of the corpus.

In all, 1132 stories were judged relevant, 250 stories were judged to contain brief mentions, and 10 stories overlapped between the set of relevant stories and the set of brief mentions. Overlaps and brief mentions were removed from the corpus before processing, leaving 1124

relevant stories for evaluation.

The TDT corpus and relevance judgments are described in more detail in the pilot study's final report[3] and are available from the Linguistic Data Consortium. The LDC is currently creating a second and larger TDT corpus (TDT2) that includes a broader range of sources, at least four times the number of stories, a larger number of judged events, as well as the audio stream and closed captioning for all broadcast sources.

3.1 Evaluation measures

Information Retrieval systems are generally evaluated on the basis of ranked recall and precision,[10, 18] though numerous other measures have been proposed.[20] Information Filtering systems are evaluated on a wider range of measures, including set-based measures and various utility measures, though no particular measure has settled out as preferred.[14]

In the TDT setting, we have chosen to measure a system's effectiveness primarily by the miss (false negative) and false alarm (false positive or fallout) rates. The major reason for choosing these is a perception of the problem as being a *detection* task rather than a ranking task: a system needs to indicate whether or not a story is new or is on the event being tracked, not provide a ranked list of stories that might discuss the event. Unfortunately, although it is fairly straightforward for systems to generate ranked lists of stories and to provide a score that creates that ranking, it is more difficult to determine a good score that can be used as a threshold. An ideal system might output a score that corresponds to the probability that the story discusses the event; ideal systems do not yet exist.

In this work, we skirt the threshold issue by using a Detection Error Tradeoff curve[15] to show how false alarm and miss rates vary with respect to each other at various threshold values. Figure 4 presents examples of DET plots showing curves for several different runs. The curves are plotted such that if the errors exhibit a normal distribution, they will result in a straight line. A perfect system would have zero misses and zero false alarms, and would have a "curve" at the origin; curves closer to the origin are generally better, though there may be applications that require good performance at particular false alarm or miss rates. For most applications, the left-hand side of the DET curve (low false alarm rate) is probably the most interesting. For the event tracking task, a false alarm rate of 1% means that as many as 158 stories per event would have been incorrectly tracked.

The DET graph is analogous to a recall/precision graph, except that "good" is in the opposite direction. Both graphs provide a means for comparing system performance across a wide range of error tradeoffs. Both allow a user to understand the tradeoff between improving one measure and worsening the other.

4 New Event Detection

New event detection operates in a strict on-line setting, processing stories from a news stream one at a time as they arrive. Our approach to the problem is a modification of the well-known single pass clustering algorithm[21]. Our algorithm processes each new story on the stream sequentially, as follows:

1. Use feature extraction and selection techniques to build a query representation for the story's content.
2. Determine the query's initial threshold by evaluating the new story with the query.
3. Compare the new story against earlier queries in memory.
4. If the story does not trigger any previous query by exceeding its threshold, flag the story as containing a new event.
5. If the story triggers an existing query, flag the story as not containing a new event.
6. (Optional) Add the story to the agglomeration list of queries it triggered.
7. (Optional) Rebuild existing queries using the story.
8. Add new query to memory.

We represent the content of each story (which we assume discusses some event) as a query. If a new story triggers an existing query, the story is assumed to discuss the event represented in the query, otherwise it contains a new event.

4.1 Detection Experiments

The new event detection algorithm was implemented by combining the ranked-retrieval mechanisms of Inquery, a feature extraction and selection process based on relevance feedback[2], and InRoute's routing architecture[5].

For comparing document d to query q we used the evaluation function of Inquery's #WSUM operator:

$$eval(q,d) = \frac{\sum_{i=1}^{N} w_i \cdot d_i}{\sum_{i=1}^{N} w_i} \qquad (1)$$

where w_i is the relative weight of a query feature q_i, and d_i is the *belief* that the feature's appearance in the document indicates relevance to the query.

The document representation used in the system is a set of belief values corresponding to each feature specified in a query. Belief values are produced by Inquery's belief function, which is composed of a *term frequency* component, *tf*, and an *inverse document* frequency component, *idf*. For any instance of document d and collection c:

$$d_i = belief(q_i, d, c) = 0.4 + 0.6 * tf * idf \qquad (2)$$

where $tf = t/(t + 0.5 + 1.5 * \frac{dl}{avg\_dl})$, $idf = \frac{log(\frac{|c|+.5}{df})}{log(|c|+1)}$, t is the number of times feature q_i occurs in the document, *df* is the number of documents in which the feature appears in the collection, *dl* is the document's length, *avg_dl* is the average document length in the collection, and $|c|$ is the number of documents in the collection.

