On-Line New Event Detection using Single Pass Clustering

Ron Papka and James Allan

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

{papka,allan}@cs.umass.edu

Abstract

This paper discusses the implementation and evaluation of a new-event detection system. We focus on a strict on-line setting, in that the system must indicate whether the current document contains or does not contain discussion of a new event before looking at the next document. Our approach to the problem uses a single pass clustering algorithm and a novel thresholding model that incorporates the properties of events as a major component. A corpus containing newswire and transcribed broadcast news was analyzed using our system, and our results compared favorably to those of other systems. We develop an evaluation methodology based on a combination of techniques that allows us to infer the expected performance of our approach in the field, and to suggest avenues for future research that may lead to better performance.

1 Introduction

On-line event detection is the classification of a time-ordered stream of documents that discuss events. The task of *new-event detection* is to identify documents discussing an instance of a new event, that is, one which has not already appeared on the stream of incoming text.

The environment we envision for on-line event detection is depicted in Figure 1. Broadcast news from various sources is monitored by the system that operates on text documents. When the source is radio or television, a speech transcription and segmentation process first converts the data to text documents, each containing a complete news story [17, 2]. If a document discusses a new event, it is fed to a tracking process that classifies the document stream based on existing news stories in an on-line unsupervised setting. The new-event detection system has two modes of operation: In on-line mode, a strict real-time application is assumed, and the system indicates whether the current document contains or does not contain discussion of a new event before looking at the next document. In delayed mode, the system can base its decision on information in "future" documents.

There are several practical applications that would benefit from a good solution to new-event detection. This task is currently performed manually by equity traders, media analysts, and on-line digital news editors that focus on collecting, interpreting, and presenting news from multiple news sources. A system that could organize events automatically would be useful for financial and world news applications where decision-making processes are based on new events and the evolution of existing events.

In the following section, previous work related to on-line event detection is discussed. Our algorithm for new-event detection is presented in Section 3. An implementation of the algorithm was tested using the experiments described in Section 4. In Section 5, we discuss these results and those obtained from two other systems.

2 Related Work

The primary motivation for this work arose out of a project called Topic Detection and Tracking (TDT). This project (joint with DARPA, CMU, and Dragon Systems) set out to explore issues related to detecting and tracking events in broadcast news. The results of the first year's efforts were reported at a workshop in October, 1997 [20].

One of the issues explored in the pilot study was the meaning of *event*, which was defined as some unique thing that happens at some point in time. The property of time is what distinguishes an event from the more general *topic*. For example, "The Eruption of Mt. Pinatubo on June 15th, 1991" is considered an event and "volcanic eruptions" is a more general topic. This definition can be extended to include the obvious spatial component of an event, namely loca-

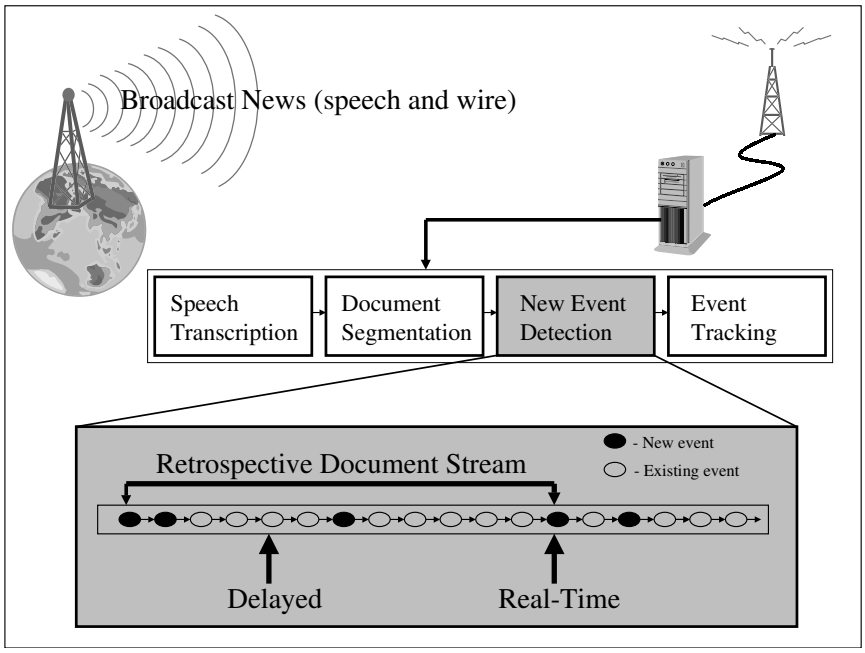


Figure 1: On-line event detection and operation modes for new-event detection.

