Discovering Event Evolution Graphs From News Corpora

Christopher C. Yang, Xiaodong Shi, and Chih-Ping Wei

***Abstract*—Given the advance of Internet technologies, we can now easily extract hundreds or thousands of news stories of any ongoing incidents from newswires such as CNN.com, but the volume of information is too large for us to capture the blueprint. Information retrieval techniques such as Topic Detection and Tracking are able to organize news stories as events, in a flat hierarchical structure, within a topic. However, they are incapable of presenting the complex evolution relationships between the events. We are interested to learn not only what the major events are but also how they develop within the topic. It is beneficial to identify the seminal events, the intermediary and ending events, and the evolution of these events. In this paper, we propose to utilize the event timestamp, event content similarity, temporal proximity, and document distributional proximity to model the event evolution relationships between events in an incident. An event evolution graph is constructed to present the underlying structure of events for efficient browsing and extracting of information. Case study and experiments are presented to illustrate and show the performance of our proposed technique. It is found that our proposed technique outperforms the baseline technique and other comparable techniques in previous work.**

***Index Terms*—Event evolution, event evolution graph, graph pruning, topic detection and tracking (TDT).**

I. Introduction

DUE to the popularity of the Internet, most news stories have electronic versions published on newswires such as CNN, BBC, CBS, etc. Infomediaries such as Google and Yahoo are available to search across multiple newswires to retrieve news stories of any ongoing incidents. Retrieving news of the same topic from multiple sources and keeping information updated becomes more convenient and easier. However, it also generates tremendous volume of news text streams. Managing, interpreting, and analyzing such a huge volume of information is a difficult task. Techniques that are capable of extracting the underlying structure of the news events are desired. They are helpful for users to understand the evolution of events on the same topic.

Manuscript received May 28, 2007; revised November 14, 2008. First published May 5, 2009; current version published June 19, 2009. This paper was recommended by Associate Editor R. Popp.

C. C. Yang is with the College of Information Science and Technology, Drexel University, Philadelphia, PA 19104 USA, and also with the Digital Library Laboratory, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: chris.yang@ischool.drexel.edu).

X. Shi is with the Digital Library Laboratory, The Chinese University of Hong Kong, Shatin, Hong Kong.

C.-P. Wei is with the Institute of Service Science, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: cpwei@mx.nthu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2009.2015885

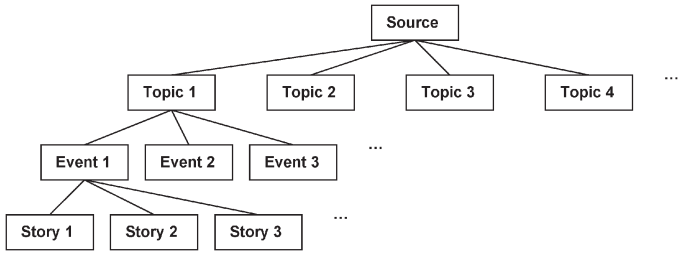


Fig. 1. Document organization in TDT.

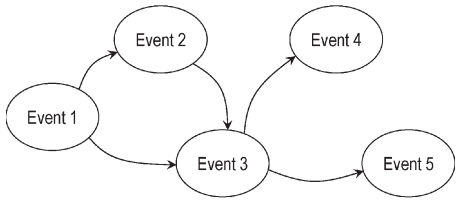


Fig. 2. Sample of event evolution graph.

In the research of Topic Detection and Tracking (TDT), different techniques have been developed to monitor news stories, spot news events, and track the progress of previously spotted events [3]–[5], [22]–[26]. The TDT research clusters news stories as a hierarchical structure, as shown in Fig. 1. An event is something that happens at some specific time. A topic is a set of events strongly interconnected with each other. Although users are able to capture the major events in an incident, it is difficult to capture the development of events, i.e., associations between them, within an incident. For example, the outbreak of a civil war may evolve to the economic and social crisis in the relevant country and then evolve to the problem of refugees. There may be some other events in the same topic, e.g., the state army won a big combat against the rebels, which are obviously not strongly associated with the last two events. TDT techniques have been attempting to detecting or clustering news stories into these events, without defining or interpreting the associations between these events. To present the development process of incidents, we must model this kind of associations between events, which we define as event evolutions.

The objective of this paper is to develop a technique that can identify the evolution relationships between the news events within a news topic. These relationships denote how events evolve or develop from the beginning to the end of the specific news affair. The events together with their relationships are then used to build a well-defined structure, *event evolution graph* (as shown in Fig. 2), which presents the blueprint of a news topic. Such event evolution graphs show the sophisticated event

Topic: Arafat's Death

Event 1: Arafat was allowed by Israel to leave compound for hospital
Event 2: Arafat entered Paris hospital and underwent medical test
Event 3: Rumors about Arafat's health and Palestine's future emerged
Event 4: Palestine police planned security in case of Arafat's death
Event 5: Palestine officials visited Arafat in Paris hospital
Event 6: Arafat was in deep coma and finally passed away
Event 7: Agreement on Arafat's funeral was reached and preparation was started
Event 8: Reactions on Arafat's death from different parties
Event 9: Abbas was elected PLO leader as Arafat's successor
Event 10: Arafat's funeral was conducted and people around the world mourned him
Event 11: Arafat's nephew checked his medical records and clarified the conspiracy behind his death

Fig. 3. List of events under the topic "Arafat's Death."

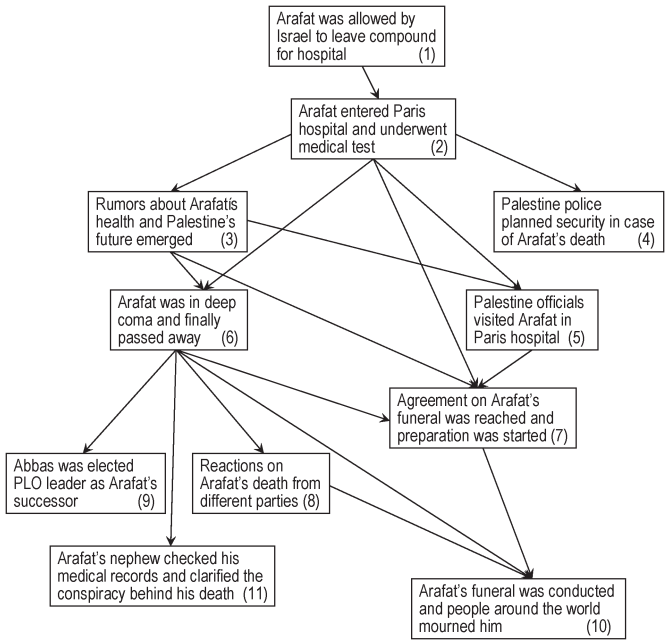


Fig. 4. Event evolution graph for the topic "Arafat's death." The numbers in the brackets indicate the temporal orderings of events represented by the rectangles. The edges between two rectangles indicate the event evolution relationships between their underlying events.

interrelationships in a graphical structure for easy navigation and browsing. Accordingly, users are able to capture the major events and understand the flow of the stories within the incident.