In our system, c is an auxiliary collection, independent of the stream. Since future feature occurrences are unknown in the strict on-line case, the number of times a feature appears in the collection cannot be determined. Therefore, one could estimate *idf* using retrospective statistics from the current stream or from an auxiliary corpus with a similar domain. The *idf* component in this detection work comes from a collection built of TREC volumes 1-3 and the TREC-4 routing volume.

A feature selection process was used to build and rebuild queries. In the experiments that follow, a query was built using the n most frequent single word features found in the stories. The query feature weight was the average value using the *tf* component calculation described above.

4.2 Detection Thresholding

One of our hypotheses regarding new event detection is that exploiting time will lead to improved detection. A side-effect of broadcast news is that stories closer together on the stream are more likely to discuss related events than stories further apart on the stream. When a significant new event occurs there are usually several stories per day pertaining to it; over time, coverage of old events is displaced by more recent events.

One place to incorporate time is in the thresholding model. Our thresholding technique uses an initial threshold for each query based on the evaluation of the query against the story from which it was created using Equations 1 and 2 above. The final threshold θ for a query is calculated as a constant percentage p of the initial threshold from the default evaluation value of 0.4 used by Inquery.

A second factor of the model is a time penalty tp, that increases the threshold for a query based on the amount of time between a query and a story. If the jth story is compared to the query resulting from the ith story, for $i < j$ we have:

$$\theta(q_i, d_j) = 0.4 + p * (eval(q_i, d_i) - 0.4) + tp * (j - i)$$

We used this model with different values for p to determine a *similarity threshold* indicating the lowest evaluation score that could trigger a query, as well as a *consolidation threshold* that determined whether a story was included when rebuilding an existing query.

Our general approach to new event detection is to build successive event classifiers from the contents of the stories from the stream. The classifiers in our implementation are queries and their thresholds.

4.3 Detection Evaluation Methodology

The standard evaluation measures in TDT are *miss* and *false alarm* rates. For the detection task, a miss occurs when the system fails to detect a new event, and a false alarms occur when the system indicates a story contains a new event when it does not.

Since only 25 events in the corpus were judged, an evaluation methodology developed for the TDT study was used to expand the number of trials. The methodology uses 11 passes through the stream. The goal of the first pass is to detect a new event in the 25 stories in which one of the 25 known events first occurs. The second pass excluded these stories, and the goal was to detect the second story for each of the 25 known events: the second story artificially becomes the first story in the stream. The process iterates to skip up to 10 stories for each event. If an event contained fewer stories than the number of stories to be skipped in the pass, the event was excluded from evaluation in that pass.

Separate training and testing phases were not performed due to the unavailability of more judged events. In order to avoid over-fitting our threshold parameters p and tp, we selected parameters based on k-fold cross-validation [11]. The general algorithm is to randomly partition the data into k sets, and to leave one set out while finding parameters that best fit the remaining $k-1$ sets. The process repeats for k iterations. The parameters that give rise to the smallest overall performance error are used. Because the TDT data contains only 25 instances, we chose $k = 25$, effectively performing *leave-one-out cross-validation.*

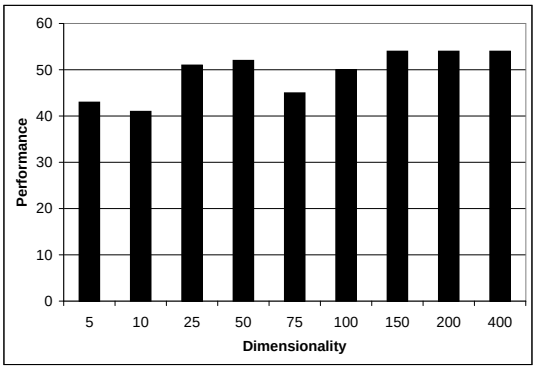

Figure 1: (Performance = 100 – *Distance from Origin*) vs. maximum number of query features.

