Learning Approaches for Detecting and Tracking News Events

Yiming Yang, Jaime G. Carbonell, Ralf D. Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu, Language Technologies Institute, Carnegie Mellon University

THE AUTHORS EXTEND EXISTING SUPERVISED-LEARNING AND UNSUPERVISED-CLUSTERING ALGORITHMS TO ALLOW DOCUMENT CLASSIFICATION BASED ON THE INFORMATION CONTENT AND TEMPORAL ASPECTS OF NEWS EVENTS.

THE RAPIDLY GROWING AMOUNT of electronically available information threatens to overwhelm human attention, raising new challenges for information retrieval (IR) technology. Traditional query-driven retrieval is useful for content-focused queries but deficient for generic queries such as "What happened?" or "What's new?"

Consider, for example, a person who has just returned from an extended vacation and needs to find out quickly what happened in the world during her absence. Reading the entire news collection is a daunting task, while generating specific queries without any knowledge of recent events is rather unrealistic. Or consider a foreign-policy specialist who wants to study the Asian economic crisis, including precursor and consequent events. A keyword-based search on the query "Asian economy crisis" would most likely miss many relevant stories about the stock market crashes in Indonesia or Korea, banking-sector insolvency in Japan, or Jusuf Habibi's rise to power in Indonesia.

In other words, query-based retrieval is useful when you know more precisely the nature of the events or facts you're seeking. It is less useful when you want specific information but can only formulate a larger category sharing few if any terms with the potentially most useful texts. In short, retrieval based on immediate-content-focused queries is often insufficient for obtaining a variety of relevant stories and tracking the gradual evolution of events through time.

In the examples above, the user would have equal difficulty formulating the "right query" or "right level of abstraction," or checking all the potentially relevant stories. What would be desirable is an intelligent system that automatically

- detects significant events from large volumes of news stories,
- presents the main content of the events to the user as summaries with multiple levels of abstraction,
- alerts the user to the onset of novel events, and
- tracks events of interest based on user-given sample stories.

This is the goal of a new line of research, *topic detection and tracking*. (*Topic* in this context means dynamically changing events. Throughout this article, we use the term *event* instead, and use *topic* in its more conventional sense, as we'll discuss later.) TDT works with chronologically ordered news stories from multiple channels of TV and radio broadcasts or newswire sources. The input data can be the original or transcribed text, or it can be the output of automated speech recognition, which typically suffers approximately 25% to 50% word-recognition error.

TDT comprises three main subtasks:

1. Segmenting speech-recognized TV and radio broadcasts into news stories,
2. Detecting events from unsegmented or segmented news streams, and
3. Tracking stories for particular events based on user-identified sample stories.

1094-7167/99/$10.00 © 1999 IEEE

We've adapted several IR and machine-learning techniques for effective event detection[1] and tracking. This article discusses our research using manually segmented documents; Carnegie Mellon's research on automatic segmentation has been reported elsewhere.[2]

Event analysis

Before exploring the solution space, let's observe the properties of events in news stories, which will shed light on what makes event detection and tracking a challenge to traditional IR and machine-learning technology.

The TDT1 corpus, developed by the researchers in the TDT Pilot Research Project, is the first benchmark evaluation corpus for TDT research. (For more information on the TDT program, see the related sidebar. The TDT1 corpus is available through the Linguistic Data Consortium. Larger and richer corpora (TDT2 and TDT3) have been and continue to be developed by the LDC—see *www.ldc.upenn.edu/TDT*. In this article, we report our experiments on the TDT1 corpus only.) Table 1 shows the 25 events manually identified in this corpus. TDT1 consists of 15,863 chronologically ordered news stories spanning from 1 July, 1994, to 30 June, 1995. Roughly one-half of these stories are randomly sampled Reuters articles; the rest are CNN broadcasts that were manually transcribed by the Journal Graphics Institute.

Event identification consists of randomly sampling from the corpus, defining the events

Table 1. The manually identified events in the TDT1 corpus.

Event ID	Count	Start time	Name
1	8	94-02-22	*Aldrich Ames*
2	10	94-08-15	*Carlos the Jackal (his capture)*
3	34	94-12-25	*Carter in Bosnia*
4	14	94-09-12	*Cessna on White House*
5	41	94-12-30	*Clinic murders (Salvi)*
6	45	94-07-16	*Comet into Jupiter*
7	2	94-12-08	*Cuban riot in Panama*
8	58	94-07-08	*Death of Kim Jong Il (N. Korea)*
9	114	94-07-??	*DNA in OJ trial*
10	12	94-07-11	*Haiti ousts observers*
11	97	94-12-17	*Hall's copter (N. Korea)*
12	22	94-10-19	*Humble, TX, flooding*
13	8	94-07-12	*Justice-to-be Breyer*
14	2	94-01-06	*Kerrigan/Harding*
15	84	95-01-17	*Kobe Japan quake*
16	44	95-03-13	*Lost in Iraq*
17	24	94-12-21	*NYC subway bombing*
18	273	95-04-19	*OK City bombing*
19	4	94-11-22	*Pentium chip flaw*
20	12	94-11-29	*Quayle lung clot*
21	65	95-06-02	*Serbians down F-16*
22	91	94-11-11	*Serbs violate Bihac*
23	7	94-07-22	*Shannon Faulkner*
24	39	94-09-08	*USAir-427 crash*
25	22	95-01-09	*WTC bombing trial*

The TDT program

The U.S. Government initiated the Topic Detection and Tracking research in 1996 and has supported it since. Three research groups participated the TDT Pilot Research Project from 1996 to 1997, including Carnegie Mellon University, the University of Massachusetts at Amherst, and Dragon Systems. Many more research groups participate in the current TDT Project, Phase 2, including CMU, UMass, Dragon, the University of Pennsylvania, IBM, BBN, and SRI.

Some of the ongoing research in other TDT-member groups is directly related to our work, particularly approaches developed by UMass and Dragon.[1,2] UMass adapted its benchmark IR systems (InQuery and InRoute) to the TDT problems, using a combination of statistical phrase finding, part-of-speech tagging, TD-IDF term weighting, single-pass clustering, and a Rocchio classification method.[2] Dragon applied speech-recognition techniques, including unigram (and later bigram) language modeling for event representation and a *k*-means clustering method for document classification. Indirectly related work includes document-clustering methods applied to retrieval and corpus-navigation problems,[3–8] and supervised-learning algorithms applied to text categorization.[9] Those results provide a rich background to our research, but do not directly address the problems of event detection and event tracking in temporal text and audio streams.

References

1. J. Allan et al., "Topic Detection and Tracking Pilot Study: Final Report," *Proc. DARPA Broadcast News Transcription & Understanding Workshop*, Morgan Kaufmann, San Francisco, 1998, pp. 194–218.
2. J. Allan, R. Papka, and V. Lavrenko, "Online New Event Detection and Tracking," *Proc. SIGIR '98: 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1998, pp. 37–45.
3. C.J. van Rijsbergen, *Information Retrieval*, Butterworths, London, 1979.
4. E.M. Voorhees, "Implementing Agglomerative Hierarchic Clustering Algorithms for Use in Document Retrieval," *Information Processing & Management*, Vol. 22, No. 6, 1986, pp. 465–476.
5. R. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing and Management*, Vol. 25, No. 5, 1988, pp. 577–597.
6. R.H. Thompson and B.W. Croft, "Support for Browsing in an Intelligent Text Retrieval System," *Int'l J. Man-Machine Studies*, Vol. 30, No. 6, 1989, pp. 639–668.
7. D.R. Cutting et al., "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections," *Proc. SIGIR '92: 15th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1992, pp. 318–329.
8. O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Proc. SIGIR '98: 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1998, pp. 46–54.
9. Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," to be published in *J. Information Retrieval*, 1999.