Considering the news topic of "Arafat's Death" as an illustration, Fig. 3 shows a list of 11 events in this topic tracked by typical TDT techniques without presenting how these events evolve as an event evolution graph. One can browse through the list and understand what events have happened according to the temporal order, but it is not easy to capture the development of events. As the number of events increases within a topic, it becomes more difficult to recognize the flow of events.

Fig. 4 shows the event evolution graph of the news topic of "Arafat's death." There are totally 11 events and 17 event evolution relationships. For example, there is an event evolution relationship from "Arafat entered Paris hospital and underwent medical test" (i.e., Event 2) to "Arafat was in deep coma and finally passed away" (i.e., Event 6). Following the event evolution relationships, we will find a terminal event "Arafat's funeral was conducted and people around the world mourned him" (i.e., Event 10). There are many possible paths from the starting event to the terminal event through event evolution relationships. Some paths are shorter and only highlight the most significant events. Some events are side events, such as "Palestine police planned security in case of Arafat's death" (i.e., Event 4). It does not evolve to other events. Through the event evolution graph, we can easily understand how the stories are developing along the timeline. It also presents a graphical summary of a large number of documents that are reporting the events in a news topic.

Event evolution graphs can be utilized in many other applications. We may integrate the event evolution graphs with automatic summarization and named entity recognition techniques to provide a well-equipped Web news infomediary agency. The automatic summarization provides a summary for each event (i.e., each node in an event evolution graph) and presents the transition from one event to another event (i.e., illustrating the event evolution relationship). The named entity recognition techniques extract the names of persons or organizations and locations involved in an event. As a result, users can easily track the persons, organizations, or locations of the events along a particular path in an event evolution graph to see whether a new person has involved or the location has changed along the event evolution. The interconnectivity of the graph structure can also support the construction of a convenient information-browsing platform for users to navigate through the development of that news affair. Besides, we can apply advanced graph algorithms on event evolution graphs to extract many interesting patterns. For examples, we can detect subgraphs by graph segmentation techniques. These subgraphs may correspond to a set of events that does not follow the main line of the topic development.

The rest of this paper is organized as follows. In Section II, we discuss some related work. In Section III, we formally define the problem and related concepts of event evolution. In Section IV, we describe our technique for modeling event evolution relationships and building event evolution graphs. Three graph pruning methods are proposed and presented in Section IV-C for generating event evolution graphs. Section V presents our evaluation measures. We will use a real-world example to illustrate event evolution and event evolution graph in Section VI. In Section VII, we detail our experiments and discuss important experimental results. In Section VIII, we summarize this paper and provide some future directions of this paper.

II. Related Work

TDT is an active research area in recent years. The objective of TDT is to organize news documents given a stream of constantly generated new stories coming from a wide range of resources such as text or audio in multiple languages. The tasks in TDT include story segmentation, topic detection, new event detection, link detection, and topic tracking [3]. The aim of story segmentation is to determine the boundaries for cohesive text fragments. Story segmentation is needed for stories from audio sources but not required for newswire sources [1]. Topic detection groups stories of the same topic together. New event detection makes binary decision on each new document whether it discusses a new topic that has not been reported yet. Link detection determines if two documents are of the same topic. Topic tracking tracks the topic to find all following stories.

There have been several techniques on detecting news topics and tracking new stories for a news topic. For example, Allan *et al.* [5] considered each incoming news document as a query that was made on the previous clustered documents to determine if the incoming news document is similar to any of the clustered documents. A story was considered as a first story if no similar documents can be found. Terms were weighted by *TF-IDF* and *surprisingness*. Inverted indexing was utilized for efficiency. Yang *et al.* [22] employed the group-average clustering and the single-pass clustering for topic detection. Yang *et al.* [24], [25] have also investigated a multistrategy approach that combines Rocchio, kNN, and language modeling for topic tracking. Carthy [8], Allan *et al.* [5], and Yang *et al.* [26] used the natural language processing approach by combining lexical chains with keywords for topic tracking and extracted seven types of name entities. Chen *et al.* [9] developed an aging theory to model the life cycle of events by considering the temporal relationships among documents. Xu *et al.* [19] extended the TDT technique to detect semantic event in video by using the webcast text.

TDT generally focuses on how to detect topics and novel events as well as how to cluster or categorize news stories into different topics (or events). News stories are usually organized into a flat hierarchical structure in which news stories discussing the same topics (or events) are grouped into clusters. Prior research paid less attention on interpreting or modeling the interrelationships or associations between different topics (or events). However, determining such interrelationships is desirable particularly at the granularity of news events since users are often concerned with the connections between news events within the same topic. Modeling such connections between events is generally out of the scope of the TDT research, and thus becomes a novel problem, i.e., event evolution. Some researchers have investigated the evolving data streams [1], [27], but it is only limited to data streams such as sensor network data and stock price data. It is not the evolution of events as discussed in this paper.

Event evolution is a new concept developed recently. Makkonen [14] was the first to conduct investigations on event evolution as a subtopic of TDT. The news documents within a topic are temporally linearly ordered. A narrative begins when the first story of the topic is detected. A seminal event may lead to several other events. The events at the beginning may have more influence on the events coming immediately after than the events at the later time. As we go through the event in the temporal order, we may see the evolution of events within an incident. The events and the event evolution relationship can be represented as a graph structure (Fig. 2). Makkonen [14] claimed to use the ontologies, including general terms, locations, names, and temporals, to measure the similarity of events. However, it did not define the concept of event evolution clearly and elaborate the structure of event evolution graph. Besides, it is lack of details in the event evolution model and fails to present any experimental evaluation results.

Mei and Zhai [15] approached the problem as theme evolution rather than event evolution. Pertaining to the area of temporal text mining, their study proposed a temporal pattern discovery technique from temporally ordered textual documents on the basis of the timestamps of the text streams. Specifically, text steams are partitioned into a number of either overlapped or nonoverlapped sliced time intervals. The theme of each interval, which is a set of key terms, is identified, and the evolution of theme between successive intervals is extracted. However, their study does not identify the events of incident. Moreover, an event in an incident may take more than one interval or only part of an interval. As a result, the theme extracted from an interval may be part of an event or a combination of several events that occur in the interval. Besides, they aim at mining temporal evolutionary patterns of key terms from texts only, without considering the underlying news events embedded in these news texts. Therefore, this technique is not capable of capturing the transitional flow of major events (i.e., event evolution relationships) in an incident topic.

Wei and Chang [18] proposed an event evolution pattern discovery technique that identifies event episodes together with their temporal relationships that occur frequently in a collection of events of the same type. An event episode is a stage or subevent of an event. Their study focuses on segmenting a sequence of news stories of a specific event into event episodes, generalizing event episodes across different events of the same nature, and extracting common event episodes and their relationships. Their study differs from this paper in that the two studies operate at two different granularities: their study deals with an event and their event episodes, whereas this paper handles a topic and their events. Furthermore, their study considers temporal relationships (i.e., a relationship exists from one event episode to another if the ending time of the former event episode is no later than the starting time of the latter event episode) rather than evolution relationships, which is the focus of this paper. The temporal relationships in [18] organize event episodes in sequences according to their temporal order, but the temporal relationships do not necessarily reflect evolutions between event episodes. In evolution relationships, the temporal order is considered, but we also consider event content similarity, temporal proximity, and document distributional proximity.