skip	# of Docs	Miss Rate	F/A Rate	Recall	Prec	F1
0	1124	36%	1.46%	64%	50%	0.56
1	1099	36%	1.40%	64%	52%	0.57
2	1074	39%	1.24%	61%	52%	0.56
3	1051	48%	1.56%	52%	43%	0.47
4	1028	36%	1.49%	64%	48%	0.55
5	1006	45%	1.63%	55%	43%	0.48
6	984	41%	1.66%	59%	45%	0.51
7	962	40%	1.59%	60%	44%	0.51
8	942	53%	1.41%	47%	41%	0.44
9	923	63%	1.33%	37%	37%	0.37
10	904	78%	1.35%	22%	25%	0.24
Avg	1008	46%	1.46%	54%	45%	0.49

Table 1: New Event Detection with $n = 400$ features.

Once the threshold parameters are obtained, we infer their expected performance using a simple bootstrap process [7]. The process randomly selects 25 instances (with replacement) from the data to form a bootstrap sample. Performance is calculated on the sample. The process repeats for many iterations, effectively producing a distribution of performance based on the threshold parameters obtained from the cross-validation procedure.

4.4 Results

The results for the new event detection system using queries containing between 5 and 400 single-word features are listed in Figure 1. Performance in this graph is based on the Euclidean distance average miss rate and false alarm rate are from the origin (recall that, in general, points closer to the origin are "better"). In general, detection performance increases by using more single-word features in the event representation; however, the gains afforded by greater dimensionality (more single-word features) were offset by much slower running times in our system. The best parameters found across dimensionality were similar, and identical for more than 75 features. The parameters of $p = 0.225$ and $tp = 0.000008$ were determined by leave-one-out cross-validation, and yielded the best performance for high dimensionality.

Performance at 400 features represents processing at full dimensionality, in that each query contains almost

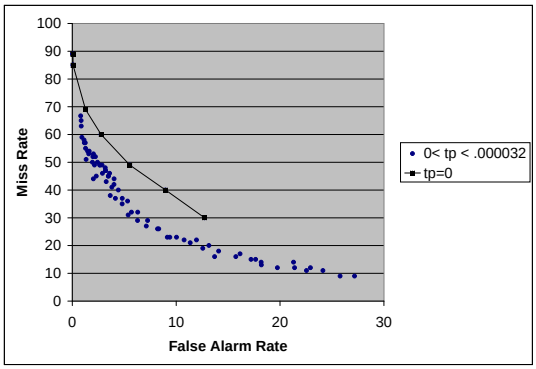

Figure 2: Effects of varying threshold parameters p and tp. (Axes have different scales.)

all the single-word features available in its corresponding story. Table 1 lists the results at 400 single-word features across the 11 passes through the corpus as described above. In these experiments, a skip value of n implies that stories 1 through n of each event were removed from the stream, and the goal was to detect the story number $(1 + n)$ for each event. A skip value of 1 implies that the second story was the goal, and so on. Averages are based on pooling all system responses across the 25 events. Performance is stable for the first few skip values, but becomes worse at higher values because fewer events are included in the pass.

The effects of the time penalty in the threshold model are illustrated in Figure 2. Each point represents average performance at a particular combination of p and tp from our parameter optimization process. The points in the graph that are connected by a line represent performance for various values of p using no time penalty (i.e., $tp = 0$). The data suggest that better overall performance is realized by using time penalties. On average, for any value of p, performance is better when $tp > 0$.

We ran the bootstrap process for 10,000 iterations to yield samples having a mean miss rate of 40.5% with a standard deviation of ±7.6%, and a mean false alarm rate of 7.8% with a standard deviation of ±4.0% . Initial experiments on the first month of data from the TDT2 corpus yielded a miss rate of 38% and false alarm rate of 10% using the same parameters from the cross validation process applied to TDT1 (this study). These results are within one standard deviation of the expected values.

The consolidation threshold was used to build lists of stories that were assumed to be related to each query. We tested various methods of combining stories that exceeded this threshold into one query. One of the methods for agglomerating queries used average link clustering [23, 18]. We found that agglomerating using low values for p had worse performance than agglomerating at higher values, but in general, agglomeration with good parameters had no effect on detection performance.

Our system processed 1300 stories per hour (sequentially) while agglomerating 10% of the incoming queries into previously created queries. It was tested on a 300 MHz DEC-Alpha running Unix.

```
q104 = #WSUM( 1.0

    1.175688 accident       1.125646 crash
    1.070033 plane          0.935901 cause
    0.935901 investigate    0.935901 look
    0.852374 air            0.852374 aircraft
    0.852374 survivor       0.752039 usair );
```

Figure 3: General "US Air plane crash" query.