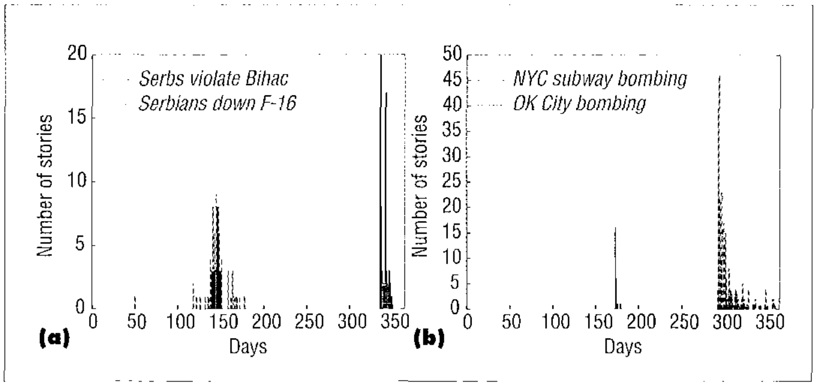

Figure 1. Histograms of events related to (a) *Serbian* and (b) *bombing,* from 1 July 1994 to 30 June 1995.

discussed in the sampled stories, and making exhaustive relevance judgments for each of those events. Each story received a label of Yes, No, or Brief with respect to each of the 25 events. Yes indicates that the article focuses on a particular event, while Brief indicates that the article mentions the event in passing but does not discuss it as a major focus. This process produced a subset of the events in the corpus. The random sampling made larger events (those reported more often) more likely to be included than smaller events.

An *event* as specified in the TDT problems is different from a *topic* in the conventional sense. An event identifies something (nontrivial) happening in a certain place at a certain time. For example, *USAir-427 crash* is an event but not a topic, and "airplane accidents" is a topic but not an event. Basically, events are instances of topics, associated with certain actions. In event detection and tracking, the system must make these distinctions automatically. Events are often associated with bursts (many occurrences in a short time period) of news stories. Figure 1 illustrates histograms of several events. Figure 2 shows the histograms of all 25 manually identified events in TDT1.

Several patterns emerged from our observations of temporal event distributions:

- News stories discussing the same event tend to be temporally proximate. This suggests using a combined measure of lexical similarity and temporal proximity as a criterion for document clustering.
- A time gap between bursts of topically similar stories often indicates different events (for example, different earthquakes, airplane accidents, or political crises). This suggests that monitoring cluster evolution over time is necessary and that using a time window to restrict the temporal extent of an event would be beneficial.
- A significant vocabulary shift and rapid changes in term frequency distribution are typical of stories reporting a new event. This indicates the importance of dynamically updating the corpus vocabulary and statistical term weights. Timely recognition of new patterns, including previously unseen proper names and proximity phrases, in the streams of stories is potentially useful for detecting a new event's onset.
- Events are typically reported in a relatively brief time window (for example, one to four weeks) and contain fewer reports than topics contain. So, we need learning methods that require only a few positive training examples to achieve satisfactory tracking performance, and that can exploit the temporal decay inherent in event reporting.

Event detection

Event detection is an unsupervised learning task, subdivided into two forms. *Retrospective detection* discovers previously unidentified events in a chronologically ordered accumulation of documents (stories). *Online detection* identifies the onset of new events from live news feeds in real time. Both forms intentionally lack advance knowledge of novel events but might have access to unlabelled historical news stories for use as contrast sets.

Methods. Given that each event usually involves multiple news stories, document clustering appears to be a natural approach to event discovery. We implemented two clustering methods: *GAC*, a divide-and-conquer version of a group-average clustering algorithm,[3] and *INCR*, a single-pass incremental clustering algorithm. GAC performs agglomerative clustering, producing hierarchically organized document clusters. It is designed for batch processing and has been used for retrospective detection. INCR produces a nonhierarchical partition of the input collection. It is designed for sequential processing and has been used for both retrospective and online detection.

Cluster representation. Our clustering algorithms are rooted in the conventional vector-space model and in traditional clustering techniques in IR.[4,5] Each document is represented by a vector of weighted terms that can be either words or phrases. For term weighting, we use a standard version (ltc) of the TF-IDF scheme:

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

where

- $w(t, d)$ is the weight of term t in document d;
- $tf(t, d)$ is the within-document *term frequency* (TF);
- $\log_2(N/n_t)$ is the *inverted document frequency* (IDF);
- N is the size of the training corpus used to compute the IDF;
- $n(t)$ is the number of training documents where t occurs; and
- $\|\mathbf{d}\| = \sqrt{\sum_t w(t,d)^2}$ is the 2-norm of vector **d**.

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

For cluster representation, we obtain a *prototype vector* (also called the cluster's *centroid*) by summing the vectors of the member documents and selecting the k most significant terms per prototype. Each document is treated as an initial cluster with a single member. To measure the distance between two clusters, we use the standard cosine similarity—that is, the cosine value of the two prototype vectors.

We modified standard TF-IDF term weighting to use adaptive IDF in addition to static IDF. Because new stories arrive continuously, how should we deal with the new

