Taking Topic Detection From Evaluation to Practice

James Allan, Stephen Harding, David Fisher,
Alvaro Bolivar, Sergio Guzman-Lara, and Peter Amstutz
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst

Abstract*— *The Topic Detection and Tracking (TDT) research community investigates information retrieval methods for organizing a constantly arriving stream of news articles by the events that they discuss. Our best system for the open evaluations of TDT has used an approach that turned out to be problematic when the cluster detection technology was deployed in a real world setting. To avoid generating "garbage" clusters, we had to revert to a different approach and to explore engineering solutions that were not motivated by the model. Our experiences also led us to propose extensions to the formal TDT evaluation.

I. Overview

The Topic Detection and Tracking (TDT) research community investigates information retrieval methods for organizing a constantly arriving stream of news articles by the events that they discuss. TDT is explored in an open and cooperative evaluation sponsored by DARPA and run by NIST; the evaluations have run every year since 1998.

One of the organization tasks included in TDT is *topic detection*, where systems cluster arriving stories into bins depending on the topic (event) being discussed. For example, stories that discuss the same bombing should be grouped together, but other bombings at the same or different locations should be grouped separately. Systems are typically required to process each story before considering the next, and do not have any knowledge of the topics (bins) that will be appearing in the news.

In this paper, we describe our experience deploying a TDT detection system in two real-world applications, the unexpected changes we had to make in the research system for it to be usable in a real setting, and how those changes have resulted in substantive changes in the TDT evaluation program (starting with TDT 2004). The point of this paper is not to serve as an indictment of TDT, nor to criticize the evaluations of TDT that have taken place. Rather, this paper serves as a cautionary note for technology evaluation communities: highlighting the possibility of a mismatch between evaluation abstractions and the real world, and reinforcing the mantra that both evaluations and applications can benefit from a cyclic relationship between the two.

TDT began as a technology development and evaluation program [1]. In the DARPA-sponsored TDT evaluations, detection systems are compared by their ability to put all stories in a single topic together. The official measure is a cost function that combines system miss and false alarm rates on a per-topic basis [8]. Currently, the best systems achieve about a 0.3 cost value, typified by one system that had a 28% miss rate and a 0.3% false alarm rate on a randomly selected topic.

We initially fielded our TDT detection technology based on the best parameter values as determined by the formal evaluation. However, it quickly became apparent that the resulting clusters were of sufficiently poor quality that they could not be used: they were either too focused or—more commonly–far too broad. We also found that relationships between topics were less crisp than in the TDT evaluation data, and that algorithmic selection of topic granularity was almost never correct. For example, the system tended to group topics from the same geographical area rather than break them into events. These effects were very clear in both newswire and Web news environments.

Our failure analysis of TDT technology has contributed to several changes in the TDT evaluation. For example, starting with TDT 2004, topics will be *expected* to overlap, to be hierarchical, and to be less rooted in a single "seminal" event. Also partly inspired by deploying the technology, other tasks in TDT are moving to a supervised version that better simulates user involvement and, in particular, correction of system errors. By transferring technology from a research setting to an operational system, we have provided not only a useful application, but an improved understanding of the research problem and contributed to a more valuable research program.

In the rest of this paper we provide more details of our experience. We start in Section II by describing the TDT environment and tasks, how TDT detection systems

are evaluated, and the algorithm that performs best in the TDT evaluations. We then discuss in Section III the deployment of our TDT clustering technology in two applications, one in collaboration with other TDT sites (Section III-A) and the other as part of a local project to explore finer-grained relationships between news topics (Section III-B). In Section IV we sketch the implications of experience and how it has impacted the TDT community's evaluation program. We conclude and discuss future work in Section V.

II. TDT and Cluster detection

The Topic Detection and Tracking (TDT) evaluation program evaluates technology for organizing news by the events that are discussed in the stories [3]. It explores the issues in an environment where the news stories are in English, Chinese, and Arabic, and come either as good quality text from newswire sources, or as the output of an automatic speech recognition system applied to broadcast television or radio sources. All of the sources are brought together into a single stream that a TDT system must process sequentially, making decisions on earlier stories before it sees subsequent news articles.

The TDT evaluation program includes several subtasks: (1) Segmentation is the problem of dividing a half hour (or more) of radio or television news audio into discrete stories for later processing; (2) New event detection requires recognizing when a story in the stream discusses a topic that had not been seen previously; (3) Clustering extends that by placing all other stories into appropriate bins (clusters), such that each bin includes stories that talk about the same topic and only that topic; (4) Tracking is a partially supervised task that involves monitoring the stream of news for subsequent stories on the same topic as a handful of training examples; and (5) Story link detection is a core technology task in which a system must decide whether or not two randomly selected stories are on the same topic.

TDT evaluations are carried out on a corpus of news stories collected by the Linguistic Data Consortium, sampled to achieve an appropriate mix of sources, and formatted for the evaluation [5]. Evaluations have been carried out on three major corpora to date, incorporating stories from most of 1998 as well as the end of 2000. Stories are collected in three languages—English, Chinese, and Arabic—and in four media—newswire, radio, television, and Web. In order to allow participants to focus on the TDT technology tasks and sidestep complex data processing issues, broadcast news stories are converted to text using an automatic speech recognition system and non-English sources are translated via machine translation. As a result, a system can work entirely within English, although it must be prepared to cope with the (often substantial) errors introduced by speech recognition and machine translations. In fact, an important early and explicit goal of TDT was to explore the impact of those errorful conversions on the TDT tasks.