Nallapati *et al.* [16] later defined the concept of event threading, given a small number of documents and events in a news topic. Their definition of event threading is close to event evolution except that they consider event threading as a

tree structure rather than a graph. To identify event threading, they employed a simple similarity measure between documents to cluster documents into events and the average document similarity to estimate dependencies between events. As with the finding reported by Makkonen [14], location and named entity features were not effective in clustering stories into events. His paper represents a good starting point to investigate the event evolution problem. However, the concept of event threading is indeed only part of the concept of event evolution, and their proposed technique cannot cope with the complexity in event evolution relationships. When the number of news documents within a news topic is large, the event evolution graph is a complex graph rather than a tree structure of event threading. In addition, their dependence modeling methods only consider the average document similarity between events. Such methods are not effective and sufficient to identifying the event evolution relationships. In this paper, we propose the similarity of event term vectors for identifying the event evolution relationships and find that it outperforms the average document similarity method proposed by Nallapatil *et al.* [16].

This paper is also related to automatic summarization [6], [21], an important topic in computational linguistics. Automatic summarization techniques provide a briefing of a certain document or topic (consisting of multiple documents) by either extracting representative sentences or abstracting the semantic meanings of the paragraphs in the texts. Automatic summarization gives a blueprint of the entire topic and reduces the information overheads to users. However, it neglects the interrelationships between news events within the same news topic and exhibits only a linear structure of the topic.

In view of the limitations of the related work in modeling the evolution of events from the news text streams, we formally define the concept of event evolution and investigate different approaches to model event evolution relationships between events within a news topic. We propose to use event content similarity between events and consider two decaying factors that include temporal proximity and document distributional proximity in the event evolution scoring function. We introduce the concept of event timestamp in measuring the temporal proximities. The complex structure exhibiting event evolutions between news events is abstracted as a graph structure, i.e., event evolution graph. Several graph-pruning methods are proposed for improving the effectiveness of resultant event evolution graphs. Our experimental results show that our proposed technique outperforms the rival techniques as well as the baseline.

III. Event Evolution

A. Story, Event, and Topic

In this section, we present the definitions of story, event, and topic, which are important for us to define the concept of event evolution later. The definitions of the three terminologies adhere mostly to the formal definition given in TDT [1], [4], [16].

Story: A story is a news article delivering some information to users, i.e., reporting a certain event. It is assumed that each news document describes a unique story and each story discusses a single (but unnecessarily unique) event. Thus, a story is the smallest atomic unit in the topic/event/story hierarchy [16].

Event: An event is something that happens at some specific time, and often some specific place. Our definition of event relaxes TDT's definition by removing the constraint that an event must always occur at "some specific place." It is because some events actually do not happen at any specific location in the world. For example, the event of "reactions from world leaders on Arafat's death" includes stories describing the reactions from political leaders all over the world. In this sense, we cannot assign any location stamp to this event to indicate exactly where it happens. Instead, we can always identify the time (or range of time) when a given news event takes place and, thus, assign a timestamp to the event.

Similar to [16], we represent the event content, i.e., what happens in an event, by a set of stories that discuss it. Following the assumption of atomicity of a story, any set of distinct events can be represented by a set of nonoverlapping clusters of news stories.

Topic: A set of events strongly interconnected with each other. Usually, a topic starts with a novel seminal event, followed by other related events. By definition, each event contains a set of stories, whereas a topic consists of a collection of events. Later we focus on constructing event evolution graphs for single topics.

The granularities of our defined event and topic are slightly different from those in TDT definitions, but they are the same as the previous work on event evolution [14], [16]. In TDT definitions [1]–[4], events refer to a large incident. For example, a terror attack incident includes several subevents such as explosions, rescues, investigations, prosecution of suspects, and so on; topics refer to a collection of incidents similar in their natures, e.g., terrorism, earthquake, etc. In this paper, events refer to those subevents previously mentioned, and topics refer to a large and complete incident with its full development process.

B. Event Evolution

Event Evolution is defined as *the transitional development process of related events within the same topic*. Obviously, for a complete topic, the event evolution should start from the beginning, i.e., the seminal event, and finally evolve to the ending event(s). As one event may evolve/develop to several other events, or several events may jointly evolve/develop to another event, there are different chains of developing events in the event evolution. The component of these chains, i.e., the development process from one event to another, is called an event evolution relationship. We formally define an event evolution relationship as *the directional logical dependencies or relatedness between two events.* If the occurrence of event B depends on the occurrence of event A, then there is an event evolution relationship from event A to event B. The word "evolution" here implies the transitional process of one event developing to another along the timeline. The word "directional" emphasizes that an event evolution relationship is not bidirectional and cannot be reversed. For example, if there is an event evolution relationship from the event "Columbia space shuttle crashed" to the event "NASA conducted investigations into the shuttle crash," we cannot deduce that there is event evolution from the latter event to the former one.

If we analyze carefully the definitions of event evolution and event evolution relationships, it is not difficult to conclude two observations. First, event evolution relationships indicate the development from one event to another, and therefore the event from which the second one is developed MUST occur before the second one. Second, event evolution relationships do not always stand for any pair of events that satisfies the first observation. That is, two events involved in an event evolution relationship MUST have logical dependencies or relatedness. This observation may not seem easy to model. However, we observe that two events having an event evolution relationship are often relatively close to each other along the timeline. In addition, previous TDT research suggested that similar or related stories/events/topics often share a similar vocabulary and more importantly similar key terms [23]. We also observe that two events having an event evolution relationship, the author(s) of the news stories of the second event often tend to refer or even cite directly some segments of the stories of the first event. These all suggest the appropriateness of using statistical language models to capture event evolution relationships between events.

Based on the aforementioned observations, we describe three conditions to assess the validity of event evolution relationships. Specifically, we define that the event evolution relationship from event A to event B should follow the below conditions.

1) Event *A* must temporally precede event *B*.
2) Event *A* shares a similar vocabulary and set of key terms with event *B*.
3) The farther two events are from each other, the less likely they have an event evolution relationship.

These three conditions are elaborated from a logical point of view and, thus, serve as the criteria for modeling event evolution relationships. In Section IV, we also propose mathematical formulations and algorithms to materialize these conditions so that we can construct an event evolution graph where each event evolution relationship involves two events that are logically dependent or related.

C. Event Timestamp and Temporal Relationships

As previously mentioned, each event can be associated with a specific time, which can be either a spot time or duration of time. With the former one, the event occurs and then immediately finishes, e.g., the exact time when Arafat passed away. With the latter one, the event occurs at the beginning of that duration of time and then continues until the end of that period, e.g., the time when NASA conducted investigations into the causes why space shuttle Columbia crashed. In Nallapati *et al.*'s work [16], only time differences between stories were considered but event times and temporal distance between events were not discussed.