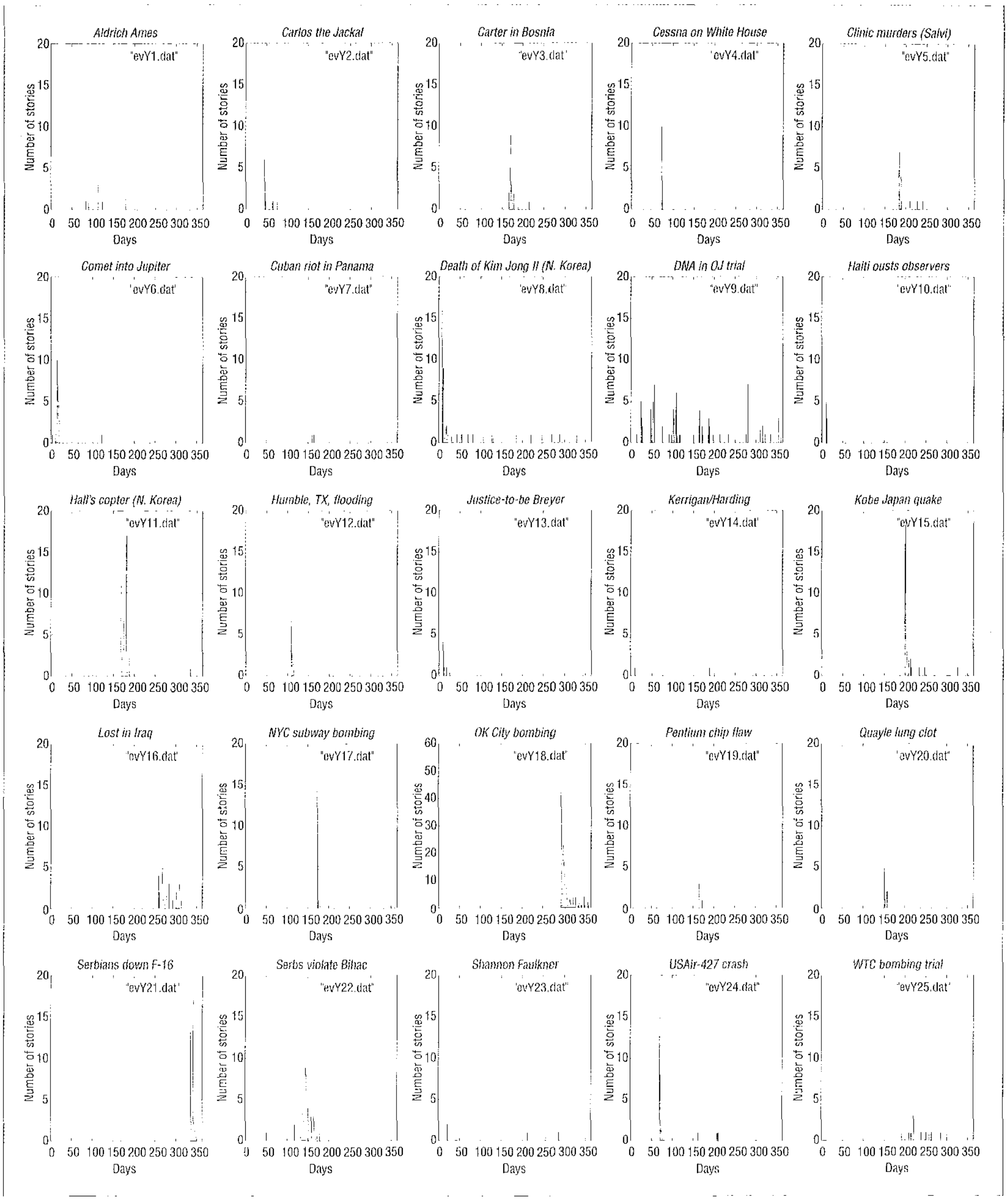

Figure 2. Histograms of the 25 TDT events.

vocabulary from incoming documents and update the corpus-level IDF statistics (which affect term weighting and vector normalization)? Related work shows that incremental IDF works effectively for document retrieval after a sufficient number of documents have already been processed.[6] For online event detection (and tracking), we use a retrospective corpus (for example, a six-month collection of CNN news stories before the TDT1 corpus) to compute the initial IDF values, and then incrementally update them with each incoming document. The incremental version of IDF is

$$IDF(t, p) = \log_2 (N(p)/n(t, p)),$$

where p is the current time point, $N(p)$ is the

$$sim(\mathbf{x},\mathbf{c})' = \begin{cases} sim(\mathbf{x},\mathbf{c}) & \text{if } c \text{ has any member document in the time window;} \\ 0 & \text{otherwise} \end{cases}$$

(a)

$$sim(\mathbf{x},\mathbf{c})' = \begin{cases} \left(1-\frac{i}{m}\right)\times sim(\mathbf{x},\mathbf{c}) & \text{if } c \text{ has any member document in the time wind} \\ 0 & \text{otherwise} \end{cases}$$

(b)

Figure 3. Determining the modified similarity between a document *x* and any cluster *c* in the past: (a) the basic formula; (b) the formula with an added linear decaying-weight function. *i* is the number of documents between *x* and the most recent member document in *c*.

number of accumulated documents up to the current point (including the retrospective corpus if used), and $n(t, p)$ is the document frequency of term *t* at time *p*. For retrospective detection, we use static IDF trained on the entire TDT corpus.

Group-average clustering. This approach maximizes the average similarity between document pairs in the resulting clusters by merging clusters in a greedy, bottom-up fashion.[3,7] Straightforward group-average clustering algorithms typically have a complexity in time and space quadratic to the number of input documents. So, they are less economical or tractable for large applications than are simpler methods such as single-link clustering or single-pass *k*-means clustering.

To address this problem, Doug Cutting and his colleagues developed *fractionation*, a divide-and-conquer strategy that compromises between cluster quality and computational efficiency.[5] This strategy grows clusters iteratively. In each iteration, it first divides the current pool of clusters into evenly sized *buckets*. Then it applies group-average clustering to each bucket locally, merging smaller clusters into larger ones. The fractionation algorithm has a time complexity of $O(mn)$, where n is the number of documents in the input corpus, m is the bucket size, and $m \leq n$.

The bucketing strategy is particularly well-suited for event detection. We found that bucketing stories based on the order in which they are reported increases not only computational efficiency but also cluster quality and detection effectiveness. In other words, this strategy gives a higher priority to grouping temporally proximate stories than to temporally disparate ones.

We implemented a modified version of this algorithm; we call our version "GAC" throughout the article. The input to GAC is a collection of documents sorted chronologically and a set of user-specified parameters. The output is a forest of binary trees of clusters. The algorithm consists of these steps:

1. Treat each document in the input collection as a singleton cluster, and set the initial partition to be the full set of the singleton clusters.
2. Divide the current partition into nonoverlapping and consecutive buckets of size m (a user-specified parameter).
3. Apply GAC to each bucket, which repeatedly combines the two closest lower-level clusters into a higher-level cluster. This occurs until the number of clusters in the bucket has decreased by a factor of ρ (a user-specified parameter) or until all the similarity scores between two clusters are below a preselected clustering threshold (another user-specified parameter).
4. Remove the bucket boundaries (assemble all the GAC clusters) while preserving the time order of the clusters. Use the resulting cluster series as the updated partition of the corpus.
5. Repeat Steps 2 to 4, until the partition is no larger than m or stops decreasing because of the minimum similarity constraint.
6. Periodically (once per k iterations in Step 5) recluster the stories within each of the top-level clusters, by flattening the component clusters and regrowing clusters internally from the leaf nodes.

The temporal bucketing and reclustering are our modifications to Cutting's algorithm. Reclustering is useful when events straddle the initial temporal-bucket boundaries or when the bucketing causes undesirable groupings of stories about different events. Reclustering reduces the initial bucketing's systematic bias but increases computation time.

Single-pass incremental clustering. The INCR algorithm is straightforward. It sequentially processes the input documents, one at a time, and grows clusters incrementally. A new document is absorbed by the most similar cluster in the past if the similarity between the document and the cluster is above a preselected *clustering threshold* (t_c). Otherwise, the document becomes the seed of a new cluster. By adjusting the threshold, we can obtain clusters at different levels of granularity. Choosing a suitable clustering threshold, therefore, is important for the effectiveness of retrospective event detection, where the granularity levels of document clusters should match the event concepts.

To apply INCR to online event detection, we introduced the *novelty threshold* (t_n). If the maximal similarity score between the current document and any cluster in the past are below this threshold, INCR labels this document New. That is, the document is the first story of a new event. Otherwise, INCR labels it Old. By tuning the novelty threshold, we can adjust online detection's sensitivity to novelty.

Both the clustering and novelty thresholds are user-specified parameters. The choice for one threshold is independent of the choice for the other. Using both thresholds permits better empirical optimization for different tasks. For instance, we found that setting $t_c = t_n$ (that is, t_n is not needed) is appropriate for retrospective clustering, but for online detection, choosing $t_c = \infty$ (that is, not growing any clusters) is better.

We also added a *time penalty*. The simplest way is to use a uniformly weighted time window. Given the current document x in the input stream to INCR, we impose a *time window* of m documents before x. We define the modified similarity between x and any cluster c in the past as shown in Figure 3a. Alternatively, we can introduce a linear decaying-weight function in the formula (see Figure 3b), where i is the number of documents between x and the most recent member document in c. The decaying-weight function more smoothly uses the temporal proximity, compared to using a uniformly weighted window. (For simplicity, we define a linear function only for the decay weighting. However, if necessary, we can easily generalize this definition to a more elaborate form, such as the interpolated decay profile extracted from an earlier development or training corpus.) These windowing strategies yielded measurable and consistent improvements in our event-detection experiments, enhancing precision with only a small sacrifice in recall, compared to not using a time penalty. Recall is the ratio of correct assignments (of event labels to documents) by the system divided by the total number of correct assignments by human judgments. Precision is the ratio of correct assignments by the system divided by the total number of the system's assignments.