In order to evaluate a system's ability to organize stories by topic, the LDC also provides topic annotations of each corpus [5]. The process has varied somewhat for each evaluation as experience has pointed out concerns and as available financial support has shifted. The current process is a search guided annotation and works roughly as follows (the LDC has described the process in more detail elsewhere [5] and provides supporting information on their Web site[1].) Starting from a description of the topic, an annotator uses a search engine to find stories that discuss the topic. Then, using the stories that were found, the searcher creates new queries that locate additional stories that are judged. Finally, the searcher is encouraged to use the knowledge he or she has developed up to that point to create new and unusual queries that might find more relevant stories. These rounds of searching are capped by a quality assurance step that compares system output to LDC "truth" and adjudicates situations where they are substantially in conflict.

A. TDT cluster detection

In this paper, we will focus exclusively on the clustering task, also referred to simply as *detection* in much of the TDT literature [7], [11], [6]. This task has appeared in all of the annual TDT evaluations since their inception in 1998.

In TDT clustering, a system is conceptually provided with a news story and must place it within the most appropriate cluster—i.e., the one that discusses the same event-based topic. If the topic had not appeared previously, the system is expected to create a new cluster at that point. (The new event detection evaluation focuses particular attention on that aspect of clustering, but it is important here, too.)

In fact, TDT systems do not process stories one at a time because that would be inefficient and unrealistic. Instead, stories are provided in batches that roughly correspond to a half hour's worth of news. Variations of the evaluation process allow the batching to be extended to several hours in order to determine whether there are advantages to deferring the decision on a story's topic until later information has arrived. To date, no system has shown a pronounced advantage to batching more than the default half hour of news.

[1] http://www.ldc.upenn.edu/TDT

B. Evaluating cluster detection

All TDT tasks are envisioned as "on-line" tasks that must completely process each story in a batch before receiving any additional stories. Decisions are irrevocable, even if a mistake is detected later. This approach models a situation where the output is consumed immediately and in a time-critical fashion. It explores the core technology rather than how it might be used in an interactive setting [12].

Output is couched in terms of a detection task, where "yes" or "no" decisions must be made [8]. Evaluation is in terms of errors (misses and false alarms) and the tradeoff between them. (As with most language tasks, the errors tend to tradeoff against each other: lowering one tends to raise the other.)

The official evaluation measure of TDT is based on a cost function, a weighted combination of miss and false alarm rates:

$$\text{Cost} = C_{\text{miss}} P(\text{miss}) P(\text{target}) + C_{\text{fa}} P(\text{fa}) P(\text{off-target})$$

where $P(\text{target})$ is the prior probability that a story will be on topic, C_x are user-specified values that reflect the cost associated with each error, and $P(\text{miss})$ and $P(\text{fa})$ are the actual system error rates. Within TDT evaluations, $C_{\text{miss}} = 10$, $C_{\text{fa}} = 1$, and $P(\text{target}) = 1 - P(\text{off-target}) = 0.02$ (derived from training data).

In fact, a normalized version of the cost function is used. A system that always answers "no" would have no false alarms, though it would have a 100% miss rate. That system would get a score of 0.2 (10×0.02). Similarly, a system that always answers "yes" would get a score of 0.98. To ensure that systems that underperform such simple approaches are visible, the cost value is divided by the minimum of the "always say yes" or "always say no" approaches; in this case, by 0.2. A normalized detection cost of 1.0 means that the system performs exactly as well as a system that does no work.

All evaluations in TDT have been carried out by the National Institute of Standards and Technology. Participating sites were provided with the corpus and information that specified the starting condition for each task. Each site generated its decisions on the stories in the evaluation set and submitted them to NIST. In turn, NIST did the evaluation and generated comparative results of all the systems [8].

C. Cluster detection technology

The cluster detection algorithm that we used in the most recent formal evaluation (TDT 2003) worked as follows. Collection-wide statistics such as document occurrence frequency of terms, average document length, and so on, are updated as a half-hour batch of stories arrives. Then, for each story in the batch:

1) Weight each term t in the story by
$$w_{t,s} = \frac{\text{tf}_{t,s} \cdot \log((0.5 + N)/\text{df}_t)}{\log(1.0 + N)}$$
where $\text{tf}_{t,s}$ is the number of times term t occurs in the story s, df_t is the number of stories to date in which term t occurs (including those in this batch), and N is the total number of stories seen to date (again, including this batch).
2) Select the top-weighted 1,000 terms from the document and create a vector of those weighted terms to represent its content. This threshold is sufficiently high that for most stories all terms are included, but particularly long stories are represented by a more focused set of terms. It also puts a cap on the storage space for vectors.
3) Compute the cosine similarity of the story to *every* previous story in the collection:
$$\text{sim}(A, B) = \frac{\sum_t w_{t,A} \cdot w_{t,B}}{(\sum_t w_{t,A}^2 \sum_t w_{t,B}^2)^{0.5}}$$
4) If the similarity to the nearest neighbor is above a threshold $\theta = 0.21$ (determined empirically), assign the new story to the cluster of that neighbor.
5) If the similarity is *below* the threshold θ, then form a new singleton cluster containing just that story. This approach is the same idea used to create single-link clusters.

