A Study on Retrospective and On-Line Event Detection

Yiming Yang, Tom Pierce, Jaime Carbonell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3702, USA
www.cs.cmu.edu/~yiming/

Abstract This paper investigates the use and extension of text retrieval and clustering techniques for *event detection*. The task is to automatically detect novel events from a temporally-ordered stream of news stories, either retrospectively or as the stories arrive. We applied hierarchical and non-hierarchical document clustering algorithms to a corpus of 15,836 stories, focusing on the exploitation of both content and temporal information. We found the resulting cluster hierarchies highly informative for retrospective detection of previously unidentified events, effectively supporting both query-free and query-driven retrieval. We also found that temporal distribution patterns of document clusters provide useful information for improvement in both retrospective detection and on-line detection of novel events. In an evaluation using manually labelled events to judge the system-detected events, we obtained a result of 82% in the F_1 measure for retrospective detection, and a F_1 value of 42% for on-line detection.

Table 1. Corpus summary using keywords of automatically generated clusters of news stories

Size*	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
...	

* Size means the number of documents included.

1 Introduction

The rapidly-growing amount of electronically available information threatens to overwhelm human attention, raising new challenges for information retrieval technology. Although traditional query-driven retrieval is useful for content-focused queries, it is deficient for generic queries such as "What happened?" or "What's new?". Browsing without guidance or a conceptual structure of the search space is useful only in miniscule information spaces.

Consider a person who returns 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, and generating specific queries about unknown facts is rather unrealistic. Thus, intelligent assistance from the computer is clearly desirable. Such assistance could take the form of a content summary of a corpus for a quick review, the temporal evolution of past events of interest, or a listing of automatically detected new events which demonstrate a significant content shift from any previously known events. It would also be useful to have structured guidelines for navigation through document clusters. Table 1 shows a sample summary of a corpus obtained by applying our hierarchical content-based clustering algorithm to a few thousand news stories (CNN news and Reuters articles from January to February in 1995) and presenting each cluster using a few (statistically significant) key terms. As the table shows, domestic politics reigns supreme as usual, the OJ trial still receives media attention, etc. However, the table also reveals that disasters struck Kobe Japan and Malibu California, and Chechnia has flared up again, events which were not present the month before. The key terms provide content information, and the story counts imply significance, as measured by media attention. If further detail is desired, the sub-clusters can be examined via query-driven retrieval, browsing individual documents or synthetic summaries across documents [2]. The utility of such computer assistance is evident even though some clusters may be imperfect and the current user interface is rudimentary.

This paper reports our work in *event detection*, a new research topic initiated by the Topic Detection and Tracking (TDT) project[1]. The objective is to identify stories in several continuous news streams that pertain to new or previously unidentified events. To be more precise, detection consists of two tasks: *retrospective detection* and *on-line detection*. The former entails the discovery of previously unidentified events in an accumulated collection, and the latter strives to identify the onset of new events from live news feeds in real-time. Both

[1]The TDT project is supported by the U.S. Government, consisting of *segmentation* of stories in a continuous news-stream, temporal *event tracking* and *event detection*. Our event tracking work will be reported in a separate paper.

forms of detection intentionally lack advance knowledge of novel events, but do have access to (unlabelled) historical news-stories for use as contrast sets.

Event detection is essentially a discovery problem, i.e., *mining the data stream* for new patterns in document content. Bottom-up document clustering appears to be a natural solution for the discovery of *natural clusters* without introducing any assumptions about the domain or down-stream applications. Moreover, bottom-up clustering can result in a cluster hierarchy, thus allowing observation at any level of abstraction in the information space. Higher levels of clusters give progressively coarse-grain overviews of the content of document groups, while lower levels provide tighter clusters corresponding to specific events, temporal phases of events, or sub-events. We have applied both hierarchical and incremental non-hierarchical clustering algorithms to explore the nature of the problem and the solution space, focusing on the combined use of context information and temporal patterns of event distribution.

Directly related to our work is the on-going research in the other TDT-member groups: the UMass information retrieval group and the Dragon Systems speech recognition group. These groups also use document clustering as their basic approach. UMass focuses on the detection of "disaster" events by monitoring sudden changes in term frequencies in news streams, and using the stories that contain disaster-related terms to construct cluster centroids. Dragon adapts unigram (and later bigram) language models to document/cluster representation, and uses k-means clustering algorithms for document grouping[10]. We compare the results of the approaches of these two groups with the results of our approaches in the evaluation section.

Other related work in the IR literature includes:

- the *Scatter/Gather* cluster-based approach to corpus navigation[3, 4];
- the studies on clustering algorithms and their applications in the context of query-driven retrieval [7, 6, 9, 8].