In addition to the binary (New or Old) prediction, for each incoming document INCR computes a score indicating how new the document is. This score is

$$score(x) = 1 - \arg\max_{c}\left\{sim(x,c)'\right\},$$

where x is the currently new document and c is any cluster in the past. We use these scores to evaluate the potential trade-off between different types of errors. That is, by adjusting the threshold on these scores for binary decisions, we can obtain the trade-off between recall and precision or between miss and false alarm (for more details, see "Online-detection results" in the next section).

Evaluation. Table 2 shows a corpus summary obtained by applying GAC to a few thousand news stories (CNN news and Reuters articles from January to February 1995) and by presenting a few top-ranking terms for each cluster. As the table shows, domestic politics reigned supreme as usual, the O.J. Simpson trial received media attention in early 1995, and so on. However, the table also reveals that disasters struck Kobe, Japan, and Malibu, California, and unrest in Chechnya flared up again—events that were not present the months before. The key terms provide content information, and the story counts imply significance, as measured by media attention.

New multidocument summarization methods[8] applied to the clusters provide additional information as to the nature of the events. And, if we desire further detail, we can examine the clusters, subclusters, and individual documents through query-driven retrieval. The utility of summarization and cluster-based browsing tools is evident from our prototypes, even though some clusters might be imperfect and the current user interface is rudimentary.

Figure 4 shows the temporal distributions of two events. Each graph's upper half is the histogram of human-labeled documents for an event; its lower half is the histogram of the system-generated cluster for the same event. The absolute value on the y-axis is the story count for the event or cluster in a particular day. If an event and a cluster are a perfect match, their histograms will perfectly mirror each other.

As the figure shows, GAC and INCR have complementary strengths and weaknesses. GAC shows almost symmetric graphs for most events, except those with significant temporal extent, so it is particularly suitable for recognizing news bursts. INCR, on the other hand, has less symmetric performance but is better at tracking long-lasting events (for example, *DNA in OJ trial* and *OK City bombing*). The observed behavior might come partly from the different biases in these algorithms and partly from the parameter settings in the particular experiments.

Retrospective-detection results. The 1998 TDT1 evaluation was the first controlled study[9] where comparative results are available. Therefore, we used it as a reference for our results here. We used the entire TDT1 corpus as the test set for evaluating detection systems, although we'd prefer to have an additional cross-validation corpus available for setting global system parameters. However, detection is an unsupervised classification task that does not involve labeled training data. So, there was no contamination of the test data in that sense, except possibly with respect to setting a handful of system parameters.

(Subsequent research in progress on an independently developed TDT2 corpus indicates that the parameter values can be effectively chosen using a retrospective corpus and cross-validation. For example, we found that the clustering threshold optimal on the TDT1 corpus for online event detection is nearly optimal on the TDT2 corpus.)

Each detection system ran on the entire TDT1 corpus, producing system-generated clusters that are either a partition of the corpus (that is, no overlapping stories between clusters) or a forest of hierarchies (overlap-

Table 2. Clusters generated by group-average clustering on the TDT1 corpus, Jan.–Feb. 1995 subset.

Documents included	Top-ranking words (stemmed)
330	republ clinton congress hous amend
217	simpson o prosecut trial jury
98	israel palestin gaza peac arafat
97	japan kobe earthquak quak toky
93	russian chech chechny grozn yeltsin
56	somal u mogadishu iraq marin
55	flood rain californ malibu rive
48	serb bosnian bosnia croat u
35	game leagu play basebal season
33	crash airlin flight airport passeng
28	clinic sav abort massachuset norfolk
27	shuttl spac astronaut mir discov
26	patient drug virus holtz infect
24	chin beij deng trad copyright
...	

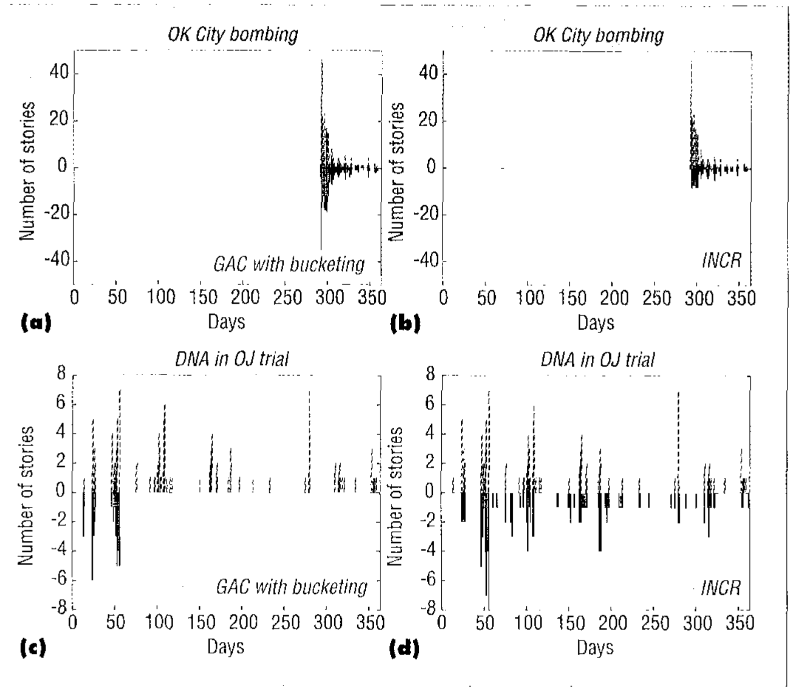

Figure 4. Temporal distributions for group-average clustering and incremental clustering for two events (from 1 July 1994 to 30 June 1995): (a) *OK City bombing,* GAC; (b) *OK City bombing,* INCR; (c) *DNA in OJ trial,* GAC; (d) *DNA in OJ trial,* INCR.

Table 3. Per-event contingency table.

	In event	Not in event
In cluster	a	b
Not in cluster	c	d

Table 4. Retrospective-detection results.

	Partition required			Cluster overlap allowed	
	CMU (INCR)	UMass (no dupl)	Dragon (multipass)	CMU (GAC)	UMass (dupl)
Microaverage					
Recall (%)	62	34	61	75	73
Precision (%)	82	53	69	90	78
Miss (%)	38	66	39	25	27
False alarm (%)	0.04	0.09	0.08	0.02	0.06
F_1	0.71	0.42	0.65	0.82	0.75
Macroaverage F_1	0.79	0.60	0.75	0.84	0.81

ping stories between clusters are allowed). We evaluated each system using the 25 clusters that best matched the 25 manually labeled events (consisting of 1,131 stories—about 7% of the total stories).

(The TDT1 corpus includes more than the 25 labeled events, and each detection system generates more than 25 clusters. However, the evaluation did not measure the match between clusters and events beyond the 25 cluster-event pairs. That is, the system detected many potential events, but we only evaluated it on the subset of the system-generated clusters that best matched the manually labeled events. The correspondence of many clusters to other potential events suggests that a detection system could provide browsing support to the user for navigation through the event space. Although we expect hierarchical clustering to be a suitable choice for navigation support, determining how to evaluate the practical impact of various kinds of navigation support requires future research.)