We and others have tried a large number of additional techniques to improve overall effectiveness, none of which has improved performance with respect to the TDT cost function, and most of which have degraded effectiveness.

- Within the IR community, a weighting function that incorporates the "Okapi tf function" [15]—i.e., roughly $\text{tf}/(\text{tf}+2)$—is generally more effective than the "raw tf" that we have been using. We and others have carried out exhaustive experiments on training data but achieved substantially lower TDT costs using raw tf than the Okapi tf [16], [4].
- We felt that it would be useful to agglomerate stories in a batch into small and very focused clusters before adding them to the larger cluster set. Our intuition was that a new topic might be strengthened if it were grouped with other, later, stories on the same topic. Other groups have had more success with this approach, though not to a huge degree [6].
- It seemed that additional supporting information would be helpful, so we tried to decide to which cluster a story should be assigned using the top $n > 1$ most similar stories rather than just the top $n = 1$. Similarly, representing clusters of stories by

their centroid (rather than by the individual stories) was not effective.
- We tried to select the features more carefully than just using the 1,000 top weighted terms. We experimented with a wide range of caps and with techniques such as up-weighting or using only terms occurring in named entities. We explored techniques for removing category-specific words that cause confusion in other TDT tasks [10].
- We observed that the likelihood that two stories are on the same topic decreases as the time between their appearance increases. We incorporated a time penalty into our similarity function, but other than in some early results [2], were never able to make it work well.

We cannot claim that any of those approaches is useless, but they did not provide any value within the scope of the TDT evaluations. As we will discuss below, the single-link (compare to $n = 1$ stories) approach is clearly problematic when using other data, and we hypothesize that some of the other decisions may be similarly context dependent—that is, with different evaluation parameters (corpora, topics), we might find some of those ideas turn out to be useful.

III. Deploying Cluster Detection

The system described above—vector space, single-link clustering, etc.—is among the best performing systems in the TDT evaluations. It represents state-of-the-art technology and is an obvious basis for transferring the technology into a real-world application. We did just that in two different settings. First, we participated in an "integrated feasibility experiment" (IFE) as part of the DARPA TIDES program to deploy our cluster detection system as a component of a report writing application for intelligence analysts. Second, in an effort to more prominently showcase the capabilities of our system, we developed our own Event Organizer application and tasked it with organizing news gathered from Web sources.

In this section we describe our experience working within the IFE environment and how it forced us to retool our technology. We then discuss the Event Organizer in more detail and sketch the directions that experience is pushing us.

A. TIDES IFE

The DARPA TIDES (Translingual Information Detection Extraction and Summarization) program includes "integrated feasibility experiments" (IFEs) as part of its technology program. On an approximately annual cycle, "best of breed" technology components are gathered from contractors (or other interested parties) and combined into an overall system to address a particular task. To date, the systems have been variations of report writing assistants for intelligence analysts. That is, the systems monitor and organize a stream of constantly arriving news material, and provide the user with search, browsing, and reporting writing capabilities. The overall system's effectiveness is measured by the proportion of material found and/or viewed that was useful.

Our contribution to this system was our cluster detection system. Coordination and evaluation of the project was handled elsewhere, and that site also provided communication stubs that we could use to make our technology available on the Internet. We developed a communications protocol that could accept batches of stories, assign them to clusters, and communicate those cluster IDs back to the submitting system. Our component of the IFE system was required to be available without significant interruption.

In addition to ensuring that our system was more robust than typically required by a research system, we had to retool it to cope with a *genuine* stream of stories rather than to simulate a stream as in the research task. For example, because the research systems worked with static corpora and merely "pretended" that the stories arrived in small batches, we were able to take shortcuts and pre-process large parts of the problem. Converting to a system that actually received the stories in a batch was entirely an engineering issue, but consumed an incredible amount of time. In situations where a research component might be fielded, this challenge is one that should be taken into account from the start.

We also had to cope with a substantial increase in the arrival rate of news stories. Although we were used to processing an entire corpus that might have 40,000 stories, the IFE system provided 2-3000 stories per day, ran for several months, and is intended to run indefinitely. The nearest neighbor approach (1-NN) requires that an arriving story be compared to *every* prior story, taking time that grows approximately linearly with the size of the collection. We found that after around 100,000 stories were in the system, we could no longer process one batch of incoming stories before the next batch arrived.

The feedback we received primarily addressed issues of cluster size: they were too large or too small. In the latter case, our system generated a large number of singleton clusters—i.e., clusters containing a single news story. For the most part, this was a mismatch of expectations: there *were* large numbers of stories discussing an event-based topic that appeared only once, but it appears the IFE users were expecting the clustering system to group them together by subject in that case. So, for example, all singletons about a particular broad subject might be put into a cluster. We felt that this

problem was best addressed by a move to hierarchically organized clusters—the TDT cluster detection system was operating correctly by TDT standards, but for actual deployment of the system, more organization was desireable.