Our detection methods are inspired by the *Scatter and Gather* paper[3], including the choice of the basic group-average clustering (GAC) algorithm. However clustering algorithms *per se* are not the major focus of this study, nor are the applications or evaluations in a query-driven retrieval paradigm. Instead, the primary contributions of this paper are applications of clustering techniques for event detection. Specifically, we investigated:

- semantic and temporal properties of events;
- document clustering based on content and temporal adjacency (rather than just content);
- event detection based on *similarity* versus *novelty*;
- evaluation methods for retrospective and on-line detection.

2 Event Analysis

In order to investigate the nature of events and to evaluate the effectiveness of detection algorithms, the TDT project prepared a collection of 15,836 news stories, in

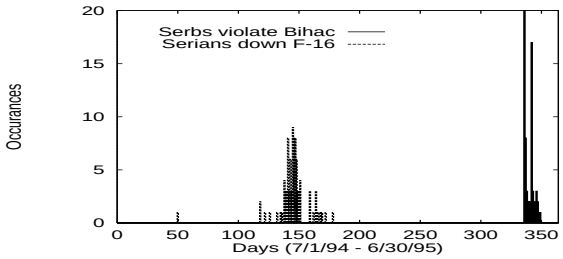

Figure 1: Histograms of *Serbian*-related events

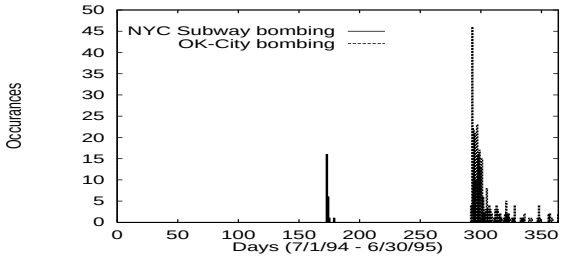

Figure 2: Histograms of *bombing*-related events

which 25 events were identified by the TDT researchers[2]. The only guideline explicitly given for event definition was that an event should identify *something (non-trivial) happening in a certain place at a certain time.* This property makes events differ from topics. For example, *the TWA-800 airplane crash* is an event but not a topic, and airplane accidents is a topic but not an event. This distinction gives rise to reporting patterns of events and their evolution over time. Since selecting the events from the TDT corpus entailed an initial random sampling of that corpus, a bias towards larger events (those reported more often) was evident. The 25 events selected contain different numbers of stories, ranging from 2 stories for *Cuban riot in Panama* to 273 stories for *OK City bombing.* The entire corpus of stories was manually labelled; each story was assigned a label of YES, NO or BRIEF with respect to each of the 25 events. The corpus contains stories about more events than the 25 labeled ones; unlabeled events were not used in the evaluation.

An interesting characteristic of news stories is that events are often associated with news bursts. Figures 1 and 2 illustrate the temporal histograms of a few events, where the X-axis of each graph is time (numbered from day 1 to 365), and the Y-axis is the story count per day. Several patterns emerged from our observations of temporal event distributions:

- News stories discussing the same event tend to be temporally proximate, suggesting the use of a combined measure of lexical similarity and temporal proximity as a criterion for document clustering.
- A time gap between bursts of topically similar stories is often an indication of different events (e.g., different earthquakes, airplane accidents, political

[2]The TDT corpus consists of 15,836 news stories in the time period from July 1, 1994 to June 30, 1995. Roughly a half of these stories are Reuters articles, and the other half are from multiple programs of CNN broadcast news. This corpus is available via CMU web site to members of the Linguistic Data Consortium – email: yiming@cs.cmu.edu.

crises, etc.), suggesting a need for monitoring cluster evolution over time, and a benefit from using a time windows for event scoping.

- A significant vocabulary shift and rapid changes in term frequency distribution are typical of stories reporting a new event, indicating the importance of dynamically updating the corpus vocabulary and statistical term weights. A timely recognition of new patterns, including previously unseen proper names and proximity phrases, in the streams of stories is potentially useful for detection of the onset of a new event.

These points will be addressed further in the next section where the design of our document-clustering algorithms for event detection are described.

3 Detection Methods

Retrospective event detection is the task of grouping stories in a corpus where each group uniquely identifies an event. On-line event detection is the problem of labeling each document as it arrives in sequence with a *New* or *Old* flag, indicating whether or not the current document is the first story discussing a novel event at that time. We investigated two clustering methods: an agglomerative (hierarchical) algorithm based on *group-average clustering* (GAC), and a single-pass algorithm (incremental clustering or INCR) which generates a non-hierarchical partition of the input collection. GAC is designed for batch processing, and is used for retrospective detection. INCR is designed for sequential processing, and is used for both retrospective detection and on-line detection.