tion. For example, The "Earthquake in Kobe, Japan" is a description of an event that uses this property.

Though it is difficult to determine the accurate definition of an event, it is easier to define the properties that specify when two events are the same. These properties define *event identity*. They are important to model, because a system that has the capability of representing these properties in order to detect event identity is trivially capable of performing new-event detection: Using an event identity process, the system determines for the current document whether it contains an event identical to one appearing in a previously processed document. If so, the system does not detect a new event; otherwise it does.

From a journalist's perspective, a news story about an event will typically specify 1) when the event occurred; 2) who was involved; 3) where it took place; 4) how it happened; and 5) the impact, significance, or consequence of the event on the intended audience [15]. However, as an event evolves, many of these properties are either not initially known, or are assumed to be known by the audience and are therefore not referenced within the text of documents relating to the same event. The lack of certain event properties within documents relating to the same event should be expected as the event evolves.

At a certain level of abstraction, the task of new-event detection is a classification task where documents are either members of the set of documents containing a new event or not. Previous work related to on-line document classification has often focused on supervised-training algorithms that use labeled documents [18, 3, 13, 16], a resource not available during new-event detection.

Most experimental Information Retrieval (IR) literature discusses evaluation using topics and not events. In [10], for example, a knowledge-based approach to text pattern-matching was used to categorize news stories into general topics. In addition, almost all TREC [9] information requests for the routing and filtering experiments are about topics even though the testing and training corpora are mostly from newsprint sources.

Some event-related work has been reported prior to the TDT workshop, but new-event detection has not been a focus. A frame-based system that attempted to detect events on a UPI newswire was constructed by DeJong in the late 1970s. He used pre-specified software 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 document contained an event. It chose the correct script with a classification accuracy of 38% for the documents for which it had a sketchy script.

Recent work by Carrick and Watters discussed an application that matches news stories to photo captions using a frame-based approach modelling some of the properties of events [6]. They concluded that using the extracted lexical features in a word-cooccurrence

retrieval model was nearly as effective as using the same features in their frame-based approach.

The frame-based approaches are perhaps useful for a specific domain; however, 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-cooccurrence retrieval in a process that specifically models event-level features in addition to topic-level features.

3 New-Event Detection Algorithm

If new events were to be sought from a time-ordered static collection, one solution would be to use document clustering techniques [22, 19] to cluster the collection, and then to return the document from each cluster containing the earliest timestamp. However, we are interested in the strict on-line case, which has real-time constraints and imposes a single pass restriction over the incoming stream of documents.

We have developed a solution to new-event detection using a modification of the single pass clustering algorithm described in [21]. Our algorithm processes each new document on the stream sequentially, as follows:

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

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

4 Experiments

4.1 Implementation

The new-event detection algorithm was implemented by combining the ranked-retrieval mechanisms of Inquery [4], a feature extraction and selection process based on relevance feedback [1], and the routing architecture of InRoute [5].

For any comparison of document d to query q we used the evaluation function of the #WSUM operator of Inquery:

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

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:

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

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 incorporated here utilizes several volumes from the TREC collection as an auxiliary corpus [9].

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 document(s). The query feature weight was the average value using the tf component calculation described above.

4.2 Thresholding Model

One of our hypotheses regarding new-event detection is that exploiting time will lead to improved performance. A side-effect of broadcast news is that documents closer together on the stream are more likely to discuss related events than documents further apart

on the stream. When a significant new event occurs there are usually several documents 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 document 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.