The greater problem, however, was that the quality of some created clusters was very poor. We experienced the well-known problem with single-link types of clustering: large "stringy" clusters were created where completely unrelated documents would be in the same cluster. This effect occurs because similarity is not transitive: if A is similar to B and B is similar to C, it does not follow that there is any relationship between A and C. We first tried raising the threshold θ to make it less likely that stories would be inserted into a cluster, but that merely fragmented useful clusters. We still found "garbage" clusters containing close to 1000 stories of nearly random content. Such clusters were usually triggered by non-news stories such as weather, sports, or financial news, that contained essentially random words (e.g., the names of sports teams or companies) that would cause a spurious match which would start a cascading series of unfortunate links.

In the end, we solved this problem by changing to a form of average-link clustering. We represented every existing cluster by its centroid, or the average of all stories in the cluster. As a result, a story is inserted in a cluster only if it matches, on average, all of the stories in a cluster more than in any other cluster. This not only (mostly) resolved the problem with garbage clusters, but addressed the scale problem: a newly arrived story is now compared to every cluster rather than to every story. We found that 100,000 stories might generate a few thousand clusters, so this change substantially increases the number of stories that can be processed in a limited time.

The clear advantage of a centroid approach over an 1-NN approach led us to verify our research results. It is definitely the case that with all other components as described above, 1-NN is substantially more effective than centroids in the TDT evaluations, but the opposite is true on this real dataset.

Even with the huge garbage clusters less likely to occur, system developers felt that the clusters being generated were too large, as it was common for some to include 100 or more stories. We explored several approaches to address this problem:

- We increased the threshold θ to make it more likely that new clusters could be created. We found that if the threshold were raised sufficiently to keep clusters small, it resulted in too much fragmentation: we ended up with only very tiny clusters.
- We allowed clusters to "age" such that it was more and more difficult to add stories to them. This was, in effect, a variation on the time penalty mentioned in Section II-C. The idea is that reporting is less and less likely to occur on a particular topic, so it should become more difficult to add stories to an older topic. Unfortunately, although this property is generally true, it is untrue sufficiently often that it is not an adequate approach to reducing cluster size.
- Our final attempt was not motivated by any property of news, but only by the goal of reducing cluster size: we effectively capped the number of stories that could be put into any cluster by making it more and more difficult for a story to be added as the cluster's size grew.

We adopted a peculiar combination of cluster size and age. Specifically, once a cluster reached a particular size, it became increasingly difficult to add stories to it. This technique worked best, providing smaller and more coherent clusters, even though nothing like it was helpful in the evaluation task.

Our final IFE system is implemented using the open source Lemur toolkit[2]. It uses a slightly different term weighting function than the TDT research had indicated (the IDF component is calculated differently) and it does not truncate vectors to only 1,000 terms.

In summary, when we deployed our cluster detection system as part of the IFE report writing environment, we were forced to:

1) re-engineer our system to reflect an environment where stories arrived in batches rather than being able to simulate such an environment;
2) convert from a single-link based approach (1-NN) that was suggested by formal evaluations to a centroid-based approach, both to generate more focused clusters and to cope with substantially increased scale; and,
3) incorporate a cluster size limitation that made it increasingly difficult for stories to be added to clusters over a selected size.

We continue to operate our cluster detection server for IFEs. As research results suggest changes in the way that event-based clustering should be done, we expect to cautiously deploy such changes in this system.

B. Event Organizer

The IFE process provided modest feedback to our research program, suggesting the need for more tightly focused clusters that do not grow too large. Beyond that one idea, we have received no suggestions that encourage

[2]Lemur is a toolkit for developing information retrieval systems in the language modeling framework. It is available at http://ciir.cs.umass.edu/lemur.

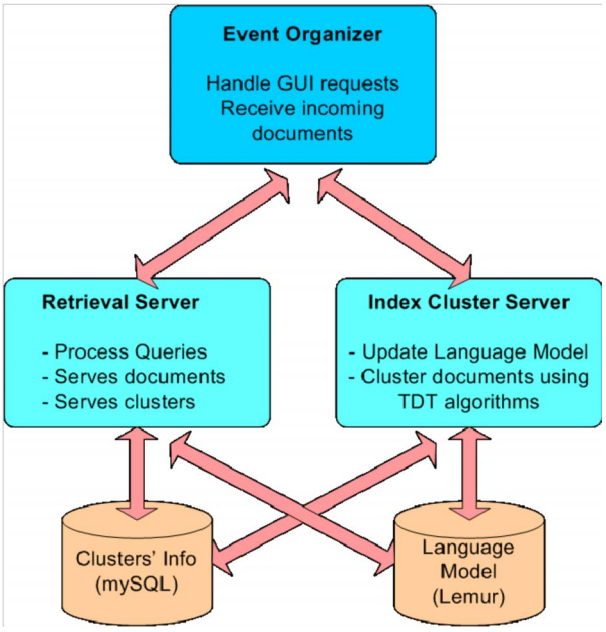

Fig. 1. Architecture of the Event Organizer system.

us to explore alternate definitions of the task. Because the IFE incorporates numerous component technology systems (e.g., summarization, retrieval, alert folders, named entity extraction, machine translation, and speech recognition), we felt that it was unlikely that user needs in cluster detection would be obvious. For that reason, we constructed Event Organizer.