To evaluate the goodness of matching between each cluster/labeled-event pair, we used the contingency table in Table 3. With that table, we define these performance measures:

- Miss: $m = c/(a + c)$ if $a + c > 0$, otherwise undefined.
- False alarm: $f = b/(b + d)$ if $b + d > 0$, otherwise undefined.
- Recall: $r = a/(a + c)$ if $a + c > 0$, otherwise undefined.
- Precision: $p = a/(a + b)$ if $a + b > 0$, otherwise undefined.
- $F_1 = 2rp/(r + p) = 2a/(2a + b + c)$ if $(a + b + c) > 0$, otherwise undefined.[7]

F_1, originally defined by Keith van Rijsbergen, is the harmonic mean of recall and precision.[7]

To measure global performance, we use two averaging methods. We obtain the *microaverage* by merging the contingency tables of the 25 events (by summing the corresponding cells) and then using the merged table to produce global performance measures. We obtain the *macroaverage* by first producing per-event performance measures, then averaging the corresponding measures. The former measures introduce a scoring bias toward frequently reported events, and the latter toward less reported ones. So, we use both to minimize the effect of hidden bias.

The GAC parameter settings were

- Bucket size = 400
- Clustering threshold = 0.2
- Terms per vector = 100
- Term weighting = ltc
- Reduction factor $\rho = 0.5$
- Number of iterations between reclustering = 5

The INCR parameter settings were

- Window size = 2,000
- Clustering threshold = 0.23
- Terms per document vector = 125
- Term weighting = ltc

Table 4 summarizes the retrospective-detection results. For comparison, we include the results for approaches developed at the University of Massachusetts (temporal-TF-based event detection and agglomerative clustering) and by Dragon Systems (multipass k-means clustering).[9] (See the sidebar "The TDT program" for more on the other participants.) Algorithms that permit cluster hierarchies (GAC) or potentially overlapping clusters (dupl) performed better than nonhierarchical algorithms that adhere to the strict partition requirement.

For the partition-producing algorithms, we were surprised that the simplest approach—INCR's single-pass clustering—worked as well as Dragon's multipass k-means clustering. This might be partly because of the temporal proximity of events, which simplifies the clustering problem. Time windowing was highly effective for INCR. With other parameters fixed, a time window of 2,000 documents (covering about 1.5 months) increased the F_1 performance score from 0.64 (with no time window) to 0.70.

The better results obtained by hierarchical clustering (GAC) or overlapping clustering (dupl) are less surprising. We believe the main reason for GAC's better results is the multileveled clusters, which enable the detection of events at any degree of granularity. GAC achieves this representational power at the cost of producing more clusters (approximately 12,000 in this particular run) than the INCR partition produces (5,907). The increase in the number of clusters might not add a significant burden to the end user in scatter-gather navigation or query-driven retrieval,[5] where only a small subset of the clusters would actually be visited by the user via selected paths on the hierarchy.

Online-detection results. To evaluate online-detection performance, we used the contingency table in Table 5. Because only 25 events are defined, and each event has only one first story, the total number of true New stories is 25 for the entire corpus. This number is too small for a statistically reliable estimation of performance. To improve the reliability, we conducted an 11-pass evaluation. The first pass used the entire corpus; the second used the modified corpus after removing ("skipping") the first story of each event; the third used the modified corpus after removing the first two stories of each event, and so on. The eleven passes are labeled as $N_{skip} = 0, 1, \ldots, 10$. We computed a contingency table for each value of N_{skip} and then obtained a global contingency table by summing the corresponding cells in the per-N_{skip} contingency tables. We derived performance scores from those contingency tables the same way we did for retrospective-detection evaluation.

The parameters used in online INCR were

- Window size = 2,500 linear decay
- Clustering threshold = ∞
- Novelty threshold = 0.16
- Terms per document vector = no limit
- Term weighting = ltc
- IDF = static from retrospective corpus plus adaptive

Table 6 summarizes the official TDT1 results, including those for UMass's and Dragon's approaches. Both Carnegie Mellon University (INCR) and UMass conducted multiple runs with different parameter settings in the TDT workshop;[9] the table shows the best result for each site for F_1. A possible reason why the CMU and UMass approaches have significantly higher F_1 scores than that of Dragon is that they both used individual documents to represent the past in online detection; Dragon's approach grew clusters instead. (UMass used a single-pass algorithm to compare a new document with all the past documents, which is similar to our INCR. Dragon used a single-pass version of its *k*-means clustering method for online detection. Multipass clustering cannot be used for online detection because, by the task definition, future knowledge is not available at the decision-making point.) Keeping individual documents without clustering them makes passing the *novelty test* (that is, scoring at or above the novelty threshold) more difficult for the current story. This is because the story must be sufficiently different from all past stories, a stronger condition compared to being different from an average of past stories.

In addition to the scores in Table 6, we also used a *detection-error trade-off curve*[10] to evaluate each online-detection system in the TDT evaluation. A DET curve is the sequence of interpolated values in the false-alarm/miss space. It is obtained by retrospectively thresholding on the system-generated scores for individual documents. Any document with a score above a particular threshold is labeled New or Old. This process produces a set of performance scores such as those in Table 6 for each threshold value. By changing the threshold value and interpolating the corresponding miss and false-alarm values, we can observe the trade-off between them (see Figure 5). Each DET curve axis (miss and false alarm) is scaled to a Gaussian (thereby compressing the midrange and expanding the extremes) such that a "random"-decision plot is a straight line passing through the 50%-50% error point.

As an alternative to DET curves, we can plot the corresponding recall and precision (see Figure 6). These recall-precision curves show that the CMU approach performs better at the high-precision area. As is especially evident in Figure 6, the CMU, UMass, and Dragon approaches behave very differently, inviting further detailed investigation. The recall-precision curves are, in IR terminology,[11] *noninterpolated* and exhibit typical nonmonotonic behavior.[4]

Table 5. Online-detection contingency table.

	New is true	Old is true
Predicted New	*a*	*b*
Predicted Old	*c*	*d*

Table 6. Online event-detection results for micro-averaging. The UMass approach used a single-pass algorithm without clustering, and Dragon's approach used a single-pass version of its *k*-means clustering.

	CMU (INCR)	UMass	Dragon
Recall (%)	50	49	42
Precision (%)	37	45	21
Miss (%)	50	51	58
False alarm (%)	1.89	1.31	3.47
F_1	0.42	0.47	0.28

Event-tracking

Event tracking is a supervised-learning task. It aims to automatically assign event labels to news stories when they arrive, based on a small number of previously identified past stories that define the event. Adaptive learning is needed because of the dynamic nature of events; that is, they start at certain time points and eventually trail off. Making fine distinctions between topically related events is another task-specific requirement; for example, *NYC subway bombing* and *OK City bombing* should be identified as different events. Moreover, quick learning is highly desirable. This means that the classifier should need only a few positive training examples per event to achieve satisfactory tracking performance, as we mentioned before.