The second factor of the model is a time penalty tp that increases the threshold for a query based on its separation in time from the document being evaluated. If the jth document is compared to the query resulting from the ith document, 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 document 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 documents from the stream. The classifiers in our implementation are queries and their thresholds.

4.3 Data

The following experiments use the TDT corpus and relevance judgements which have been published by the Linguistic Data Consortium [1]. The text comprises 15863 documents evenly distributed between CNN transcribed broadcast news and Reuters newswire dating from July 1, 1994 to June 30, 1995. The average document contains 460 (210 unique) single-word features after stemming and removing stopwords.

Relevance judgements were assessed for documents pertaining to the 25 events listed in Figure 2. The judgements were obtained by two independent groups of assessors and then reconciled to form a set of final judgements. Documents were judged on a ternary scale to be non-relevant, to have content relevant to the event, or to contain only a brief mention of the event in a generally non-relevant document. 1132 documents were judged relevant, 250 documents were judged to contain brief mentions, and 10 documents overlapped between the set of relevant documents and the set of brief mentions. Overlaps and brief mentions were removed from the corpus before processing, leaving 1124 relevant documents for evaluation.

[1] www.ldc.upenn.edu

1. Aldrich Ames spy case
2. The arrest of 'Carlos the Jackal'
3. Carter in Bosnia
4. Cessna crash on White House lawn
5. Salvi clinic murders
6. Comet collision with Jupiter
7. Cuban refugees riot in Panama
8. Death of Kim Jong II
9. DNA evidence in OJ trial
10. Haiti ousts human rights observers
11. Hall's helicopter down in N. Korea
12. Flooding in Humble, Texas
13. Breyer's Supreme Court nomination
14. Nancy Kerrigan assault
15. Kobe Japan earthquake
16. Detained U.S. citizens in Iraq
17. New York City subway bombing
18. Oklahoma City bombing
19. Pentium chip flaw
20. Quayle's lung clot
21. Serbians down F-16
22. Serb's violation of Bihac safe area
23. Faulkner's admission into the Citadel
24. Crash of US Air flight 427
25. World Trade Center bombing

Figure 2: 25 judged events in TDT corpus.

The document frequency for features was obtained from an auxiliary corpus. The corpus comprises Tipster volumes 1, 2, 3, and the TREC-4 Routing volume. The volumes contain 1,246,925 documents from the Associated Press(1988-90), Department of Energy abstracts, Federal Register(1988-9), San Jose Mercury News(1991), Wall Street Journal(1987-91), and Ziff-Davis Computer-select articles. The average document contained 421 single-word features after stemming and removing stopwords.

4.4 Evaluation Methodology

The task being evaluated is the detection of new events without the use of relevance judgements. Evaluation is based only on system performance for the 25 events listed in Figure 2. The performance metrics that are meaningful for new-event detection are system miss and false alarm rates. They measure performance error. Misses occur when the system does not detect a new event, and false alarms occur when the system indicates a document contains a new event, when in truth, it does not. When the user has an equal aversion to misses and false alarms, the Euclidean distance between a performance point and the origin can be used as a single-value metric that combines both misses and

false alarms. In addition to these performance metrics we calculated the traditional recall and precision metrics, and F1-Measure [12]. More specifically, assume a system returns a set of documents S it flags as discussing new events, and

a = number in S discussing new events,
b = number in S not discussing new events,
c = number in $\bar{S}$ discussing new events,
d = number in $\bar{S}$ not discussing new events;

then,

$Recall = R = \frac{a}{a+c}$,
$Precision = P = \frac{a}{a+b}$,
$F1\text{-}Measure = \frac{2PR}{P+R}$,
$Miss\ Rate = 1 - Recall = M = \frac{c}{a+c}$,
$False\ Alarm\ Rate = Fallout = F = \frac{b}{b+d}$, and
$Distance\ from\ Origin = \sqrt{M^2 + F^2}$.

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 documents in which one of the 25 known events first occurs. The second pass excluded these documents, and the goal was to detect the second document for each of the 25 known events: the second document artificially becomes the first document in the stream. The process iterates to skip up to 10 documents for each event. If an event contained fewer documents than the number of documents 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 overfitting 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*.