4.5 Failure Analysis

Misses occur when stories containing new events are labeled as "not new". When the representation used a small number of features, misses were mostly the result of failing to weight specific event features more heavily than features descriptive of a class of events. For example, story 3057 is about the "Crash of US Air flight 427" (event 24). The query in Figure 3 uses 10 words created from story 104 and contains many features related to the broad class of plane crash events. Story 104 is *not* related to event 24 (it is about a different plane crash), but causes the system to miss the start of the event on 90% of the passes. Story 104's query becomes a general query for crashes of US Air planes. The classification of the "Oklahoma City bombing" (Event 18) had a similar problem because of a query created from a story about the earlier bombing at the World Trade Center (Event 25). When the representation includes more features, the two bombing events were separable, but the airline crashes were not. We expect that the use of phrases—e.g., "flight 427"—may help with these problems.

At higher dimensionality and using the best parameters, the system could not distinguish between stories from the "O.J. Simpson trial" (Event 9) and stories pertaining to other court cases. In addition, the corpus contained a heavy coverage of the events related to the problems in Bosnia, which caused our system to miss "Carter's visit to Bosnia" (Event 3). These examples indicate that the system was unable to detect certain events that are discussed in the news at different levels of granularity. However, we hypothesize that several of the problems revealed in the failure analysis could be resolved with a different weight assignment strategy for query features.

5 Event Tracking

The TDT evaluation approach is different than the more established TREC filtering task. The latter provides a large amount of training data with queries and relevance judgments, and requires that sites generate filtering queries that will work on a test set provided later. In the TREC-6 filtering track,[22] the training data includes anywhere from 6 to 887 relevant documents, with a mean of 123 (the routing track had between 8 and 2,431 relevant documents with a mean of 576). Although there are settings where that much training information is possible, it is difficult to argue that the setting is commonplace.

For the Event Tracking task, on the other hand, we are interested in substantially smaller numbers of training stories—in fact, we are interested in how few stories are needed for successful tracking. However, a more important problem for this task is that to model a real world setting, the tracking needs to begin as soon as possible after the training stories are "presented." Consider the case of tracking a bombing event: it is not interesting

to evaluate a tracking system on news that is reported weeks after the event—the goal of the system is to begin tracking *immediately*. Unfortunately, events occur at different times, meaning that it is nearly impossible to use the same training and test set for each event.

For those reasons, the TDT corpus is split into training and test information at a different point for each event. If the system is being evaluated for 4 training stories, then the training corpus is all stories up to and including the fourth training story and the test corpus is the remainder of the corpus. Note that this also means that different numbers of training stories create different training and test corpora.

In this study, we let the number of training stories, N_t, take on values 1, 2, 4, 8, and 16. If an event has fewer than N_t training stories it is neither tracked nor evaluated at that N_t value.[1] To compare system performance across N_t values, the system is trained on N_t stories, but always evaluated on the $N_t = 16$ test set—i.e., its performance on the stories between the N_t^{th} and the 16^{th} training story is ignored. Ten of the 25 events have fewer than 16 training stories, so cross-N_t comparisons are done using only 15 events.

The effect of this per-event, per-N_t separation of the corpus into training and test data is to create a "rolling" evaluation corpus. A tracking system will be testing some events while it is simultaneously training others.

5.1 Tracking algorithm

We approach the tracking problem using methods based primarily on Information Filtering. We represent stories by vectors of features. The features were found by applying a shallow tagger (part of Inquery) to the stories and selecting all nouns, verbs, adjectives, and numbers. Names of countries, states, and large cities were treated as single features by the tagger even if they are multiword. There was no stopword list, but most common stopwords do not fall into the parts of speech used. The features were stemmed.

Queries are represented by a similar vector of *tf·idf* feature weights. Queries and stories are compared by:

$$sim(Q, D) = \frac{\sum_{i=1}^{N} q_i \cdot d_i}{\sum_{i=1}^{N} q_i}$$

$$d_i = \frac{tf}{tf + 2} \cdot (1 - \log_N df_i)$$

where q_i is the weight of feature i in the query, d_i is the weight in the story, tf is the number of times the feature occurs in the story, df_i is the number of the times the feature occurs in the collection, and N is the number of stories in the collection. (This weighting function is a simplification of the more complex weighting scheme used for detection; it assumes that all stories are of roughly the same length, that the collection is never empty, and that df_i is not zero.)