3.1 Cluster representation

We share a common representation for documents and clusters in our detection and tracking algorithms. We employ the conventional vector space model[5] which uses the *bag-of-terms* representation. A document (story) is represented using a vector of weighted terms (words or phrases). The normalized vector sum of documents in a cluster is used to represent the cluster, and called the *prototype* or *centroid* of the cluster. Terms in a document vector or a cluster prototype are statistically weighted using the term frequency (TF) and the Inverse Document Frequency (IDF) and are appropriately normalized. We only keep the k top-ranking terms (at the most) per vector, and ignore the remaining terms. The value of k is empirically chosen to optimize detection or tracking performance. We use the standard cosine similarity, i.e., the cosine value between document and cluster prototype vectors to measure their similarity.

We employ the SMART 11.0 system (developed at Cornell)[5] for document preprocessing, including the removal of stop words, stemming, and term weighting. SMART also provides several term weighting schemes, of which we found the *ltc* option yielded in the best detection results in our experiments. Given term t in document d, the *ltc* weight is defined as:

$$w(t,d) = (1 + \log_2 TF_{(t,d)}) \times IDF_{(t)} / \|\vec{d}\|.$$

The denominator $\|\vec{d}\|$ is the 2-norm of vector $\vec{d}$, i.e. the square root of the squared sum of all the elements in that vector. IDF, standing for *Inverse Document Frequency*, is a corpus-level statistic, defined to be N/n_t where N is the total number of training documents, and n_t is the number of training documents which contain term t.

3.2 GAC-based hierarchical clustering

Basic GAC algorithm

Group Average Clustering (GAC) is an agglomerative algorithm which maximizes the average similarity between document pairs in the resulting clusters [7, 9]. Straightforward GAC algorithms typically have a complexity in time and space quadratic to the number of input documents [3], which is less economical or tractable for large applications than simpler methods, such as *single-link* clustering. Cutting et al. presented an iterative bottom-up algorithm aiming for a compromise between cluster quality and computational efficiency[3]. In each iteration, it divides the current set of active clusters/documents into *buckets*, and does local clustering within each bucket. The process repeats and generates clusters at higher and higher levels, until a pre-determined number of top-level clusters are obtained. This 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 $m \leq n$.

Bucketing and reclustering

When applying the above algorithm to event detection, we based the bucketing of documents/clusters on the temporal order of the documents. Our motivation is not just computational efficiency, but the exploitation of temporal proximity of news stories discussing a given event. Most of the manually labelled events in the TDT corpus last no longer than 2 months. The fact that events tend to appear in news bursts makes it reasonable to bucket stories according to their order in time. In other words, our strategy gives a higher priority to grouping consecutive stories rather than temporally disparate ones.

The input to the GAC algorithm is a document collection, and the output is a forest of cluster trees with the number of trees specified by the user. Clusters are produced by growing a binary tree in a bottom-up fashion: the leaf nodes of the tree are single-document clusters; a middle-level node is the merged cluster of the two most similar lower-level clusters. By default, the bottom-up clustering continues until the root node is created, which represents the universal cluster containing all clusters and therefore all the stories. If the desired number of clusters is pre-specified, then the algorithm stops when that number of clusters is reached rather than proceeding to the root. The algorithm consists of the following steps:

1. Sort the stories in chronological order, and use this as the initial partition of the corpus where each cluster consists of a single document.

2. Divide the current partition into non-overlapping and consecutive buckets of fixed size.

3. Apply GAC to each bucket by combining lower-level clusters into higher-level ones in a bottom-up fashion until the bucket size (number of clusters in it) is reduced by a factor of ρ, called the reduction factor.

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 2-4, until a pre-determined number of top-level clusters is obtained in the final partition.

6. Periodically (once per k iterations in Step 5) re-cluster the stories within each of the top-level clusters, by flattening the component clusters and re-growing GAC clusters internally from the leaf nodes.

The re-clustering step is our augmentation to Cutting's algorithm. This step is useful when events straddle the initial temporal-bucket boundaries; subsets of stories discussing the event within different buckets are often clustered together with somewhat similar stories at a low level, and are only later assembled in a higher level node of the cluster tree. Subsequent re-clustering reduces the systematic bias of the initial bucketing, and therefore results in tighter clusters than those obtained without re-clustering.