We define the time when an event takes place as the timestamp of that event, called *event timestamp*. Each event has a single but not necessarily unique event timestamp. Multiple events may have the same event timestamp, which implies that these events happen at exactly the same time. Event timestamp is implemented as *time interval* object in this paper. We define *time interval* as the duration of time between two spot times inclusively along a straight linear timeline, with one ahead of or equal to the other. We call the anterior spot time the *start time* of the time interval and the posterior spot time as the *end time* of the time interval. Let the *start time* and *end time* be st and en, respectively. A time interval t can be represented as $t = [st, en]$. If $st = en$, the time interval t is actually a spot time.

The distance between two time intervals, $t_1 = [st_1, en_1]$ and $t_2 = [st_2, en_2]$, is formally defined (assuming st_1 is before st_2) as follows:

$$d(t_1, t_2) = \begin{cases} en_1 - st_2 & \text{if } en_1 \text{ is before or equal to } st_2 \\ 0 & \text{else.} \end{cases} \tag{1}$$

From the above equation, we can see that the distance between two time intervals is the distance between the end time of the anterior time interval and the start time of the posterior time interval. If the two time intervals overlap, then their distance is set to zero.

D. Event Evolution Graph

An event evolution graph is a directed acyclic graph (DAG) consisting of events as the nodes and event evolution relationships as the directed edges between nodes. Given a set of n distinct news stories $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$ on a given news topic, we have a set of m events $\mathbf{E} = \{e_1, e_2, \ldots, e_m\}$ and their event timestamps $T = \{t_1, t_2, \ldots, t_m\}$. We can build a one-to-one mapping function τ that maps each event to its corresponding event timestamp

$$\tau(e_k) = t_k (1 \le k \le m). \tag{2}$$

For each of the n stories, we assign it to one of the m events, either manually or automatically with the aid of document clustering techniques

$$\forall s_i \ \exists e_k \in \mathbf{E}, \ \ s_i \in e_k \qquad (1 \le i \le n, 1 \le k \le m) \tag{3}$$

$$\forall i, j \ \ i \ne j, \ \ e_i \cap e_j = \varnothing \qquad (1 \le i \le m, 1 \le j \le m) \tag{4}$$

$$\forall i, \ \ e_i \cap \mathbf{S} \ne \varnothing \qquad (1 \le i \le m). \tag{5}$$

Equations (3) and (4) state that every story belongs to exactly one of the m events. Equation (5) denotes that each event is nonempty and has at least one associated story. We can also build a many-to-one mapping function f that associates a story with any of the m events

$$f(s_i) = e_k \quad \text{iff} \quad s_i \in e_k \quad (1 \le i \le n, 1 \le k \le m). \tag{6}$$

We denote the event evolution relationship from event e_i to event e_j (where $i \ne j$), if existing and valid, as (e_i, e_j). A directed edge from vertex e_i to e_j is created in the event evolution graph accordingly. In this case, we call event e_i the parent of event e_j and e_j the child of e_i. Later, we will use e_i for the event and the vertex in the graph interchangeably. Similarly, we will use (e_i, e_j) for the event evolution relationship and the directed edge interchangeably.

After identifying all valid event evolution relationships between m events, we have a set of directed edges $\mathbf{L} = \{(e_i, e_j)\}$

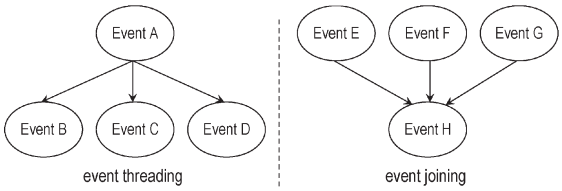


Fig. 5. Event threading and event joining.

where $e_i, e_j \in \mathbf{E}$. Accordingly, we construct an *event evolution graph* $\mathbf{G} = \{\mathbf{E}, \mathbf{L}\}$ as a DAG that contains all the events (i.e., $\mathbf{E}$) within the same news topic and all valid event evolution relationships between these events (i.e., $\mathbf{L}$). Our research question is then formulated as how to mine the event evolution graph effectively from a large set of news stories within the same news topic.

Structurally, an event evolution graph is similar to a cognitive map, which consists of causal concepts as nodes and causal connections as directed edges between nodes [11], [12]. Each causal connection is associated with a strength (e.g., a positive "+" or negative "−" sign, or a value in the interval $[-1, 1]$) that specifies the causal value of the causal connection. Cognitive maps assist individuals to structure problems by representing an individual's perception and understanding of a target problem in a graphical structure [11], [12]. Our proposed event evolution graph uses a graphical representation similar to cognitive maps and intends to show the sophisticated event interrelationships of a news topic. Such event evolution graphs can facilitate users' understanding of the flow of the news stories and navigating and browsing concerning news stories within a news topic.

E. Event Threading and Event Joining

As shown in Fig. 5, we can abstract two patterns of event evolution from an event evolution graph, i.e., event threading and event joining. *Event threading*[1] is a subgraph in which one event (i.e., a node whose outdegree is greater than one) evolves to several other events directly. *Event joining* is a subgraph in which several events evolve directly to a single event (i.e., a node whose indegree is greater than one). Event threading implies that the parent event causes many effects or evolves into several branch events. On the other hand, event joining implies that several parent events jointly cause one effect or evolve into a subsequent event. Event threading is obviously a simple tree structure, while event joining is an inverted tree structure. We define the number of edges in event threading or event joining as its *degree*.

The existing event models presented by Makkonen [14] and Nallapati *et al.* [16] consider only event threadings because the news topics investigated in their studies are mostly simple with only about five events per topic on average. However, in this paper, we consider news topics with an average of 78 stories per topic and 18 events per topic. Therefore, our event evolution graphs are more complicated and usually include both event threadings and event joinings. Complicated event evolution graphs embedding considerable event threading and event joining patterns usually exhibit abundant valuable information in the event structure of certain topics.

The degree of event threading and event joining is an important factor in constructing event evolution graphs. Obviously, a higher degree means that the graph is complicated and more detailed event evolutions are presented to users. Contrarily, if the degree is small, the resultant graph is simple and fewer information are brought to users (but probably more information that is useful may be missing). In Section IV, we propose different methods to tune the degree of event threading and event joining, and thus balance between event evolution detail and information overload.

[1]Here, the term *event threading* is the same as the one used in Nallapati *et al.*'s paper [16]. We consider *event threading* as a simple one-layer tree structure, just like e-mail threading. However, Nallapati *et al.* did not impose formally such restrictions on their "event threading" model even though they used tree structures when demonstrating their models.