Methods. We found that two well-known learning methods—*k-nearest neighbor classification* and *decision-tree induction-* -are well-suited to this task, after some extensions to the standard algorithms. The *k*NN algorithm uses the same document representation used for event detection—that is, a document is represented as a bag of terms with statistical weights. The decision-tree algorithm, on the

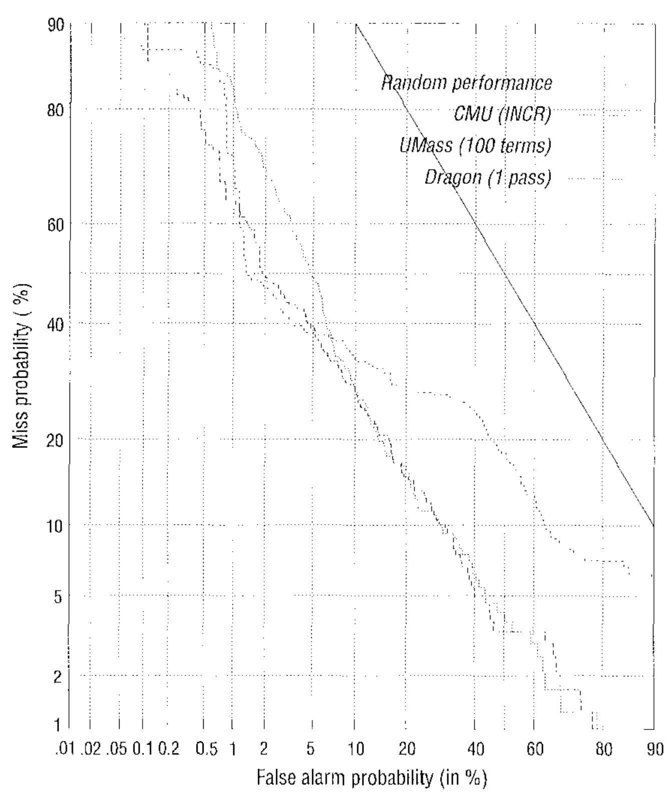

Figure 5. Online-detection Detection-Error Trade-off curves.

other hand, uses binary term weighting only (1 for terms present and 0 for terms absent).

k-*nearest neighbor classification.* *k*NN is an instance-based classification method well-known in pattern recognition and machine learning. It has been applied to text categorization since the early stages of TC research.[12,13] We chose *k*NN for event tracking because, in addition to its generally good performance, it makes the fewest assumptions about terms, stories, and optimal decision surfaces for tracking, compared to other methods (see the sidebar, "Text categorization approaches").

Official TDT evaluations require that each event be tracked independently, without any knowledge about other events. That is, for each particular event, the training stories are labeled either Yes, No, or Brief. According to this constraint, we adapted our conventional *M*-ary-classification *k*NN (developed for text categorization in general)[13] to the binary-classification problem of event tracking.

We trained a specific *k*NN classifier for each event. As an input story arrives, the system converts it into a vector, compares it to the training stories (see the section "Event Tracking Results" for the training-set construction), and selects the *k* nearest neighbors based on the cosine similarity between the input story and the training stories. The system computes the confidence score for a Yes prediction on the input story by summing the similarity scores for the positive and the negative stories in the *k*-neighborhood, and taking the difference between the two sums:

$$s1\left(YES|\mathbf{x}\right)=$$
$$\sum\nolimits_{\mathbf{d}\in P(x,k)}\cos(\mathbf{d},\mathbf{x})-\sum\nolimits_{\mathbf{d}\in N(x,k)}\cos(\mathbf{d},\mathbf{x}) ,$$

where $\mathbf{x}$ is the input story, $P(x, k)$ is the set of positive training stories in the k-neighborhood, and $N(x, k)$ is the negative training stories in the k-neighborhood.

We obtain binary decisions by thresholding locally on the confidence scores generated by each event-specific classifier. Our experiments showed good results (see Table 7) for *k*NN when the threshold was at the zero value of the confidence score. However, when moving the threshold beyond that point, we found that it resulted in a somewhat unsatisfactory DET curve. More specifically, it has difficulty gaining a high recall without sacrificing precision significantly. The reason, we believe, is that the positive examples are extremely sparse (for most events) in the training set and are therefore often "blocked away" by densely populated negative examples. One solution to this problem is to discount the influence of negative examples by sampling a small portion in the *k*-neighborhood and ignoring the remaining negative examples.

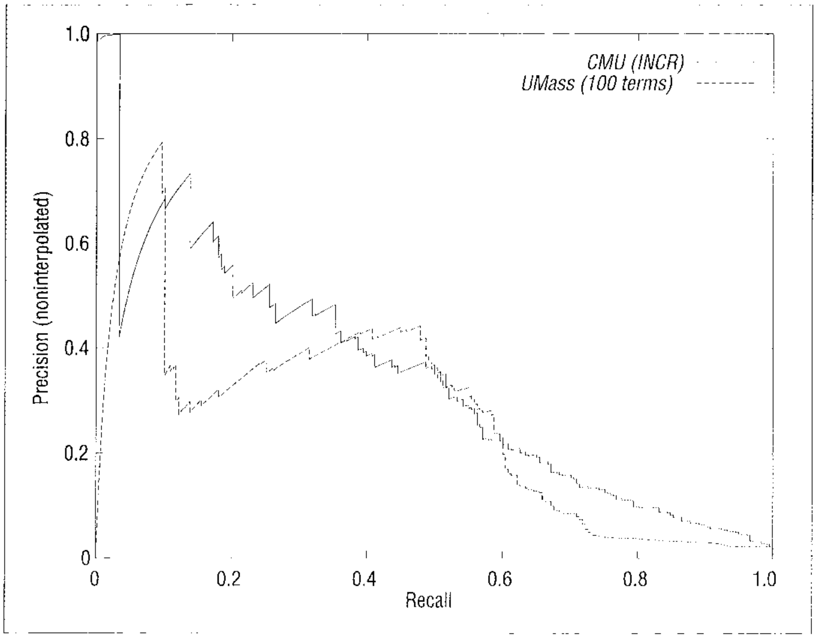

Figure 6. Online-detection recall-precision curves.

This idea leads to a modified version of *k*NN; we call the original version *k*NN-a and the modified version *k*NN-b. In the modified version, we take the $k1$ ($\leq k$) nearest positive examples ($P(x, k1)$) and $k2$ ($\leq k$) nearest negative examples ($N(x, k2)$) from the k-neighborhood, and average the similarity scores of the two subsets. The confidence score for a Yes prediction on the input story is

$$s2\left(YES|\mathbf{x}\right)=$$
$$\frac{1}{k1}\sum_{\mathbf{d}\in P(x,k1)}\cos(\mathbf{d},\mathbf{x})-\frac{1}{k2}\sum_{\mathbf{d}\in N(x,k2)}\cos(\mathbf{d},\mathbf{x}) .$$

By adding the parameters $k1$ and $k2$ and by suitably choosing the parameter values, we can effectively adjust the tracking system's DET behavior. In principle, these parameters can be empirically tuned based on the optimization of event tracking on a validation collection of stories. In reality, when only a very small number of positive examples are identified for an event, we would not want to use these positive examples for validation instead of training. The official event-tracking evaluation restricted the number (N_t) of positive training examples per event to 1, 2, 4, 8, and 16 (see "Event Tracking Results"). Under such a condition, we used this rule of thumb to determine the parameter values:

- For *k*NN-a, $k = \min\{N_t, 5\}$.
- For *k*NN-b, $k1 = \min\{P(x, 100), N_t\}$; $k2 = \min\{N(x, 100), 16\}$.

Another heuristic we used in event tracking is a time window. That is, any test story that is *k* stories away from the last positive training example is labeled No. We set the window size from 1,800 to 2,000 stories (about 1.5 months of data). We based this on the commonsense conclusion that most events last no longer than one or two months, and on the observation that a 1.5-month window is close to optimal for online detection.

Decision-tree induction. Decision trees are classifiers based on the principle of a sequential greedy algorithm that at each step strives to maximally reduce system entropy.[14] Their construction follows these steps:

1. Select the feature with maximal information gain as the root node.
2. Divide the training data according to the value of this feature. The partitioned subsets thus created form the branches.
3. For each branch, find the feature that

maximizes information gain over the training instances on that branch.

4. Repeat Steps 2 and 3 recursively.

We chose d-trees as an alternative to *k*NN for TDT tracking because they represent a very different technology, one with relatively reasonable performance in text-categorization evaluations on the benchmark Reuters collection. One potential disadvantage is that, unlike *k*NN, d-trees cannot generate a continuously varying trade-off between miss and false alarm or between recall and precision.