Tunable parameters

Several tunable parameters are used in our algorithm, including:

1. the bucket size (number of clusters) which limits the scope of the GAC clustering in each iteration;

2. the reduction factor ρ in each iteration;

3. the minimum similarity threshold for two clusters to be combined;

4. the number of terms to keep in each cluster prototype;

5. the term weighting scheme;

6. the number of iterations between re-clustering.

Parameter tuning is an empirical issue. Table 2 shows parameter values typical of those used in our retrospective detection experiments.

Table 2. Typical parameters used in retrospective GAC

bucket size	= 400
clustering threshold	= .2
terms per vector	= 100
term weighting	= ltc
reduction factor ρ	= 0.5
# of iterations b/w re-clustering	= 5

3.3 Single-pass clustering

The incremental clustering algorithm is quite simple. It sequentially processes the input documents, one at a time, and grows clusters incrementally. A new document is absorbed by the most similar cluster generated previously, if the similarity between this document and the prototype of that cluster is above a pre-selected threshold; otherwise, the document is treated as the seed of a new cluster. By adjusting the threshold, one can obtain clusters at different levels of granularity. We made additional efforts to exploit the dynamic nature of the input data and the temporal properties of events; these efforts are described in the following sections.

Incremental IDF

A task-specific constraint in on-line detection is the prohibited use of any information about future stories, i.e., documents posterior to the current point of processing. This raises the issue about how to deal with the growing vocabulary from incoming documents and the dynamic updating of corpus-level statistics such as IDF, which impact term weighting and vector normalization and thus affect document clustering.

Two possible approaches to the above problems would be to:

1. Obtain a fixed vocabulary and static IDF statistics using a retrospective corpus in a similar application domain (e.g., CNN or WSJ news stories prior to the period of the TDT stories), and use this vocabulary and IDF values for term weighting in newly coming documents/clusters. Assign a constant weight to the out-of-vocabulary (OOV) terms, or use some other kind of *smoothing* of term weights.

2. Incrementally update the document vocabulary and recompute IDF each time a new document is processed. An empirical analysis shows that an incremental IDF approach can be effective in document retrieval after a sufficient number of "past" documents have been processed[1].

We chose to combine both approaches, starting with IDF statistics of a retrospective corpus, and updating the IDF with each incoming document. The incremental Inverted Document Frequency (IDF) is defined to be:

$$IDF_{(t,p)} = \log_2(N_{(p)}/n_{(t,p)})$$

where p is the current time, t is a term, $N_{(p)}$ is the number of accumulated documents up to the current point (including the retrospective corpus if used), and $n_{(t,p)}$ is the number of documents which contain term t up to the current point.

Time window and decaying function

For on-line detection, we use a time window to limit prior context to m previous stories. For each current document in the sequential processing, the similarity score of each document in the time window is computed. A flag of *New* is assigned to the document if all the similarity scores in the window are below a pre-determined threshold. The *confidence score* for this decision is defined to be:

$$score(x) = 1 - \max_{d_i \in window} \{sim(\vec{x}, \vec{d_i})\}$$

where x is the current document, d_i is the i-th document in the window, and $i = 1, 2, \ldots, m$.

We also tested a decaying-weight function where documents further removed in time have progressively less influence on the current decision. We use a modified formula for the confidence score of document x:

$$score(x) = 1 - \max_{d_i \in window} \{\frac{i}{m} sim(\vec{x}, \vec{d_i})\}.$$

This method provides a smoother way to use the temporal proximity than a uniformly-weighted window. Note that for simplicity we define the time window over documents, rather than clusters or time periods; however, it is easy to generalize these definitions from documents to such larger groupings.

These windowing strategies yielded measurable improvements in our on-line detection experiments, enhancing precision with only a small sacrifice in recall. The i/m linear-decay temporal window yielded consistently better results than the uniformly-weighted window.

Similarly, we investigated time windowing in INCR clustering for retrospective detection. In the experiments with other parameters fixed, using a window of 2000 documents (covering about 1.5 months of time) improved the performance score from 0.64 to 0.70 in the F_1 measure[7] (defined in the evaluation section).

Detection thresholding

We use two user-specified thresholds to control the detection decisions by the incremental algorithm: the *clustering threshold* (t_c), and the *novelty threshold* (t_n). The former determines the granularity of the resulting clusters, which is essential for retrospective event detection, and the latter determines the sensitivity to novelty, which is crucial for on-line detection.