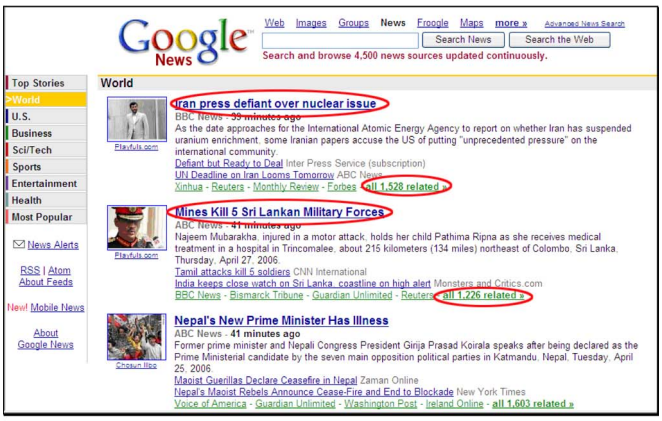
IV. Constructing Event Evolution Graphs

In this section, we present our proposed technique for constructing an event evolution graph for a given news corpus of the same topic. Our technique can be decomposed into three steps: 1) generating representations of events from news stories, 2) modeling event evolution relationships between these events, and 3) pruning edges that correspond to invalid or weak event evolution relationships.

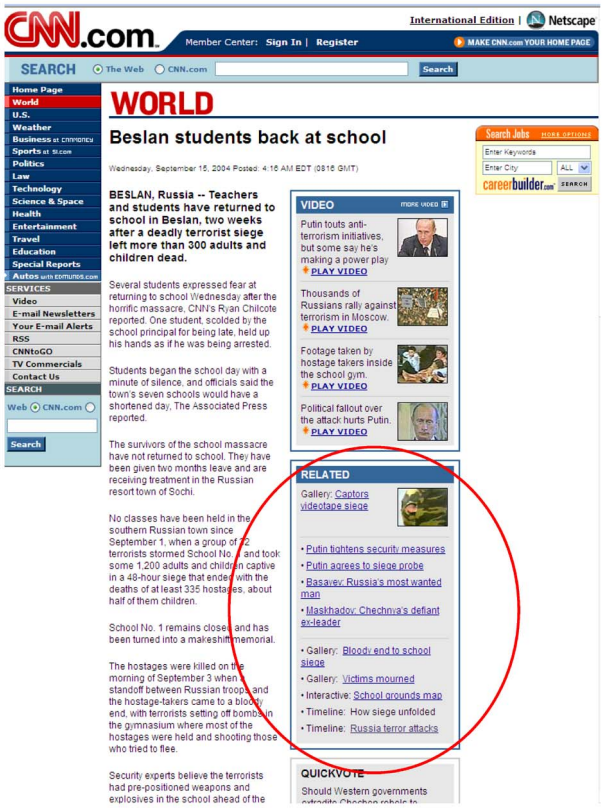
A. Generation of Events

The task of identifying and generating events from news stories has been well addressed by the subtask of event detection and tracking in TDT. Different techniques have been developed to cluster or categorize news stories into events. Prior studies commonly employ k-means [10] or hierarchical document clustering algorithms [17] for this purpose. For example, Yang *et al.* [24] extended the hierarchical agglomerative clustering algorithm and proposed an augmented Group Average Clustering to cluster news stories into events in their event detection research. Recently, Li *et al.* [13] proposed a probabilistic model for retrospective news event detection, which can also be employed to generate events from news stories and assign news stories into news events previously identified.

As we are not aiming at improving existing event detection and tracking techniques, we focus solely on the construction of event evolution graph using manually generated and annotated news events. Manually generated events can eliminate the biases that may be created by different event detection and tracking techniques and, hence, provide a best platform on which we can fairly compare different event evolution identification techniques. It should also be noted that there are some online sources of well-generated news events (either manually or automatically by undisclosed techniques), e.g., Google News as in Fig. 6(a). Google News aggregates news from multiple sources such as CNN, ABC, BBC, etc. It also traces related news for each news article. These sources, after necessary preprocessing, can serve as inputs to our proposed event evolution identification technique. In our experiment, as described in Section VII, we only used the related news extracted from CNN News Website [as shown in Fig. 6(b)] to avoid duplicated news from multiple sources.



(a)



(b)

Fig. 6. (a) Google News provides links to related news which can be used to cluster similar news stories into events (the red circles highlight the title of the news articles and the link to related news. (b) CNN News provides related news in each news article (the red circle highlights the links to related news).

Mathematically, we have a story-to-event mapping function $f : \mathbf{S} \rightarrow \mathbf{E}$ that is already manually configured, and we focus only on modeling $\mathbf{L}$. It is important to note that our technique is independent of f, which means that it can also be plugged into some automatic systems capable of clustering stories into events.

B. Modeling Event Evolution Relationships

In modeling event evolution relationships, we propose to utilize the term vectors, temporal distances between events, and distributions of news documents along the timeline as the fundamental features based on the observations discussed previously. In Section III, we have defined three conditions to evaluate the event evolution relationship from event A to B. The first condition implies that the temporal orders of events are very important in determining the validity of event evolution relationships. We define a symbol $\rightarrow$ to represent the assertion "(event A) temporally precedes (event B)." In contrast, we use $\not\rightarrow$ for the negative statement of $\rightarrow$. Let $\tau(e_i)$ denote the timestamp of event e_i. If e_i temporally precedes event e_j, $\tau(e_i)$ and $\tau(e_j)$ must possess the following temporal relationships as defined by Allen [7]: $\tau(e_i)$ before $\tau(e_j)$, $\tau(e_i)$ meets $\tau(e_j)$, $\tau(e_i)$ overlaps $\tau(e_j)$, $\tau(e_i)$ starts $\tau(e_j)$, $\tau(e_i)$ equal to $\tau(e_j)$, $\tau(e_i)$ contains $\tau(e_j)$, $\tau(e_i)$ started by $\tau(e_j)$, and $\tau(e_i)$ finished by $\tau(e_j)$. If e_i does not temporally precede e_j, we claim that there should not be any event evolution relationship from e_i to e_j. In this case, we set its event evolution score to be zero

$$\forall\, i,\ j \text{ if } i \neq j \qquad \tau(e_i) \not\rightarrow \tau(e_j), \text{score}\left((e_i, e_j)\right) = 0. \quad (7)$$

The second condition implies that, if an event evolution relationship exists from one event to another, the two events should share some common information in their content. Accordingly, we adopt the vector space model to measure the relatedness of events. A k-term vector for $\mathbf{S}$ is denoted as $\omega = \langle \omega_1, \omega_2, \ldots, \omega_k \rangle$. Let a story i be represented as a weighted term vector $\omega_i = \langle \omega_{i1}, \omega_{i2}, \ldots, \omega_{ik} \rangle$. On the basis of the traditional TF-IDF function, ω_{ik} is defined as

$$\omega_{ik} = \frac{tf_{ik}}{\max\limits_{l} tf_{il}} \log \frac{N}{df_k} \quad (8)$$

where tf_{ik} is the frequency of term k in the news document i, N is the total number of news documents in that topic, df_k is the number of news documents in that topic containing term k, and $\max\limits_{l} tf_{il}$ is the maximum term frequency for all terms in document i.

If we only consider the TF factor only, ω_{ik} becomes

$$\omega_{ik} = \frac{tf_{ik}}{\max\limits_{l} tf_{il}}. \quad (9)$$

In this paper, we investigate both TF-IDF and TF formulations because measuring the similarity between events based on their stories is not the same as measuring similarity between documents in a corpus in traditional information retrieval applications. The number of stories within a topic is usually much less than the number of documents in a corpus. Besides, the stories within a topic are all related to some extent. However, the content of documents within a corpus varies significantly across different topics.