As mentioned in Section 4.1, the statistics for the *idf* part of the weighting function are not known for the entire stream. In contrast to our solution for detection, our implementation for tracking utilized incremental *idf* based on the *current* values for N and df_i up to and including the last training story for each event. Query weights were held constant, and document weights were recalculated based on incrementally updated values. We ran experiments on the TDT corpus itself, but seeded the initial values with those obtained from an auxiliary corpus ("past"). In the experiments that follow, we used 31,188 CNN news stories from January 1, 1993, through June 30, 1994, which provided 18 months of data prior to the start date of the TDT corpus.

A final step in the scoring process normalizes scores across all events. The comparison function above results in a ranking of stories where the higher in the rank the more likely a story is to discuss the event in question. However, a score of 0.45 could mean "very likely to match" for one query and "very unlikely" for another query. Our goal is to normalize the scores so that 0.45 (and every other score) has roughly the same meaning no matter what query and story are being compared. This should result in more "meaningful" scores for stories and more appropriately matches the assumptions behind the DET curve discussed above.

To normalize scores, we calculate the similarity of the query against the N_t training stories and find the average of those similarities. That average value is used as a normalization factor, and all scores (for that event) are divided by it. As a result, although scores can range from zero through well above 1.0, a particularly "good" story (for any event) should score 1.0 or higher. That is, its unnormalized score will be near those of the N_t training stories, so dividing by that average score, will result in a normalized score near 1.0. The interpretation of 1.0 as "very like the training stories" is more likely to be true across all events than before normalization.[2]

5.2 Using common words

The simplest approach to tracking is to select useful words or phrases from the N_t training stories and use those to form a query and a threshold for matching the query. As in any filtering task, all subsequent stories are compared against the query and, if the match is above the threshold, the story is selected—here, it is assumed to be about the same event.

We used the top n most commonly occurring features in the N_t training stories, with weight equal to the number of times the feature occurred in those stories multiplied by its incremental idf value (set after the N_t^{th} story). We found that performance was very similar across all values of n, though larger queries are generally less effective; the optimal values appear to be 10-20 features. We suspect a small number of features is sufficient because news reporting tends to refer to an event with few words and phrases, relying on the audience's having a context to recognize the event: capturing the one or two "killer features" is sufficient to track the event with high accuracy. We have not investigated this issue in depth.

Figure 4 shows 10-feature queries at several values of N_t. The curves show that more training helps the performance, but that by the time there are four sample stories, adding more provides little help. This rapid peaking of effectiveness is similar to that observed in the TREC routing tasks.[1]

Early efforts found that incremental *idf* information from the "past" corpus hurts in both cases when compared to using the less-accurate "tdt" idf statistics. It

[1] Note that this means that for larger values of N_t, only heavily reported events will be evaluated. Results for one value of N_t should not be assumed valid for other values.

[2] We have not yet performed any statistical analysis of the normalization to know details of its effect. It does improve the detection error tradeoff as represented by the DET curve, so we believe that we are achieving something useful for situations where that sort of measure is important.

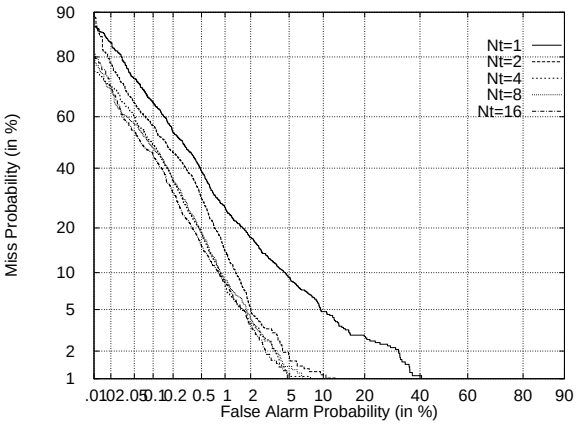

Figure 4: Comparing values of N_t. Once N_t reaches 4, adding more stories for training is only marginally helpful.