Letting $t_c \geq t_n$, and $sim_{max}(x) = 1 - score(x)$, our on-line detection rules are defined to be:

- If $sim_{max}(x) > t_c$, then set the flag to *OLD*, and add document x to the most similar cluster in the window;
- if $t_c \geq sim_{max}(x) > t_n$, then set the flag to *Old*, and treat document x as a new singleton cluster;
- if $t_n \geq sim_{max}(x)$, then set the flag to *New*, and treat document x as a new singleton cluster.

Using both thresholds permits better empirical optimization for different tasks. For instance, $t_c = t_n$ is appropriate for retrospective clustering (i.e., t_n is not needed), but for on-line detection we found that not using clustering ($t_c = \infty$) is better. Tables 3 and 4 show the parameter values typically used for in our retrospective detection and on-line detection experiments with INCR.

Table 3. Typical parameter values in retro. INCR

window size	= 2000
clustering threshold	= .23
terms per doc vector	= 125
term weighting	= ltc

Table 4. Typical parameter values in on-line INCR

window size	= 2500 linear decay
clustering threshold	= ∞
novelty threshold	= .16
terms per doc vector	= no limit
term weighting	= ltc
IDF	= retro + on-line updating

4 Evaluation

Detection effectiveness was evaluated using the 25 human-labeled events (about 7% of the total stories) in the TDT corpus, although the detection systems were run on the entire corpus and (presumably) detected many more events outside these 25 on which they were not evaluated.

4.1 Retrospective detection results

The official evaluation in the TDT project required each retrospective detection system to generate a partition of the corpus, i.e., non-overlapping clusters which together span the entire TDT corpus. A system may generate any number of clusters, but is only evaluated on the 25 reference events. After the partition is generated, the cluster that best matches each of the 25 labeled events is used for evaluation, via 25 contingency tables.

Table 5. A cluster-event contingency table

	in event	not in event
in cluster	a	b
not in cluster	c	d

Table 5 illustrates the two-by-two contingency table for an cluster-event pair, where a, b, c and d are document counts in the corresponding cases. Five evaluation measures are defined using the contingency table, including miss, false alarm (f), recall (r), precision (p) and the F_1 measure (F_1):

- $miss = c/(a+c)$ if $a+c > 0$, otherwise undefined;
- $f = b/(b+d)$ if $b+d > 0$, otherwise undefined;
- $r = a/(a+c)$ if $a+c > 0$, otherwise undefined;
- $p = a/(a+b)$ if $a+b > 0$, otherwise undefined;
- $F_1 = 2rp/(r+p) = 2a/(2a+b+c)$ if $(2\times a+b+c) > 0$, otherwise undefined.

To measure global performance, two averaging methods are used. The *micro-average* is obtained 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. The *macro-average* is obtained by producing per-event performance measures first, and then averaging the corresponding measures.

Table 6 shows our best result by the incremental clustering algorithm in the official TDT retrospective detection evaluation where each detection system is required to produce a partition of the entire corpus. Table 7 shows the improved results that are obtained when potentially-overlapping clusters are permitted. The CMU results correspond to the modified GAC method described earlier. The (available) results by UMass and Dragon are also included for comparison, according to their reports at the TDT workshop[10].

Table 6. Retrospective detection results – partition required

	CMU (INCR)	UMass (no-dupl)	Dragon (multi-pass)
Recall (%)	62	34	61
Precision (%)	82	53	69
Miss (%)	38	66	39
False Alarm (%)	.04	.09	.08
micro-avg F_1	.71	.42	.65
macro-avg F_1	.79	.60	.75

Table 7. Retrospective detection results – cluster overlap and hierarchy allowed

	CMU (GAC)	UMass (dupl)
Recall (%)	75	73
Precision (%)	90	78
Miss (%)	25	27
False Alarm (%)	.02	.06
micro-avg F_1	.82	.75
macro-avg F_1	.84	.81

These results show that allowing cluster hierarchy (CMU:GAC) and cluster overlap (UMass: *dupl*) yielded better results than requiring a corpus partition. We believe the main reason for the better performance of GAC is the multi-leveled clusters which enable the detection of events at any degree of granularity. This representational power of GAC comes with a cost of resulting a larger number of clusters (about 12,000 in this particular run), than the number of clusters (5,907) in the partition by INCR. This difference, however, may not have a significant effect on the end-user, if the cluster hierarchies will be used for a scatter-gather type of navigation or query-driven retrieval, where the search steps needed are much less than the total number of clusters.

In the results of the partition-producing algorithms, we were surprised that the simplest approach – the single-pass clustering by INCR (CMU) – worked as well as the multi-pass k-means clustering method by Dragon. This may be partly because of the temporal proximity of events which simplifies the clustering problem.