We further define the event term vector of event e_j using the average of the term vectors of stories that belong to e_j as: $\omega'_j = \langle \omega'_{j1}, \omega'_{j2}, \ldots, \omega'_{jk} \rangle$, where

$$\omega'_{jk} = \frac{1}{n_j} \sum_{\forall s_i \in e_j} \omega_{ik} \quad (10)$$

where n_j is the number of stories in $\mathbf{S}$ that belongs to event e_j.

The event content similarity is then defined as follows:

$$\cos\_sim(e_i, e_j) = \frac{\sum_{x=1}^{k} \omega'_{ix}\omega'_{jx}}{\sqrt{\left[\sum_{x=1}^{k} \left(\omega'_{ix}\right)^2\right]\left[\sum_{x=1}^{k} \left(\omega'_{jx}\right)^2\right]}}. \quad (11)$$

In this case, we treat an event as a collection of stories and view the information content of that event as the average of the term vectors of its stories. Thus, the content similarity between two events is the cosine similarity of their event term vectors. It is worth to note that Nallapati *et al.*'s approach [16] used the average of the similarities of all pairs of stories between e_i and e_j to measure the similarity between the two events. In our experiment, we shall present the comparative evaluation results of using our event content similarity and Nallapati *et al.*'s average pairwise similarity of stories between two events.

Other than the temporal orders, the temporal distance between two events is also helpful in estimating the likelihood of event evolution between the two events, as stated in the third condition in Section III. Obviously, if two events are far away from each other along the timeline, the event evolution is less likely to exist between them. On the contrary, if two events are temporally close, an event evolution between them is more likely to exist. In this paper, we use the *temporal proximity* between events to measure their relative temporal distance between two events, defined as the following decaying function:

$$tp(e_1, e_2) = \mathrm{e}^{-\alpha\left[\frac{d(\tau(e_1),\tau(e_2))}{T}\right]} \quad (12)$$

where T is the *event horizon* defined as the temporal distance between the start time of the earliest event timestamp and the end time of the latest event timestamp in the same topic. α is the time decaying factor which is between zero and one.

Temporal proximity can only capture the diminishing of event evolution scores along the event horizon. However, it may not perform well in some other cases. For example, for many news topics reporting sudden seminal events, there is usually a burst of number of events and stories in the beginning stage. Even though these events are temporally localized within a relatively short period, their differences in nature are still significant. Therefore, we utilize the distribution of documents in order to counterwork the shortcomings of temporal proximity. Specifically, we define the *document distributional proximity* as the second decaying function

$$df(e_1, e_2) = e^{-\beta\frac{m}{N}} \quad (13)$$

where m is the number of documents that belong to the events happening in-between event e_1 and e_2. N is the total number of documents in the topic. The document distributional proximity is similar to temporal proximity except that it substitutes the time with the distribution of documents.

Having defined the event content similarity, temporal proximity, and document distributional proximity, we formally propose our event evolution scoring function to estimate the likelihood that an event evolution relationship exists from an event e_1 to another event e_2 as

$$\text{score}\left((e_1, e_2)\right) = \begin{cases} 0 & \text{if } \tau(e_1) \not\rightarrow \tau(e_2) \\ \cos\_sim(e_1, e_2) & \\ \times tp(e_1, e_2) \times df(e_1, e_2) & \text{if } \tau(e_1) \rightarrow \tau(e_2). \end{cases} \quad (14)$$

C. Pruning Event Evolution Graphs

Given a complete graph with directed edges for each pair of events, a pruning method removes those directed edges corresponding to invalid or weak event evolution relationships and generates an event evolution graph. In this section, we propose three pruning methods, namely, static thresholding, static pruning, and dynamic pruning. The pruning of directed edges depends on the value of event evolution scores and/or the degree of event threading and event joining. A low value of event evolution score implies that the event evolution relationship between the corresponding events is weak, and therefore the corresponding edge should not be retained. In some cases, there are several edges entering or leaving an event. In reality, an event typically only evolves to a few events or is evolved from a few events. We use an upper bound to limit the number of edges entering or leaving an event.

1) Static Thresholding: This method uses a static threshold λ (where $0 < \lambda < 1$) for pruning. Specifically, if the event evolution score of a candidate event evolution relationship is lower than λ, we consider the candidate an invalid event evolution relationship and thus eliminate the candidate; otherwise, we retain the candidate. Formally, we represent the static thresholding method as

$$\mathrm{G} = (\mathrm{E}, \mathrm{L}')$$

where

$$L' = \left\{(e_i, e_j) | \text{score}\left((e_i, e_j)\right) \geq \lambda\right\}. \quad (15)$$

2) Static Pruning: Besides the use of a static threshold λ, the static pruning method also sets upper bounds for the degree of event threading and event joining in the target event evolution graph. This is equivalent to setting the maximum number of outgoing or incoming edges allowed per vertex. If the degree of event joining exceeds the upper bound N_i, we sort the incoming edges in a descending order according to their event evolution scores and retain only the first N_i incoming edges. For the degree of event threading, we define an upper bound N_o and perform the same pruning process as that for event joining.

We first define a function $g((e_i, e_j), e_i)$ that maps the event evolution relationship (e_i, e_j) to an index μ_i of its position in the sorted list of all event evolution relationships with e_i as their parent event in descending order according to their event evolution scores, i.e.,

$$g\left((e_i, e_j), e_i\right) = \mu_i. \quad (16)$$

For example, if score$((e_i, e_j))$ is the third largest among all the event evolution relationships with e_i as its parent event

(i.e., the outgoing edges from vertex e_i), then $\mu_i = 3$. Similarly, we also define $g((e_i, e_j), e_j)$ for (e_i, e_j) and e_j as the index μ_j of its position in the sorted list of all event evolution relationships with e_j as their child event in descending order according to their event evolution scores, i.e.,

$$g\left((e_i, e_j), e_j\right) = \mu_j. \tag{17}$$

Formally, the static pruning method generates an event evolution graph as

$$\mathrm{G} = (\mathrm{E}, \mathrm{L}')$$

where

$$L' = \{(e_i, e_j) | \left[\text{score}\left((e_i, e_j)\right) \geq \lambda\right] \cap \left[g\left((e_i, e_j), e_i\right) \leq N_o\right] \cap \left[g\left((e_i, e_j), e_j\right) \leq N_i\right]\}. \tag{18}$$