The Event Organizer system consists of several parts, the core of which is the same cluster detection engine used for the IFE system. We developed our own stream of news stories by "scraping" the Web and, most importantly, developed a user interface for interacting with the resulting clusters. This interface, rather than being a general tool to integrate technologies, is designed *primarily* to explore the effectiveness and utility of event-based clustering.

We show the architecture of the system in Figure 1. The cluster detection component is in the lower-right ("index cluster server"). It is implemented using the Lemur toolkit and handles the clustering of arrived stories as well as providing a searchable index of individual news stories. The "retrieval server" (lower left) handles all user queries and serves the results back to the user interface. In addition to directly accessing the Lemur index of stories and clusters, it maintains a traditional relational database of correspondences between stories and clusters, cluster titles, and so on. This split of functionality between indexing and retrieval was done for the sake of efficiency, allowing some parallelism between continual clustering of new data and constant requests from users.

The GUI itself is not depicted in the architecture diagram, but is also connected to the Event Organizer module. The GUI is based upon the common file/folder model of storing information. In this case, news stories are placed in folders that represent event-based clusters. The user can create a profile that captures clusters of interest—a profile is in turn represented as a folder that contains all clusters that matched the query. Figure 2 shows an example of creating a profile of the 10 clusters that are most strongly related to *california*. In the figure, the profile has already been created and the top ten clusters are shown in the hierarchical view on the left.

Clusters are named with the 10 most highly weighted terms occurring in the stories inside the cluster. In this instance, clusters appear to be about forest fires in California, Schwarzenegger's election as governor of the state, a grocery store worker strike, an earthquake, wine growing issues, and even the weather.

The Event Organizer itself is the glue that holds things together. It is an event-driven component that accepts requests from the GUI, passes them to the retrieval server, and then returns responses when they are ready. It also moves data between the Web scrapers and the clustering system.

We scrape Web pages from Lycos News, CNN, Business Week, CNET, BBC, Wired News, and the Palestine Chronicle. This small set of pages results in approximately 200 added news stories each day. This is a substantially smaller number than the IFE provided, but is sufficient to highlight some limitations of the TDT technology, to wit:

- TDT is a technology evaluation and does not include user interface issues. For example, it was immediately necessary to highlight newly arrived clusters that match a profile or stories that appear in an existing cluster inside a profile. We implemented the Event Organizer in such a way that the "update profile" button (in Figure 2) would cause new stories to be added to a cluster if appropriate. Figure 3 shows (among other things) the result of an update. Although it is difficult to distinguish it in a grayscale rendering of the figure, clusters that now appear in the top 10 that did not before are highlighted in blue, whereas clusters that would no longer appear in the top 10 are marked in red. Clusters that were there and remain after the update are presented in black.
- There is no way to provide supervision to the system when it makes mistakes. We have shown that the TDT tracking task can be more effective if a user can redirect the system when it makes mistakes, even if the user does not provide regular