4.2 On-line detection results

The required output of an on-line detection system is a decision of *New* or *Old* for an incoming document with a confidence score for that decision. Since there are only 25 events (containing 1131 stories) defined in the TDT corpus, and each event has only one story as the first report of that event, only 25 stories should have a flag of *New* for the entire corpus. This is too small a number for a statistically reliable estimation of performance. To improve the reliability, an 11-pass detection evaluation was conducted. After each pass, the first story of each events is removed, and detection and evaluation are applied again to the corpus. The eleven passes are labeled by $N_{skip} = 0, 1, \ldots, 10$. For each pass, a contingency table is constructed for evaluation, as shown in Table 8.

Table 8. On-line detection contingency table

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

Table 9 and Figures 3 and 4 summarize the results by CMU, UMass and Dragon. Both CMU and UMass conducted multiple runs with different parameter settings; here we present the best result for each site with respect to the F_1 measure. CMU's results correspond to the parameters discussed earlier (Table 4). Note that both CMU and UMass chose to use individual documents instead of clusters to represent the past in on-line detection, while Dragon used a single pass of their k-means clustering approach[3]. We interpret the better results of not using clustering as the following: in order to pass the *novelty test*, a story has to be sufficiently different from *all* of the past stories; this is a stronger condition than being more novel than the *average*.

Table 9. On-line new event detection results

	CMU no clust	UMass no clust	Dragon one-pass clust
Recall (%)	50	49	42
Precision (%)	37	45	21
Miss (%)	50	51	58
False Alarm (%)	1.89	1.31	3.47
micro-avg F_1	.42	.47	.28
macro-avg F_1	.42	.47	.28

Note that the scores in Table 9 only measure how well each system did at a specific trade-off level of recall and precision. In order to measure continuous trade-off between recall and precision, we present the recall-precision curves (Figure 3) and the Decision Error Trade-off (DET) curves [4]. These curves were obtained by moving thresholds on the confedence scores of detection decisions. We used the DET software provided in the TDT project to generate the DET curves, and converted each data point (a pair of miss/false-alarm values) in these DET curves to the corresponding recall and precision values (non-interpolated) to obtain the recall-precision curves. The CMU results are depicted by the solid lines, which show better performance at the high precision area. As is especially evident in Figure 3, the CMU, UMass and Dragon approaches exhibit very different behaviors, inviting further detailed investigation.

4.3 Behavior analysis

In order to compare the behavior of our algorithms to human judgments, we contrast the temporal histograms of system-generated clusters for retrospective detection with corresponding histograms by human judgments. Figures 5-8 show the pairwise histograms on two events for GAC and INCR, respectively. Figure 9 shows the GAC performance on all the 25 events. The upper half of each graph is the histogram of human-labeled documents for an event; the 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, then their histograms will be completely symmetric, mirroring each other.

As the figures show, GAC and INCR have complementary strengths and weaknesses. GAC shows surprisingly symmetric graphs for most events except those with significant temporal extent, and GAC is particularly suitable for recognition of large news bursts. INCR, on the other hand, has less symmetric performance compared to GAC, but is better at tracking long-lasting events (*DNA in O.J. trail* and *Death of Kim Jong Il*). The observed behavior may come partly from the different biases in these algorithms and partly from the parameter settings in the particular experiments.

[3] Multi-pass clustering is not allowed because, by the task definition, future knowledge is not available at the decision making point.

[4] The Decision Error curves, which plot miss and false alarm, are analogous to precision-recall trade-off curves. Better performance corresponds to proximity to the origin. The original DET software was provided by the TDT project sponsor, and the adapted version was implemented by the UMass group with richer options.