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.

4.5 Results

The results for the new-event detection system using queries containing between 5 and 400 single-word features are listed in Figure 3. Performance in this graph is based on the Euclidean distance average miss rate and false alarm rate are from the origin. 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 dimensionality greater than 75. 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 all the single-word features available in its corresponding document. 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 documents $1..n$ were removed from the stream, and the goal was to detect the $(1+n)$-th document for each event. Hence, a skip value of 1 implies that the second document 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.

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.

The effects of the time penalty in the threshold model are illustrated in Figure 4. 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

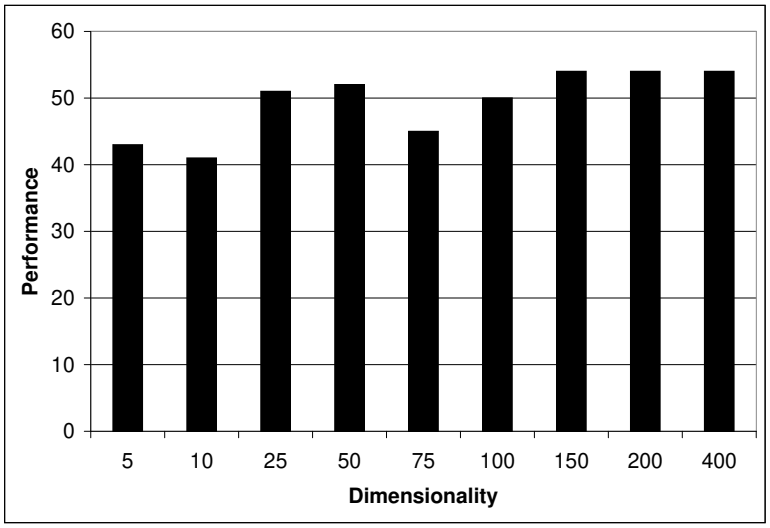


Figure 3: (Performance = 100 – *Distance from Origin*) vs. maximum number of query features.

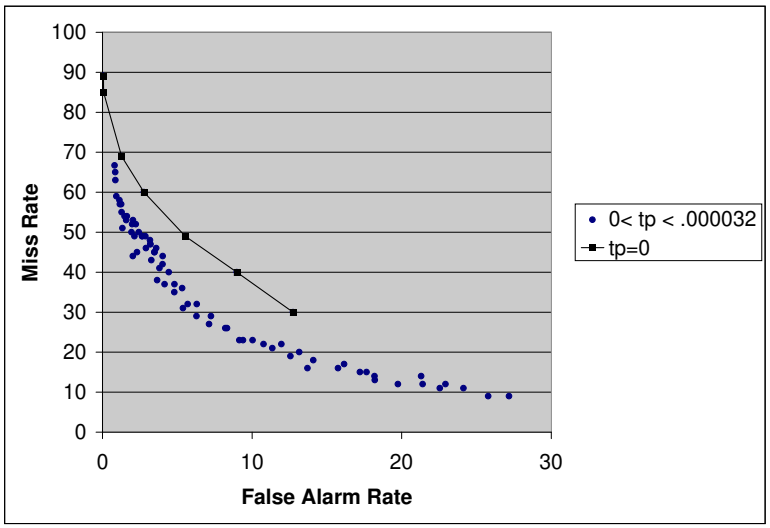


Figure 4: Effects of varying threshold parameters p and tp. (Axes have different scales.)