We developed our own d-tree method rather than using the standard C4.5 algorithm.[14] We wanted a version that is fast, scalable, and easily tunable for text categorization, although not necessarily optimized for other machine-learning tasks. We also did not want extra features such as C4.5's rules-from-trees option. Training 25 decision trees (one per event tracked), each with up to 15,000 stories, requires less than two minutes on a standard 300-MHz Sun Ultra II. We use the same information-gain metric (minimizing total entropy) and greedy root-to-leaves d-tree construction as in C4.5.

The primary tunable parameters are

- The minimal number of training instances at a leaf node,
- The percentage of positive instances at a leaf node,
- Whether or not to use word roots or stems instead of surface forms,
- Whether or not to distinguish between single and multiple occurrences of a word in a document,
- The size of the time window for training data (fixed or adaptive),
- The limit on positive or negative training examples, and
- The limiting of features to the top *N* (for example, top 1,000) by global information gain.

These parameters are tuned by cross-validation. Some parameters, such as the time window, make a significant difference. We optimize performance by using only the most recent 1.5 to 2 months' training data. Other parameters such as stemming (using stemmed words as features—for example, "bomb" instead of "bombing") make only a small difference in overall performance.

Results. We evaluated the event-tracking systems using N_t positive training examples plus all available negative examples, where N_t is 1, 2, 4, 8, and 16. For each event and particular N_t value, we split the TDT1 corpus at the point right after the N_tth positive example of that event. We used the stories before that point for training, and the remaining stories for testing. Fifteen of the 25 events have more than 16 Yes stories; we used those for event-tracking evaluation. (The stories judged as Brief were used for training but excluded from testing.) We tested each system on all the pairs (15×5) of training/test sets, resulting in 75 two-by-two contingency tables (for the predicted Yes or No versus the true Yes or No for an event). The microaveraged and macroaveraged performance scores were computed from the 75 contingency tables. Table 7 shows the results.

To illustrate the learning behavior of *k*NN-a and d-trees with respect to the number of positive training examples, Figure 7 presents the interpolated curves for microaverage and macroaverage F_1. Both methods work reasonably well with the small N_t values. d-trees are not as good as *k*NN when $N_t = 1$ or 2; also, their curve asymptotes at $N_t = 8$. Our interpretation of this behavior is that d-trees select only a few "good" features, but they (over)generalize quickly. This proves to be problematic when the input data is noisy, as evidenced by the speech-recognition-result discussed below. On the other hand, *k*NN is based on the local training examples surrounding a test story but uses all the terms in those stories as features.

To investigate the trade-off potential, we evaluated the false-alarm and miss rates of *k*NN-a and *k*NN-b when varying the decision thresholds on their confidence scores. We applied decision thresholding locally within each event-specific *k*NN in the experiment with a fixed value of N_t. This produced 75 local DET curves per system for the 15 events and five N_t values in total. To observe the trade-off in average, we divided the false-alarm range into 5,000 evenly sized intervals. We then computed the average miss rate in each interval on a per-event and per-N_t basis and averaged these averages over the events. Finally, we interpolated the resulting points in all the intervals.

Figure 8 shows the average DET curves of *k*NN-a, *k*NN-b, and d-trees for $N_t = 8$. The comparison indicates that *k*NN-a performs best for high-precision (or low false-alarm) event tracking, while *k*NN-b is best for high-recall-oriented applications. This comparison is under the (crucial) condition of very small N_t and is not necessarily generalizable

*k*NN and other text-categorization approaches

k-nearest neighbor classification has been used as a baseline in recent TC comparative research on the benchmark Reuters corpus of newswire stories, where the top-performing methods include *k*NN and the Linear Least Squares Fit mapping by Yiming Yang,[1] Generalized Instance Sets by Wai Lam and C.Y. Ho,[2] decision trees with boosting by Sholom M. Weiss and his colleagues,[3] Support Vector Machines by Thorsten Joachims and Susan Dumais,[4,5] and neural networks by Eric Wiener and his colleagues.[6] Other methods that performed less well in TC include Naive Bayes classifiers, decision trees without boosting, and rule-induction algorithms.[1,7]

References

1. Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," to be published in *J. Information Retrieval*, Vol. 1, 1999, pp. 69–90.
2. W. Lam and C.Y. Ho, "Using a Generalized Instance Set for Automatic Text Categorization," *Proc. SIGIR '98: 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1998, pp. 81–89.
3. S.M. Weiss et al., "Maximizing Text-Mining Performance," *IEEE Intelligent Systems*, Vol. 14., No. 4, July/Aug. 1999, pp. 63–69.
4. T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. ECML '98: European Conf. Machine Learning*, Springer-Verlag, Berlin, 1998, pp. 137–142.
5. S.T. Dumais, "Using SVMs for Text Categorization," *IEEE Intelligent Systems*, Vol. 13, No. 4, July/Aug. 1998, pp. 21–23.
6. E. Wiener, J.O. Pedersen, and A.S. Weigend, "A Neural Network Approach to Topic Spotting," *Proc. SDAIR '95: Fourth Ann. Symp. Document Analysis and Information Retrieval*, Information Science Research Inst., Univ. of Nevada, Las Vegas, Nev., 1995, pp. 317–322.
7. Y. Yang and X. Liu, "A Reexamination of Text Categorization Methods," *Proc. SIGIR '99: 22nd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, to be published by ACM Press, Menlo Park, Calif., 1999.

Table 7. Event-tracking results: performance averaged over all events, N_t = 1, 2, 4, 8 and 16.

Classifier	*k*NN-a	d-tree	UMass (RF-10T)	Dragon
Microaverage				
Recall (%)	89	80	64	65
Precision (%)	44	50	51	30
Miss (%)	56	50	36	70
False Alarm (%)	0.04	0.08	0.39	0.10
F_1	0.59	0.61	0.57	0.41
Macroaverage F_1	0.62	0.53	0.63	0.42

to different conditions. In other words, our focus here is to evaluate our systems under a task-specific constraint, and we found that *k*NN-b effectively smoothes the detection error trade-off in event tracking better than *k*NN-a or d-trees.

EVENT DETECTION AND TRACKing represent a new family of tasks for information retrieval and machine learning. We studied a set of retrieval techniques and learning algorithms addressing these challenges:

- Analyzing the nature of events in news stories,
- Identifying suitable learning algorithms for event detection and tracking,
- Suggesting special-purpose changes to standard learning algorithms, and
- Evaluating the suggested techniques and comparing the results to those by other research groups using different techniques.

Our empirical evaluations suggest these points:

- For retrospective detection, conventional document representation and relatively simple clustering algorithms (GAC and INCR) can be highly effective, especially when they are adapted to use both context similarity and temporal proximity in document clustering.
- Online novel-event detection is somewhat more difficult than retrospective detection. Nonclustering approaches appear to have better detection accuracy than clustering. However, this requires further investigation.
- For event tracking, both *k*NN and d-trees exhibit encouraging performance in quick learning, with their performance curve approaching a plateau after a very small number (4 or 8) of positive training examples.
- In event tracking, the detection-error tradeoff flexibility of *k*NN-b increases significantly with a suitable nearest-neighbor sampling and score normalization.

Important research questions for further investigation include,

- Are there better learning algorithms for the TDT problems?
- How can we model event evolution over time more accurately than through time windowing or simple linear decay?
- How can we combine document clustering and text summarization for user support in event detection and tracking?