3) Dynamic Pruning: This method is similar to the static pruning method except that we set the upper bounds for the degree of event joining and event threading dynamically instead of constantly. Let N_v be the number of nodes (i.e., events) in the event evolution graph. A fully connected undirected graph will have $N_v(N_v - 1)/2$ possible edges, leading to $(N_v - 1)/2$ edges per vertex. We consider the upper bounds as a function of that number. Specifically, we define the upper bound for the degree of event joining for an event evolution graph with N_v events as

$$N_i = \theta_i \frac{N_v - 1}{2}. \tag{19}$$

Similarly, we define the upper bound for the degree of event threading for the event evolution graph as

$$N_o = \theta_o \frac{N_v - 1}{2}. \tag{20}$$

After determining the two upper bounds, we perform the same procedure as that in the static pruning method. Formally, the dynamic pruning method refines an event evolution graph as

$$\mathrm{G} = (\mathrm{E}, \mathrm{L}')$$

where

$$L' = \{(e_i, e_j) | \left[\text{score}\left((e_i, e_j)\right) \geq \lambda\right] \cap \left[g\left((e_i, e_j), e_i\right) \leq N_o\right] \cap \left[g\left((e_i, e_j), e_j\right) \leq N_i\right]\}. \tag{21}$$

V. Evaluation Measures

Assuming the manually created event evolution graph $\mathbf{G} = (\mathbf{E}, \mathbf{L})$ is the true structure, the automatic system using certain algorithms generates an event evolution graph $\mathbf{G}' = (\mathbf{E}, \mathbf{L}')$. Since $\mathbf{E}$ is predefined in this paper, our evaluation focuses on the differences between the sets of event evolution relationships $\mathbf{L}$ and $\mathbf{L}'$.

We adopt the measurement of precision and recall for our evaluation. Precision and recall measurements are widely applied in many information retrieval tasks where both the size of retrieved answer set and the number of correctly retrieved results are important to users. Obviously, users expect to have as many correct/relevant event evolution relationships identified as possible in the generated event evolution graph, but usually it is at the cost of fast increasing volume of returned results. In that case, users need to spend many time on distinguishing between correct/relevant event evolution relationships and incorrect/irrelevant ones, given a very large answer set. Thus, we need to measure both the number of correct/relevant event evolution relationships retrieved and the total size of returned results and allow users to make balance between them.

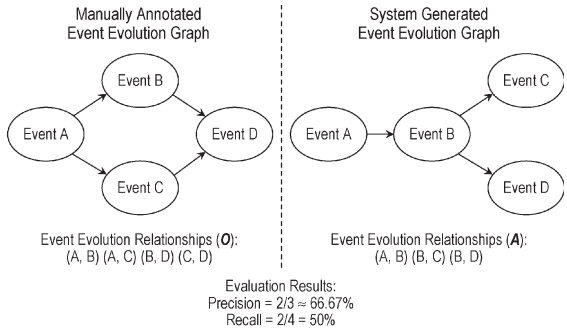


Fig. 7. Example of evaluation measures using precision and recall.

Assuming the manually annotated set of event evolution relationships as the truth set O and the system generated by certain algorithms as A; $O = \mathbf{L}$ and $A = \mathbf{L}'$. Therefore, the retrieved relevant set C is the overlapping part of O and A

$$C = O \cap A = \mathbf{L} \cap \mathbf{L}'. \tag{22}$$

1) **Precision (P):** It is the ratio of the number of true and valid event evolution relationships retrieved by the automatic system to the total number of event evolution relationships retrieved by the automatic system.

$$P = \frac{|C|}{|A|} = \frac{|O \cap A|}{|A|} = \frac{|L \cap L'|}{|L'|}. \tag{23}$$

2) **Recall (R):** It is the ratio of the number of true and valid event evolution relationships retrieved by the automatic system to the total number of true and valid event evolution relationships annotated manually.

$$R = \frac{|C|}{|O|} = \frac{|O \cap A|}{|O|} = \frac{|L \cap L'|}{|L|}. \tag{24}$$

Fig. 7 shows the computation of precision and recall rates for a sample event evolution graph. Obviously, we can tune the input parameters of our models and then generate different sets of precision and recall rates. Theoretically, if the tuning of model parameters is continuous, we can plot a smooth precision and recall curve.

VI. Case Study

In this section, we illustrate the event evolution graph of a terror attack incident, Chechen Terrorist Seizing Beslan School. For this terror attack case, we collected from 32 CNN news documents organized into eight events. Table I shows the details of the eight events. We also include this set of data as one of the ten topics in our experiment presented in Section VII.

TABLE I
EVENTS IN "CHECHEN TERRORISTS SEIZING BESLAN SCHOOL" TERRORIST ATTACH CASE

Event	Event Summary	Num. of doc.	Start time	End time
1	Chechen terrorists seized the Beslan school with hostages, negotiated, freed some hostages	5	2004-09-02 01:46	2004-09-03 07:08
2	Special task force assaulted terrorists and hundreds of hostages were dead	3	2004-09-03 14:46	2004-09-05 05:14
3	Responses of different parties on the Beslan school hostage tragic	5	2004-09-04 15:45	2004-09-07 13:04
4	Russia approached to identify the suspects of Beslan tragedy	6	2004-09-06 01:07	2004-09-08 17:54
5	Russia conducted investigation and determined to put terrorists on trial	4	2004-09-08 15:44	2004-09-24 11:36
6	Beslan school resumed classes after the hostage tragic	3	2004-09-14 08:12	2004-09-15 12:33
7	Russia claimed to strike Chechen terrorism	3	2004-09-14 08:52	2004-09-17 12:38
8	Russia's successive efforts against terrorism	3	2004-09-29 12:01	2004-12-17 13:53

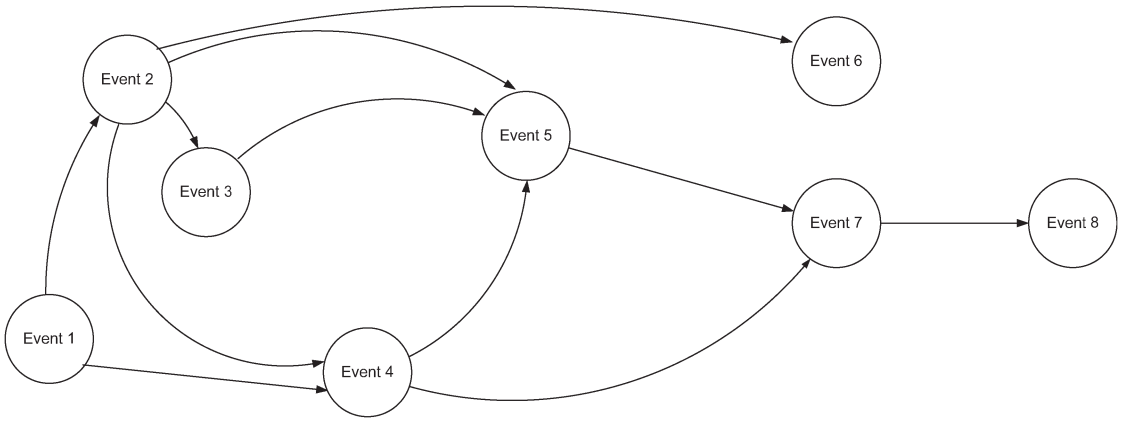


Fig. 8. Event evolution graph of the topic "Chechen Terrorists Seizing Beslan School."