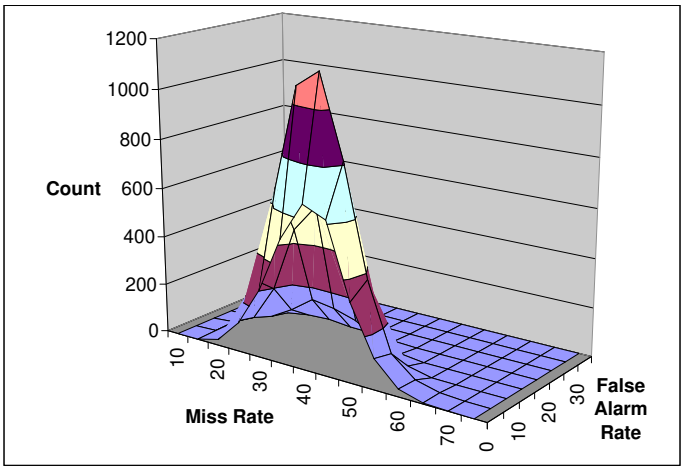


Figure 5: Performance distribution based on bootstrap procedure.

penalties. On average, for any value of p, performance is better when $tp > 0$.

We ran the bootstrap process for 10,000 iterations to produce the 3D-histogram of performance in Figure 5. The process yielded samples having a mean miss rate of 40.5% with a standard deviation of $\pm 7.6\%$, and a mean false alarm rate of 7.8% with a standard deviation of $\pm 4.0\%$.

The consolidation threshold was used to build lists of documents that were assumed to be related to each query. We tested various methods of combining documents that exceeded this threshold into one query. One of the methods for agglomerating queries used average link clustering [22, 19]. 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.

In terms of real-time performance, our system ran at 1300 documents per hour while agglomerating 10% of the incoming queries into previously created queries. It was tested on a 300 MHz DEC-Alpha running Unix.