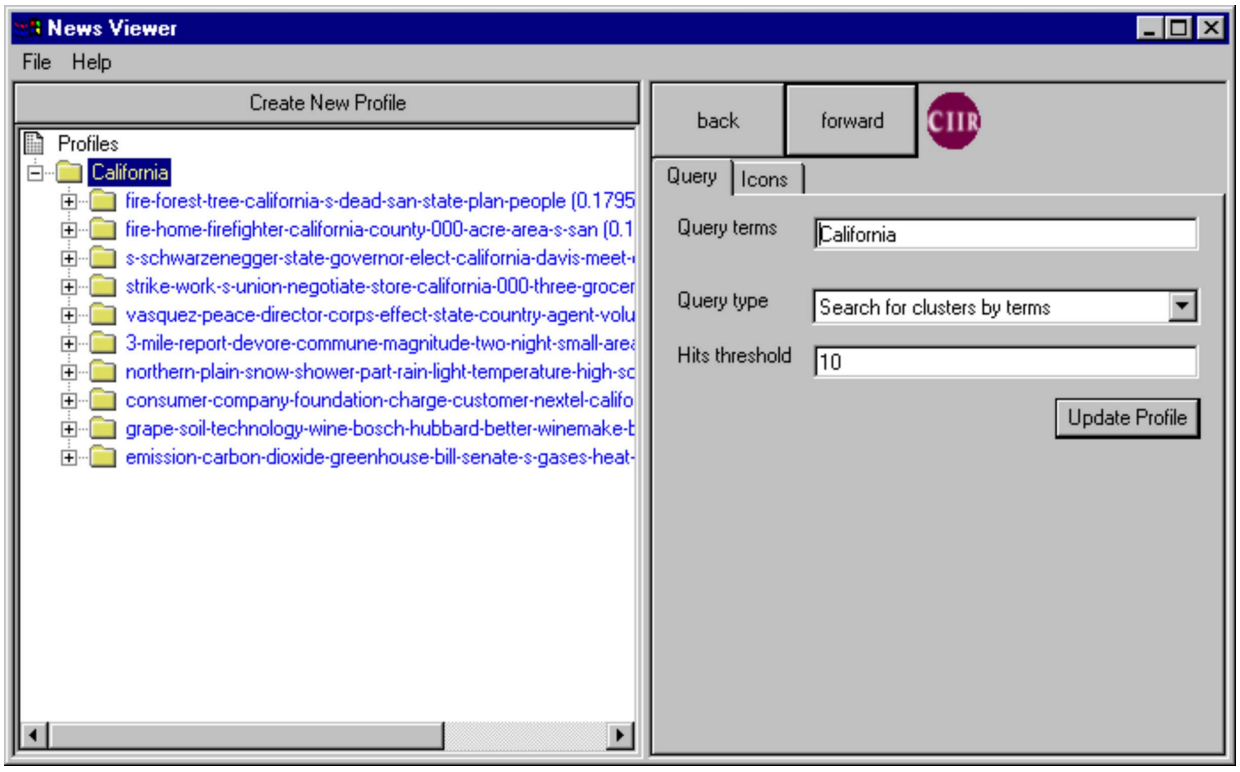

Fig. 2. Creating a new profile in Event Organizer. The simple query "california" is used both as a query and as a title for the profile. In this case the profile contains clusters that match the query.

or substantial feedback [12]. Accordingly, we have engineered Event Organizer to allow the user to correct the system's errors by removing stories from clusters and creating new clusters. Figure 3 shows a cluster about a tiger attacking Roy Horn (of Siegfried and Roy) that this user does not want to keep in the California profile. In the dropdown menu, the user is preparing to copy the cluster so that it can be used to start a new profile about that topic. It could also be deleted from the profile if the user merely wished to suppress it.

- Topic granularity is difficult to specify *a priori*. Depending on a user's interest, a topic could be brush fires in California, fighting a particular brush fire, a particular fire company's efforts to fight a brush fire, or even the burning of a specific home. The TDT evaluation program has "addressed" this problem to date by fiat: the boundaries of a topic are defined by annotator "truth" so ideally can be discovered by some sort of pattern analysis in training data. Based on our deployment experience, we believe it is critical that topics (hence clusters) be represented hierarchically so that a user can rapidly drill down or move up to find the cluster that is appropriately broad or narrow. Event Organizer supports hierarchical clusters, including the capability to create new clusters of clusters, to move clusters between clusters, and so on, though the underlying TDT technology cannot yet generate such clusters.
- Events within topics are related by more than just a contains/container connection. Although technology for discovering such relationships is in its infancy [13], we have designed Event Organizer so that clusters can be linked to clusters by a typed set of relationships. In theory, this might mean, for example, that it would be possible to jump from a trial event directly to the crime that is at issue or to the arrest of the accused. We incorporate typed links within clusters that can point to (and from) other clusters, allowing a user to jump directly to events that occurred earlier or after, that involve the same people and locations, or that discuss a similar topic.

IV. Implications and Impact

Because the Event Organizer is in its early stages, it has not been deployed outside of small pilot settings. Although we have had to battle numerous engineering issues to create an efficient and appealing interface, we