TDT research and evaluations have not thoroughly addressed how to optimally use system-generated clusters and how to best match these clusters to the "true events." In principle, finding a meaningful mapping without any information (for example, positive and negative examples given an event) or knowledge about the target events (for example, event descriptions) is unrealistic. Document clustering can be useful only if event identification takes user input (or interaction) into account. Furthermore, users should receive suggested browsing strategies along with the system-generated clusters. Investigating potential strategies for traversing through cluster hierarchies or corpus partitions, and measuring their practical impact for end users in terms of time saving and

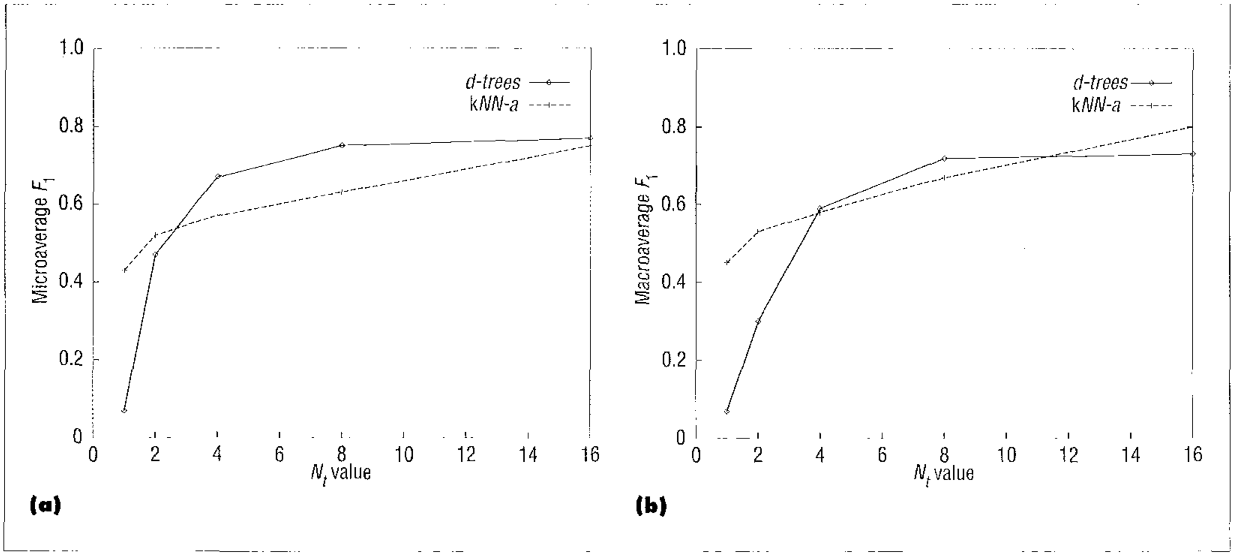

Figure 7. Learning curves of event-tracking systems: (a) microaverage; (b) macroaverage. N_t is the number of positive training examples.

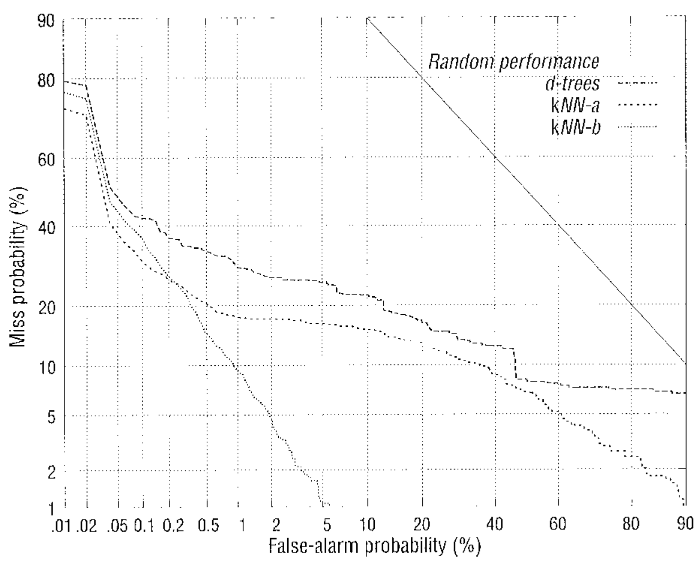

Figure 8. Detection Error Trade-off curves of *k*NN and d-trees.

error reduction, will be a crucial part of future research in event detection and tracking. ■

Acknowledgments

We thank Charles Wayne and George Doddington for their guidance in TDT task definition and evaluation, and James Allan at the University of Massachusetts and Jon Yamron at Dragon for sharing ideas and results in the research. The Department of Defense sponsored the TDT research in part. Any opinions or conclusions in this article are the authors' and do not necessarily reflect those of the sponsors.

References

1. Y. Yang, T. Pierce, and J. Carbonell, "A Study on Retrospective and Online Event Detection," *Proc. SIGIR '98: 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1998, pp. 28–36.

2. D. Beeferman, A. Berger, and J. Lafferty, "Statistical Models for Text Segmentation," *Machine Learning*, Vol. 34, 1999, pp. 1–34.

3. R. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing and Management*, Vol. 25, No. 5, 1988, pp. 577–597.

4. G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading, Mass., 1989.

5. D.R. Cutting et al., "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections," *Proc. SIGIR '92: 15th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1992, pp. 318–329.

6. J. Callan, "Document Filtering with Inference Networks," *Proc. 19th Ann. ACM/SIGIR Conf.*, ACM Press, New York, 1996, pp. 262–269.

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

8. J.G. Carbonell and J. Goldstein, "The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries," *Proc. SIGIR '98: 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1998, pp. 335–336.

9. J. Allan et al., Topic Detection and Tracking Pilot Study: Final Report, *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Morgan Kaufmann, San Francisco, 1998, pp. 194–218.

10. A. Martin et al., "The DET Curve in Assessment of Detection Task Performance," *Proc. EuroSpeech 1997, Vol. 4*, 1997.

11. D.K. Harman, ed., *The Second Text Retrieval Conf. (TREC-2)*, US Government Printing Office, Washington D.C., 1994.

12. B. Masand, G. Linoff, and D. Waltz, "Classifying News Stories Using Memory Based Reasoning," *Proc. SIGIR '92: 15th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, 1992, ACM Press, New York, pp. 59–64.

13. Y. Yang, "Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval," *Proc. SIGIR '94: 17th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, New York, 1994, pp. 13–22.

14. J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81–106.

Yiming Yang's biography appears on p. 31.

Jaime G. Carbonell is the Allen Newell Professor of Computer Science at Carnegie Mellon University and the director of the Language Technologies Institute. His research interests span several areas of artificial intelligence, including machine learning, machine translation, information retrieval, and automated text summarization. He received his PhD in computer science from Yale University. Contact him at the Language Technologies Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3072; jgc@cs.cmu.edu.

Ralf D. Brown is a systems scientist at Carnegie Mellon University's Language Technologies Institute, where he performs research in machine translation and information retrieval. He received his PhD in computer science from Carnegie Mellon. Contact him at the Language Technologies Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3072; ralf@cs.cmu.edu.

Thomas Pierce is a research programmer and master's student at Carnegie Mellon University's Language Technologies Institute, where he conducts research in information retrieval and topic detection. Contact him at the Language Technologies Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3072; tomp@cs.cmu.edu.

Brian T. Archibald was an undergraduate student at Carnegie Mellon University's Computer Science Department. He participated in the research in event tracking and text categorization at the Language Technologies Institute. He received his BS in computer science from Carnegie Mellon.

Xin Liu is a PhD student at Carnegie Mellon University's Language Technologies Institute, where he performs research in information retrieval and text categorization. He received his MS in computer science from Carnegie Mellon. Contact him at the Language Technologies Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3072; xliu@cs.cmu.edu.