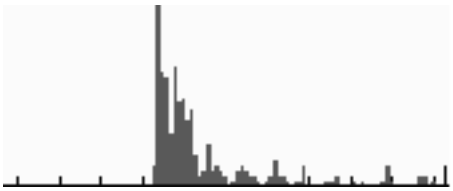

Figure 5: Number of news stories per day covering the OK city bombing event.

appears that the lack of retrospective data (in "tdt") creates more volatile *idf* values that drop rapidly and create a time penalty similar to that used in detection (section 4.2). When we explicitly model a time penalty in the tracking task, the incremental *idf* works equivalently in both corpora. It is not clear that incremental *idf* is of greater value than a static retrospective *idf* such as that used in the detection task.

5.3 Surprising features

It is a characteristic of news reporting that stories about the same event often occur in clumps. This effect is particularly true for unexpected events (e.g., disasters or major crimes) where the news media exhibit strong interest in a story and report in nearly endless detail about it. As the triggering event fades into the past, the stories discussing the event similarly fade. For example, Figure 5 shows how many stories appeared per day in the TDT corpus for the Oklahoma City bombing event. The sudden rise and then gradual fall of the stories is characteristic of this type of event.

A second characteristic of news coverage is that the people, places, and other items of interest in a story are likely not to have been mentioned very often in the past. This supposition is obviously not true for all features (e.g., the name of the President of the U.S. is likely to reoccur), but there must be *something* about the story that makes its appearance worthwhile. We call those features that have not occurred recently *surprising*.

An analysis of the events in this corpus shows that this effect is almost always true. We measure surprise based on the distance between this occurrence of a word and all past occurrences. For document sequence number *docid*, and word number *word*:

$$Surprise(word, docid) = \left(\sum_{i=1}^{df_{word}-1} \frac{1}{\log(docid - id_{word\,i})} \right)^{-1}$$

where df_i is the number of stories to date containing word i and id_i is the sequence number of the most recent story that contained word i. The formula is the inverse of the sum of the inverses of the log of the distances from this word to all of its previous occurrences.[3] Table 2 shows the surprise values of selected words for some of the events. Of interest are words like *kobe* that are very surprising and occur frequently in the $N_t = 4$ training stories and *earthquake* which is entirely unsurprising but still common in those same stories. In general, we find that words that are common in the training set but that have little to no surprise value represent "event class" features covering broad news areas such as politics, death, destruction, and warfare. (We expect that event-class- and event-level features can be combined in a meaningful way, perhaps with the class features providing a form of "query zone."[19])

Because many of the "surprising" features appear to be strong indicators of the event being discussed, we had expected they could be used to build superior tracking queries. Unfortunately, the evaluation does not support that hope. Two problems arise: (1) the surprising words do not provide a broad enough coverage to capture all stories on the event (e.g., omitting "bosnia" for an event in that area of the world because it is not a surprising word), and (2) many of the words are useless for retrieval, either because they are misspellings or because they are surprising by chance (e.g., the name of someone interviewed). For retrospective tasks, where a feature's occurrence characteristics can be measured *after* its "surprising" appearance, we expect that a measure of a feature's

Feature	Surprise	rcf	rdf	Event
Kobe	1.29	19	4	Kobe
427	2.30	5	3	USAir 427 crash
cessna	1.09	5	4	Cessna
f-16	0.13	14	4	F-16
dna	0.11	15	4	OJ & DNA
lawn	0.05	17	4	Cessna
quake	0.13	13	3	Kobe
OKCity	0.25	12	4	OKCity
Breyer	0.22	36	4	Breyer
Intel	0.14	35	4	Pentium chip flaw
Rosario Ames	inf	14	3	Spy
bosnia	0.00	50	4	F-16
earthquake	0.04	27	4	Kobe
death	0.00	10	2	Salvi
death	0.00	5	3	OKCity

Table 2: Surprise values for a few words/phrases and a few events. The feature is shown along with its surprise value, the number of time it occurs in the $N_t = 4$ training set, the number of those 4 stories it occurs in, and the name of the event being considered.

[3] This particular formula is primarily *ad hoc* to explore its value, though it is supported by some data exploration and empirical evidence. An information theoretic or probabilistic measure may prove more appropriate when we have a better understanding of the task.