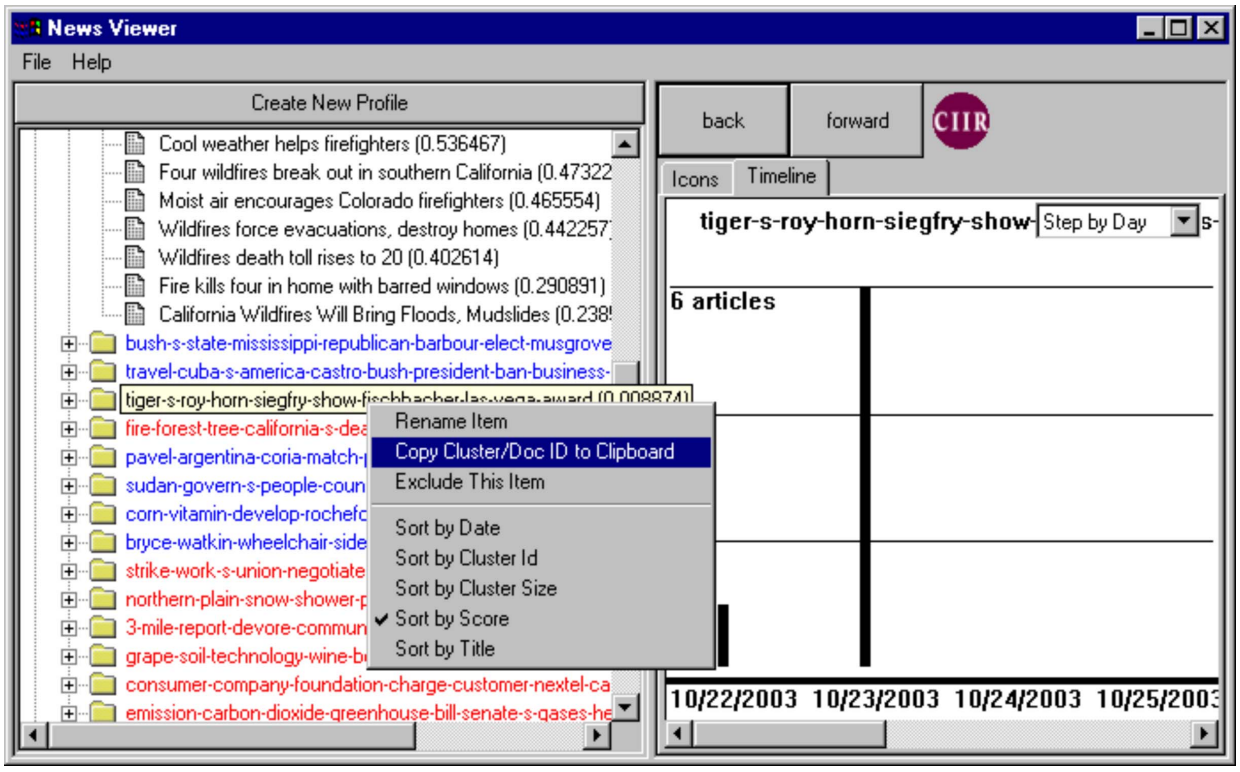

Fig. 3. The user is preparing to "cut" a cluster from this updated profile in order to use it as to "query by cluster" when starting a new profile. In this screen snapshot, the profile has just been updated and changes in which clusters should appear are highlighted by color. In addition, a timeline view of the documents in the cluster is shown on the right.

believe that the interface has already proven valuable. In combination with our IFE experience, the Event Organizer has demonstrated several directions in which the TDT technology ought to change to be more "realistic" and in which TDT evaluations should change to support it. Those changes fall into four broad areas: allowing stories to appear in multiple clusters, allowing clusters to contain other clusters, recognizing the need to find relationships between clusters, and supporting some form of interaction to correct system errors.

A. Overlapping clusters.

Since the original pilot study that set the foundation for the TDT evaluations [1], the evaluation has made the simplifying assumption that a news story discusses exactly one topic. Even at that point the community knew this assumption was not true and that there would be problems. For example, if a story discussed two topics, the simplying assumption required that it be put in a single cluster and that if it were put in the other (or, indeed, caused another cluster to be created), it would be an error.

Quick examination of early corpora had suggested that only a small percentage of stories covered multiple topics: news stories are generally very focused. However, as the size of the TDT test corpora grew, and as the number of topics being considered for evaluation also increased, the problems caused by limited topic overlap became larger. Our IFE and Event Organizer experiences have shown that the problem is much more pronounced in the "real" news we collected than in the TDT news collections.

Based on all of that experience, we recommended last year that the TDT 2004 evaluation tasks be changed to support and to require that stories be able to appear in multiple clusters. In practice, this will mean that systems will put stories into any cluster that matches above a threshold rather than to the best matching cluster.

B. Hierarchical clusters.

One of the largest areas of contention within the TDT research community has been the definition of "event." Within TDT, topics are triggered by a "seminal event" and then contain all stories that discuss that event and any that follow inevitably from it [8]. To make that definition less vague, the topic annotation process includes "rules of interpretation" that describe how different types of topics (elections, scandals, wars,

crimes, etc.) should be scoped: what is and what is not part of the topic [5].

Unfortunately, even with those rules, it turns out that the selection of "seminal event" is not rigorously controlled. As a result, it is possible for a (non-seminal) event within topic T_1 to be seen as a seminal event of another topic, T_2, and for either $T_1 \subset T_2$ or $T_2 \subset T_1$[3]. That means that the "correct" topic—i.e., the "truth" against which a system would be judged—depends on the selection of seminal events, something that is not known to a system in advance.

For that reason (at least), the TDT 2004 evaluation has been extended to support hierarchical topics. To implement this, the topic annotation process has been changed so that it no longer discourages topics that occur within existing topics, and so that a much larger number of topics are annotated. Evaluation will be done in two ways for the TDT 2004 evaluation. The primary form will find the subgraph within the hierarchy that is the most similar (by the standard cost measure) to a "true" topic (based on LDC annotations). The cost will be increased by the "travel cost" necessary to move from the root of the graph to the selected subtree, so that structures that make it "easy" to locate the cluster will be rewarded. In the second approach, hierarchical clusters will be collapsed into a set of overlapping clusters and evaluated that way. As a simple example, if $(A, B) \subset C$ so A and B are contained within the larger cluster C, evaluation would consider the clusters A, B, and C. Note that this approach would not have been viable in the past because clusters A and B overlap with cluster C.

C. Interaction.

As another simplying assumption, the TDT research program has modeled its tasks as unsupervised, so they proceed without any human intervention. Our experiences with "garbage" clusters indicates that this model may be inadequate when deploying the system. If a user notices that a cluster contains nonsense, it seems odd not to permit the user to correct the error. Other tasks that are similar in spirit to TDT—notably that of information filtering [14]—have for many years incorporated a notion of supervised adaptation into their evaluation model. We have shown that such supervision within the TDT tracking task can result in a substantial improvement in accuracy [12].