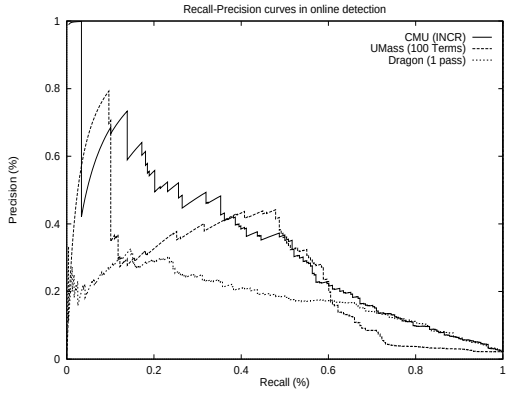

Figure 3: On-line detection Recall-Precision curves

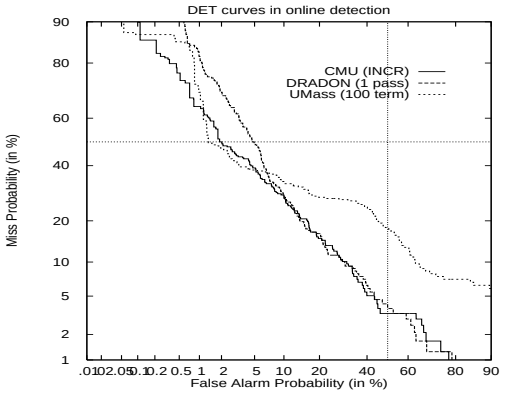

Figure 4: On-line detection DET curves

5 Concluding Remarks

Event detection, both retrospective and on-line, represents a new family of tasks for IR. The results of our pilot study on these tasks (reinforced by the results of UMass and Dragon) show that basic techniques such as document clustering can be highly effective if the problems are well defined, and when content information and temporal information are jointly and properly used.

For retrospective detection, when requiring a strict partition of the document space, GAC, INCR and the k-mean clustering algorithm by Dragon exhibit comparable performance; when the partition requirement is relaxed, the hierarchical GAC approach is the best.

On-line novel-event detection is somewhat more difficult than retrospective detection. Non-clustering techniques have demonstrated better detection accuracy than clustering approaches, although further investigation is needed for a better understanding.

In spite of the reasonable results obtained by CMU, Dragon and UMass, much work remains to be done. Research questions for further investigation include:

- How can we exploit multiple input streams (e.g. CNN, AP, UPI, ...) to reinforce each other, cross-validating topical clusters?
- How can we better exploit the temporal patterns of proper names or proximity phrases which appear to be highly informative (to humans, at least) as event indicators?

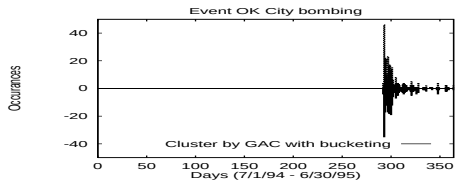

Figure 5: Event *OK City bombing* vs GAC-cluster

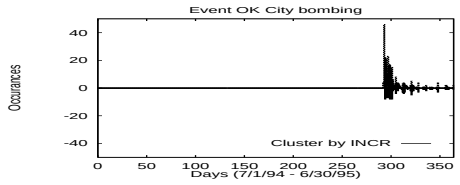

Figure 6: Event *OK City bombing* vs INCR-cluster

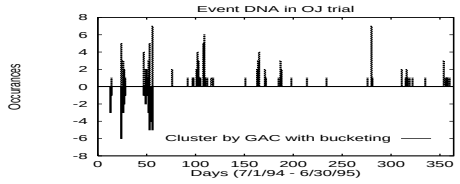

Figure 7: Event *DNA in OJ trial* vs GAC-cluster

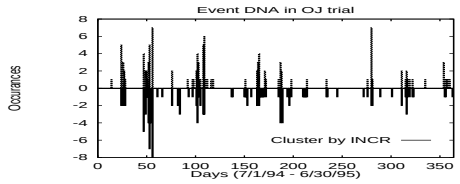

Figure 8: Event *DNA in OJ trial* vs INCR-cluster

- How can we provide a global view of the information space of retrospectively clustered events and emerging newly-detected ones?
- How can we make the user *actively* involved in cluster-based navigation, e.g. by permitting zoon-in and zoom-out options, and by providing summaries at different degrees of granularity, i.e. at a corpus level, a cluster level, a document level, and a sub-document level?
- How can we evaluate and compare the usefulness of a cluster hierarchy (or a cluster set) in assisting the user in *query-free* or *query-driven* retrieval? Shall we measure, for example, how quickly the user can find the relevant cluster(s), and use the time as an evaluation criterion?

Acknowledgements We thank Charles Wayne and George Doddington from DoD for their guidance in the TDT task definition and evaluation, and James Allan at UMass and Jon Yamron at Dragon for sharing ideas/results in the research. The TDT research is sponsored by DoD. However, any opinions or conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