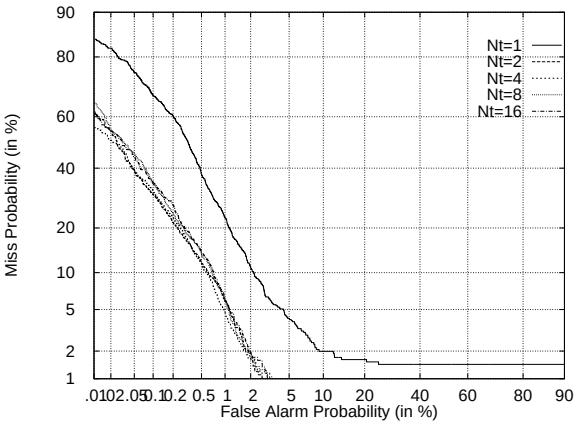

Figure 6: Adaptive filtering and the number of training instances.

surprise value and its subsequent longevity may provide more useful information.[4]

5.4 Adaptive tracking

One of the reasons for a query's inability to track stories is that the discussion of an event changes over time. This effect is particularly well illustrated by the Oklahoma City bombing event. When the bomb exploded outside the Murrah building, its origin was a mystery. Six days later, Timothy McVeigh was arrested and charged with the crime. Indeed, there is no mention of McVeigh until the 61st story that is relevant to that event, so using the approaches of Section 5.1, it is impossible for his name to appear in a query for any value of N_t less than that. Several other events exhibit similar reporting characteristics, and a tracking method that accommodates the shifting reportage should prove useful. The issue is similar to drifting queries in information filtering.[1, 12]

We have implemented an adaptive version of the tracking system that may rebuild the query after it "tracks" a news story on a given event. This idea is a form of unsupervised learning and except for its incremental nature, is similar to the notion of "pseudo-relevance feedback" that has proved highly successful at TREC workshops.[22]

When a tracking query is first created from the N_t training stories, it is also given a threshold. We used an initial threshold of 0.8 for all events (recall that scores range from zero to just over one). During the tracking phase, if a story S scores over that threshold, we assume that S is relevant and the query is regenerated as if S were among the N_t training stories—so there are $N_t + 1$ then $N_t + 2$ and so on training stories.

The adaptive approach is highly successful at generating superior queries, particularly at low false alarm rates. The threshold value has a noticeable impact on the effectiveness of the adapting. The higher the threshold, the less likely a query is to be regenerated. We found that a threshold of 0.8 improves performance, and one of 1.0 hurts it. Smaller thresholds (e.g., 0.6) cause performance to get consistently worse because they are adding stories that are less and less likely to be relevant.

[4]Preliminary studies in the Retrospective Detection task of the TDT pilot study[3] support this intuition.

One of the nice features of adaptive tracking is that when it works well, it allows the system to operate effectively with fewer sample stories. Figure 6 suggests that two sample stories is sufficient to achieve high-quality tracking: adding more causes almost no improvement—that effect is not achieved until $N_t = 4$ without adapting. Although the $N_t = 1$ curve is noticeably worse that then others, a comparison with Figure 4 shows that it still results in a substantial improvement in effectiveness.

We started off this section by talking about problems with words such as "McVeigh" and their not appearing in early stories. The final queries for $N_t = 4$ show that adaptive tracking successfully captures those features. [Timothy] *McVeigh* and [Terry] *Nichols* (the two suspects that have since been convicted) are both added to the Oklahoma City bombing event even though no mention is made of them until after the 60th stories; Scott O'Grady's name appears in the event describing the downing of F-16 pilot in Serbian territory, though it is six days and 38 stories later that the name is revealed.

We believe that adaptive event tracking is a necessary approach to this problem, as long as the system must work without user feedback after the N_t stories. We hypothesize that "surprise" information will be a useful indicator of valuable new features in adapting: a feature that appears suddenly and persists for a few stories is very likely to be important to the event.

6 Conclusions and future work

New event detection is an abstract document classification task that we have shown has reasonable solutions using a single pass clustering approach. We have presented an evaluation methodology based on miss and false alarm rates, measures that are more closely related to the task than recall and precision. System misses and false alarms were used to measure detection error in a cross-validation approach that found stable system parameters for our implementation. We described overall system performance using a bootstrap method that produced performance distributions for the TDT corpus.