4.6 Failure Analysis

Misses occur when stories containing new events are labeled as "not new". At low dimensionality, misses were mostly attributed to the inability of the feature extraction process, and thus the query representation, to weight event-level features more heavily than the more general topic-level features. For example, document 3057 is about the "Crash of US Air flight 427" (Event 24). But the query using 10 words created from document 104 (shown in Figure 6) contains many features pertaining to the more general topic of plane crashes. The system misses Event 24 on 90% of the passes, because the first documents for the event triggered the query already created from document 104, which is about the "Crash of US Air flight 1016". Query 104 becomes a general query for US Air crashes. The classification of the "Oklahoma City bombing" (Event 18) had a similar problem stemming from a query created from a document about the earlier bombing at the World Trade Center (Event 25). At higher dimensionality, the two bombing events were separable, but the airline crashes were not. However, the use of phrases, such as "flight 427", may help with these problems.

```
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 6: General "US Air plane crash" query.

At higher dimensionality and using the best parameters, the system could not distinguish between documents from the "O.J. Simpson trial" (Event 9) and documents 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 TDT System Comparison

5.1 CMU and Dragon Approaches

The TDT workshop included an evaluation of three new-event detection systems. Each research group used an approach based on single pass clustering. Our system is described above.

The CMU system used SMART for query and document representations. CMU used a clustering strategy with a *detection threshold* that governed the minimum document-cluster similarity score required for the system to label the current document as containing a new event. They also used a *combining threshold*, which was the minimum similarity score required for adding a document to an existing cluster. Time was incorporated in the detection decision by limiting comparison to documents which appeared within a constant window size of time from the document being processed. They reported that experiments using a cluster representation between dimensionality 50 and 100 yielded the best results. They also reported that experiments using no agglomeration yielded better results than those using agglomeration.

The Dragon system used single word (unigram) frequencies for cluster and document representations: their document representation did not use *tf.idf* scores, which were used by the other systems. Their document-cluster comparison function is a modification of the Kullback-Leibler distance measure. The modification included a decay term which decreased the similarity measure for clusters containing documents closer in sequence to the current document on the stream. In addition, they used a pre-processing step in which an iterative *k-means* clustering algorithm was used to build 100 background models (clusters) from an auxiliary corpus. In their model, a document is considered to contain a new event when it is closer to a background model than to an existing story-cluster.

5.2 Cross-System Comparisons

The detection results using the data and the systems described above are presented in Figure 7. The UMASS and CMU systems are using representations of dimensionality 100, and the Dragon system is using full dimensionality. The figure is a Detection Error Tradeoff (DET) graph that illustrates the *estimated* performance error tradeoff between miss rate and false alarm rate [14]. The graph is scaled based on a normal distribution of the performance metrics. Each curve can be viewed as an analogue to a recall-precision curve which is used to depict the retrieval performance tradeoff between recall and precision. The points on the curve are determined by varying an external threshold parameter applied to the ranked list of decision scores a system produces for each document. Points closer to the origin indicate better overall performance. The graph also contains the evaluation point corresponding to pooled average performance.

The UMASS system has miss rates that are lower than the other systems between false alarm rates of 1% and 10%. Below the 1% false alarm rate, the UMASS and CMU systems outperform the Dragon system. At the 10% level of false alarms the systems converge, and at 30%, the UMASS system experiences higher miss rates than the CMU and Dragon systems.

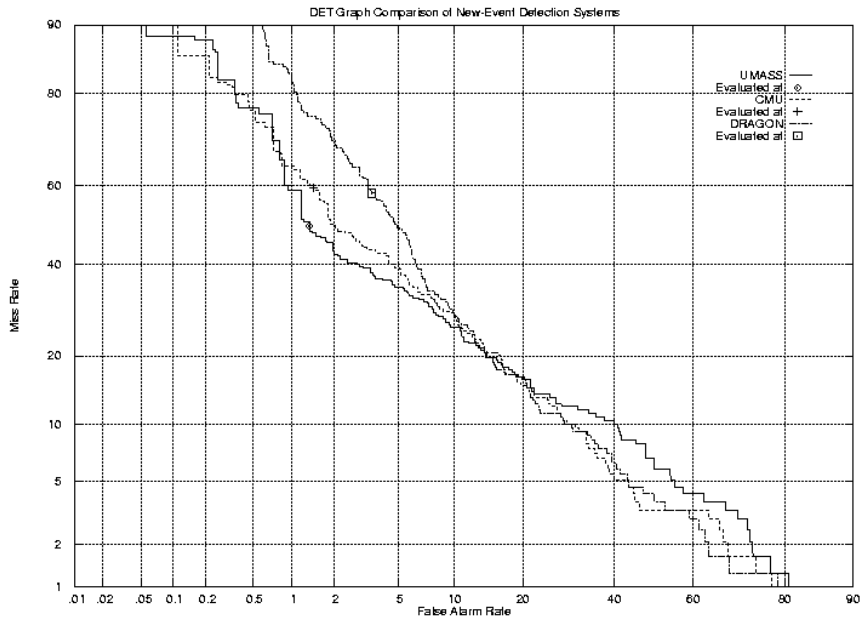
A comparison of average performance using various metrics is also listed in Figure 7. The UMASS system has a miss rate that is 18% lower than the other systems at similar false alarm levels. The difference in performance indicates that the UMASS system, on average, correctly detects 2 additional new events. This improvement is not significant based on a sign test across event level performance.

6 Conclusion 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 performance error in a cross-validation approach that found stable system parameters for our implementation. We describe overall system performance using a bootstrap method that produced performance distributions for the TDT corpus.