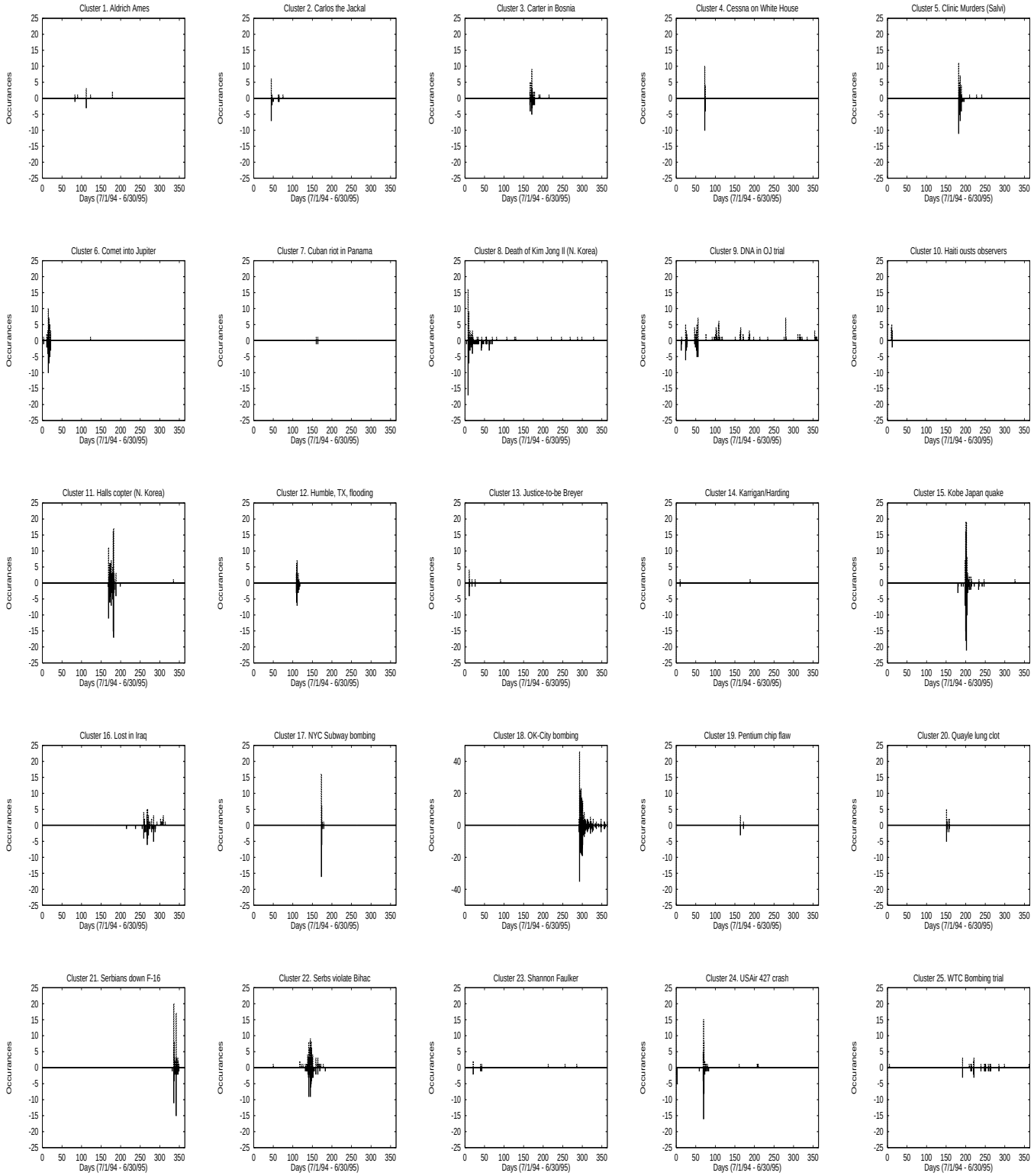

Figure 9: Pairwise histograms of the 25 TDT events vs the best-fit clusters by GAC

References

[1] Jamie Callan. Document filtering with inference networks. In *Proceedings of the 19th Annual ACM/SIGIR conference*, pages 262–269, 1996.

[2] J.G. Carbonell, J. Goldstein, and Y. Geng. Automated query-relevant summarization and diversity-based reranking. In *Proceedings of the IJCAI-97 workshop on AI in Digital Libraries*, Nagoya, Japan, 1997.

[3] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *15th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, 1992.

[4] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Poceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 434–444, 1988.

[5] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Pennsylvania, 1989.

[6] R.H. Tohompson and B.W. Croft. Support for browsing in an intelligent text retrieval system. In *International Journal of Man-Machine Studies*, pages 30(6)639–668, 1989.

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

[8] Ellen M. Voorhees. Implementing allgomerative hierarchic clustering algorithms for use in document retrieval. In *Information Processing & Management*, volume 22:6, pages 465–476, 1986.

[9] R. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management.*, 25(5):577–597, 1988.

[10] Y. Yang, J.G. Carbonell, J. Allan, and J. Yamron. Topic detection and tracking: Detection-task. In *Proceedings of the Workshop of Topic Detection and Tracking*, 1997.