New event detection shares some characteristics of online information filtering. The emphasis on time and "event" rather than general "topic" should eventually lead to different methods for processing the arriving text. Which approaches and how well they work remain open questions. Other questions include: How can we describe the relationship between two events, or between an event and a sub-event? Will we need user models to capture preferred notions of event granularity, or are there broadly acceptable definitions? Is there a way to select only "interesting" events from the stream of news and exclude entirely uninteresting stories? Is this an application where natural language processing could help identify features related to who, what, where, and when?

To illustrate the Event Tracking problem, we constructed and evaluated a system that built simple event queries. We showed that very few training stories are needed to build a high-quality query with a small number of features. We discussed the notion of surprising features and showed how adaptive tracking is a useful method for capturing those features in story sequences about disaster or crime events, and for reducing the number of training stories needed. We also presented an evaluation methodology for Event Tracking that uses "rolling" training and test sets, and a Detection Error Tradeoff (DET) plot to measure accuracy on the system.

Significant advances in Event Tracking accuracy are most likely to be obtained using some limited form of story parsing and "understanding." It is likely to be useful to capture notions of who, what, where, when, why, and how—although the well-known past experience from IR suggests that the gains may not be large.

Acknowledgments

We thank Charles Wayne, George Doddington, Yiming Yang, Jaime Carbonell, and Jon Yamron with whom we worked to define the Topic Detection and Tracking tasks. George Doddington deserves particular credit for his guidance in establishing the evaluation methods for the TDT effort. We are also grateful to David Jenson and Warren Greiff for their comments on our parameter estimation technique and Mike Scudder for help with the evaluation software. A suggestion of Bruce Croft's about statistically anomalous features led eventually to the idea of "surprising" features. Jay Ponte and a total of twelve anonymous reviewers provided helpful critiques of earlier drafts of this work.

The TDT initiative is a DARPA-sponsored project that supported this work. This material is also based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

[1] J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of SIGIR '96*, pages 270–278, 1996.

[2] J. Allan, L. Ballesteros, J. Callan, W. Croft, and Z. Lu. Recent experiments with inquery. In *The Fourth Text REtrieval Conference (TREC-4)*, pages 49–63, 1995.

[3] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998. Forthcoming.

[4] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proceedings of SIGIR '95*, pages 351–357, 1995.

[5] J. Callan. Document filtering with inference networks. In *Proceedings of SIGIR '96*, pages 262–269, 1996.

[6] C. Carrick and C. Watters. Automatic association of news items. *Information Processing & Management*, 33(5):615–632, 1997.

[7] P. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, 1995.

[8] G. DeJong. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 3:251–273, 1979.

[9] P. Hayes, L. Knecht, and M. Cellio. *A News Story Categorization System*, pages 518–526. Morgan Kaufmann Publishing, San Francisco, 1997. Originally appeared in *Proceedings of the 2nd Conference on Applied Natural Language Processing*, 1988.

[10] K. S. Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishing, San Francisco, 1997. Chapter 4, pages 167-256.

[11] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.

[12] W. Lam, S. Mukhopadhyay, J. Mostafa, and M. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proceedings of SIGIR '96*, pages 317–325, 1996.

[13] D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of SIGIR '96*, pages 298–306, 1996.

[14] D. D. Lewis. The TREC-5 filtering track. In E. M. Voorhees and D. K. Harman, editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 75–96, Nov. 1997. NIST Special Publication 500-238.

[15] A. Martin, T. K. G. Doddington, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of EuroSpeech'97*, volume 4, pages 1895–1898, 1997.

[16] B. Masland, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *Proceedings of SIGIR '92*, pages 59–65, 1992.

[17] R. Papka, J. Callan, and A. Barto. Text-based information retrieval using exponentiated gradient descent. In *Proceedings of the 10th Annual Conference of Advances in Neural Information Processing Systems*, pages 3–9, 1996.

[18] G. Salton. *Automatic Text Processing*. Addison-Wesley Publishing Co, Massachusetts, 1989.

[19] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *Proceedings of SIGIR '97*, pages 25–32, 1997.

[20] J. Tague-Sutcliffe. Measuring the informativeness of a retrieval process. In *Proceedings of SIGIR '92*, pages 23–36, 1992.

[21] C. van Rijsbergen. *Information Retrieval, 2ed.* Butterworths, Massachusetts, 1979.

[22] E. M. Voorhees and D. Harman. Overview of the sixth text retrieval conference. In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference (TREC-6)*, 1998. NIST Special Publication, forthcoming.

[23] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[24] Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In *Proceedings of SIGIR '98*, 1998.