In retrospect, the comparison of the TDT systems indicated that different views of the task lead to different text representations resulting in similar performance. In addition to performance, the common element among the systems is an underlying model of word-cooccurrence that is used to determine when two documents discuss the same event. This model has been previously used to classify documents into more general topics, which suggests that performance improvements will come from modelling the properties of events, more so than modifying the existing retrieval mechanisms.

New-event detection shares some characteristics of on-line information filtering. The emphasis on time



System	Miss Rate	F/A Rate	Recall	Precision	F1
UMASS	50%	1.34%	50%	45%	0.45
CMU	59%	1.43%	41%	38%	0.39
DRAGON	58%	3.47%	42%	21%	0.28

Figure 7: Comparison of systems presented at the first TDT workshop.

and "event" rather than general "topic" should lead to different methods for processing the arriving text. Which approaches can be leveraged 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?

Our future work for the strict on-line case of new-event detection will focus on the aspects of the implementation that caused detection errors. These problems included aspects of :

1. Parameter value estimation.
2. Feature extraction and selection.
3. Feature weight assignments.

The current implementation uses constant dimensionality and threshold parameters. Our experiments indicated that a good set of values existed for these parameters in 80% of the events tested. We expect significant performance improvements if good values are determined automatically for each query. The feature extraction process that builds queries should be extended to include multiword features. Preliminary experiments which tested features comprising constant sized windows of pairs of words appeared to provide performance improvements at low dimensionality. Another problem involved the feature weights, which are based only on a feature frequency component. We plan to conduct tests that adjust feature weights in an unsupervised setting, based on the document labels implied by the threshold model. We expect that this additional step is needed to get detection performance improvements through agglomeration. The goal of the weight learning process will be to weight event level features more heavily than general topic level features. With these issues resolved, we foresee an extended framework that can solve other abstract event classification problems in an unsupervised setting. One example is whether a document contains good news or bad news.

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. We are also grateful to David Jenson and Warren Greiff for their comments on our parameter estimation technique, Mike Scudder for help with the evaluation software, and Jay Ponte for valuable comments on an earlier draft of this study.

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, L. Ballesteros, J. Callan, W.B. Croft, and Z. Lu, "Recent Experiments with Inquery," *Proceedings of TREC-4*, 49-63, 1995.

[2] D. Beeferman, A. Berger, and J. Lafferty, "Text Segmentation Using Exponential Models," *Proceedings for Empirical Methods in NLP*, 1997.

[3] C. Buckley and G. Salton, "Optimization of Relevance Feedback Weights," *Proceedings of SIGIR*, 351-357, 1995.

[4] J. P. Callan, W.B. Croft, and S.M. Harding, "The INQUERY Retrieval System," *Database and Expert Systems Applications: Proceedings of the International Conference in Valencia Spain.* A.M. Tjoa and I. Ramos eds., Springer Verlag, New York, 1992.

[5] J.P. Callan, "Document Filtering With Inference Networks," *Proceedings of SIGIR*, 262-269, 1996.

[6] C. Carrick, and C. Watters, "Automatic Association of News Items," *Information Processing & Management*, 33(5):615-632, 1997.

[7] P.R. 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] D.K. Harman, *Proceedings of Text REtrieval Conferences (TREC)*, NIST Special Publication, 1993-7.

[10] P.J. Hayes, L.E. Knecht, and M.J. Cellio, "A News Story Categorization System," *Proceedings of the 2nd Conference on Applied Natural Language Processing* (1988), reprinted in *Readings in Information Retrieval*, K. Sparck Jones and P. Willet editors, Morgan Kaufmann Publishing, San Francisco, CA, 518-526, 1997.

[11] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.

[12] D.D. Lewis, and W.A. Gale, "A Sequential Algorithm for Training Text Classifiers," *Proceedings of SIGIR*, 3-13, 1994.

[13] D. Lewis, R. Schapire, J. Callan, and R. Papka, "Training Algorithms for Linear Text Classifiers," *Proceeding of SIGIR*, 298-306, 1996.

[14] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET Curve in Assessment of Detection Task Performance," in *Proceedings of EuroSpeech'97*, 4:1895-1898, 1997.

[15] P. E. Mayeux, *Broadcast News: Writing & Reporting*, 2ed, Brown & Benchmark Publishers, Guilford CT, 1996, p. 79.

[16] R. Papka, J.P. Callan, and A.G. Barto, "Text-Based Information Retrieval Using Exponentiated Gradient Descent," *Proceedings of the 10th Annual Conference of Advances in Neural Information Processing Systems*, 3-9, 1996.

[17] J.M. Ponte and W. B. Croft. "Text Segmentation by Topic," *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, 113-125, 1997.

[18] J.J. Rocchio, "Relevance Feedback in Information Retrieval," in *The Smart System - Experiments in Automatic Document Processing*, 313-323, Englewood Cliffs, NJ: Prentice Hall Inc. 1971.

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

[20] *Proceedings of the TDT Workshop*, University of Maryland, College Park, MD, October 1997. (Unpublished)

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

[22] P. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing & Management*, 24(5): 577-597, 1988.