Motivated by these observations, starting in TDT 2004, the tracking evaluation will incorporate supervision, allowing systems to request confirmation of some early decisions. The community opted not to incorporate supervision into the cluster detection task—the other changes already mentioned complicate issues enough—though it will eventually be valuable to include it there, too.

[3]It is even theoretically possible for neither to be a subset of the other, though that is unlikely given the rules of interpretation.

D. Inter-cluster connections.

A final deficiency of TDT that has become apparent because of our experiences is the failure to provide connections between an event within topics. This problem is somewhat related (1) to hierarchical topics since a critical type of link is the container/contains relationship that such a hierarchy suggests, and (2) to overlapping topics since an event can appear in multiple topics. But other types of links are important: ranging from relatively straightforward connections—related topic, same people, same location, same date—to needing substantial research—cause and effect.

There are no plans to incorporate connections between topics in TDT 2004. The issues are extraordinarily complex and it is not at all clear how an evaluation should be carried out. We have been investigating some of the possibilities that connections enable as a problem we call event threading [13]. In that task, we create smaller and more focused clusters that represent events rather than the larger topics of TDT (similar in spirit to some approaches to the detection task [9]). We also construct links between the event clusters that we hope represent time dependencies. Our thought is that a TDT topic would appear within such a graph structure as a set of inter-connected events. However, those same events would connect to events in other topics—those connections are not at all bound by rules of interpretation. Our early results are intriguing and suggest that event threading may be a new and interesting direction for TDT technology to move.

V. Conclusion

There has been almost eight years of research on the problem of TDT cluster detection. The six formal evaluations have demonstrated "clearly" what shape a state of the art system should take to address this task. However, a few months' experience with "real world" data and "real users" showed that evaluation clarity does not necessarily translate to deployment success.

Our experiences pointed out several serious limitations with the TDT evaluation program and led us to suggest several dramatic changes. Some of those—hierarchical clusters, overlapping clusters, and supervision of tracking—have been adopted by the TDT community and will be explored in TDT 2004.

What we learn in the new TDT evaluations will be transferred into the Event Organizer and IFE settings to see how well they address the problems we found. We

have completed one cycle of research to practice and back with valuable lessons to and from each: the research created core technology needed to deploy a system, and the deployment suggested ways that the research could be more pertinent. This project has been and continues to be a successful illustration of research and deployment synergy.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

REFERENCES

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of Conference on Information Retrieval Research (SIGIR)*, pages 37–45, 1998.

[3] James Allan. Introduction to topic detection and tracking. In James Allan, editor, *Topic Detection and Tracking: Event-based Information Organization*, pages 1–16. Kluwer Academic Publishers, Boston, 2002.

[4] Thorsten Brants and Francine Chen. A system for new event detection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 330–337. ACM Press, 2003.

[5] Christopher Cieri, Stephanie Strassel, David Graff, Nii Martey, Kara Rennert, and Mark Liberman. Corpora for topic detection and tracking. In James Allan, editor, *Topic Detection and Tracking: Event-based Information Organization*, pages 33–66. Kluwer Academic Publishers, Boston, 2002.

[6] S. Dharanipragada, M. Franz, J. S. McCarley, T. Ward, and W.-J. Zhu. Segmentation and detection at ibm. In James Allan, editor, *Topic Detection and Tracking – Event-based Information Organization*, pages 135–148. Kluwer Academic Publisher, 2002.

[7] David Eichmann and Padmini Srinivasan. A cluster-based approach to broadcast news. In James Allan, editor, *Topic Detection and Tracking – Event-based Information Organization*, pages 149–174. Kluwer Academic Publisher, 2002.

[8] Jonathan G. Fiscus and George R. Doddington. Topic detection and tracking evaluation overview. In James Allan, editor, *Topic Detection and Tracking: Event-based Information Organization*, pages 17–31. Kluwer Academic Publishers, Boston, 2002.

[9] Martin Franz, Todd Ward, J. Scott McCarley, and Wei-Jing Zhu. Unsupervised and supervised clustering for topic tracking. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 310–317. ACM Press, 2001.

[10] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *Proceedings of ACM SIGIR*, 2004. Forthcoming.

[11] Tim Leek, Richard Schwartz, and Srinivasa Sista. Probabilistic approaches to topic detection and tracking. In James Allan, editor, *Topic Detection and Tracking – Event-based Information Organization*, pages 67–84. Kluwer Academic Publisher, 2002.

[12] A. Leuski and J. Allan. Improving realism of topic tracking evaluation. In *Proceedings of ACM SIGIR*, pages 89–96, 2002.

[13] R. Nallapati, A. Feng, F. Peng, and J. Allan. Event threading within news topics. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2004. Forthcoming.

[14] S. Robertson and I. Soboroff. The TREC 2002 filtering track report. In *Proceedings of the Text Retrieval Conference (TREC-2002)*, 2003. NIST special publication 500-251.

[15] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *Proceedings of the Text Retrieval Conference (TREC-3)*. NIST, 1995.

[16] Yiming Yang, Jaime Carbonell, Ralf Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems Special Issue on Applications of Intelligent Information Retrieval*, 14(4):32–43, 1999.