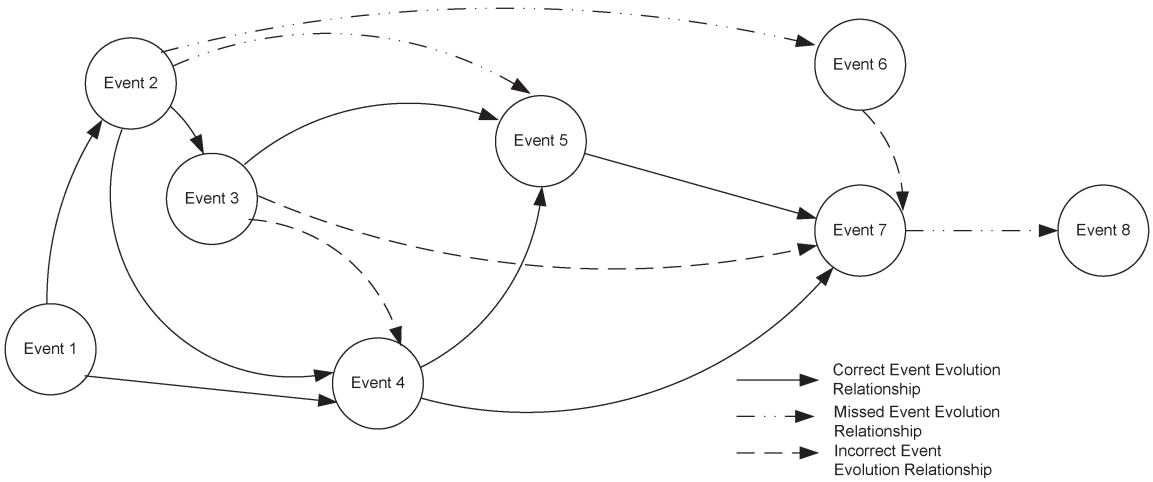


Fig. 9. Automatically generated event evolution graph with a threshold of 0.55.

Fig. 8 shows the event evolution graph of this terror attack case generated by two professional annotators. There are 11 event evolution relationships. Event 2 "Special task force assaulted terrorists and hundreds of hostages were dead" has four out-links which is the maximum in the event evolution graph. Thus, it can be considered as a seminal event that causes the sequences of events such as the responses of antiterrorism from different parties (event 3), the investigation of the attack (event 5), and identifying the suspects (event 4). Event 6 "Beslan school resumed classes after the hostage tragic" and event 8 "Russia's successive efforts against terrorism" are the terminal events, concluding the end of the terror attack case.

Figs. 9 and 10 show the event evolution graphs generated with the threshold of event evolution score as 0.55 and 0.60, respectively. In the former case $(\text{threshold} = 0.55)$, the precision is 0.73, and the recall is 0.73. In the latter case $(\text{threshold} = 0.60)$, the precision and recall are 0.85 and 0.55, respectively. Specifically, when we increase the threshold from 0.55 to 0.60, we reduce the number of incorrect event evolution relationships, but increase the number of missed event evolution relationships. As a result, the precision increases at the cost of recall.

As we observe in the automatically generated event evolution graphs, some event evolution relationships from Event 2 are

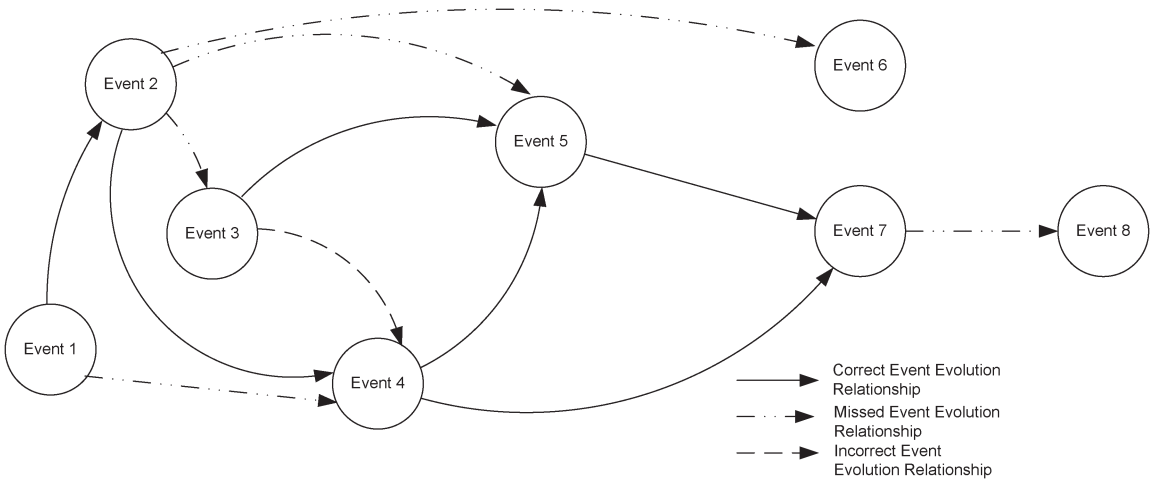


Fig. 10. Automatically generated event evolution graph with a threshold of 0.60.

missing. It is mainly because the temporal distances between Event 2 and its two child events (i.e., Events 5 and 6) are relatively larger than those between Event 2 and its two other child events (i.e., Events 3 and 4). The event evolution scores between Events 2 and 5 and between 2 and 6 become lower than the threshold while the event evolution scores between Events 2 and 3 and between Events 2 and 4 are still higher than the threshold. In addition, we also observe that the temporal distance between Events 4 and 7 is large. However, the event evolution score is still higher than the threshold because their event term vectors are similar. For example, the Chechen terrorists are identified as the suspects in Event 4, and they are the same group that the Russian are targeted to strike in Event 7. As a result, the event evolution relationship from Event 4 to 7 is retained. On the other hand, our proposed techniques generate an incorrect event evolution relationship from Event 3 to 4. It is because there are some overlapping content in the stories of both events and the temporal distance between the two events is small. According to the human annotations, there is no logical relationship between these two events. It is difficult to achieve perfect precision and recall; however, the proposed technique is promising in producing an event evolution graph with satisfactory precision and recall to support user navigation and understanding of the development of events in a given topic.

VII. Experiments and Analysis

A. Data Set

The news stories in our experiments are extracted from the CNN News website. All stories are written in English. The corpus is collected by automatic crawling and searching with the support of filtering by a human annotator. Given the URL of a beginning news story, the crawler analyzed the hyperlinks in the "related stories" section on each page of a news story, eliminated invalid links, and crawled relevant news stories via valid links. The crawler repeated this process until no more links were available or it reached a predefined depth. Since the section of "related stories" is often manually created, it is a good indicator of relevant stories in the topic. We collected ten topics, including politics, lawsuits, accidents, disasters, terrorism, etc.

TABLE II
Statistics of our Evaluation Data Set

Feature	Value
Average Number of Stories per Topic	78.2
Average Number of Words per Story	589
Average Number of Stories per Event	4.44
Average Number of Events per Topic	17.6
Average Number of Event Evolution Relationships per Topic	24.4
Average Number of Event Evolution Relationships per Event	1.4
Average Number of Days of Event Horizons of Topics	262

After crawling all linked news stories and filtering unrelated stories for each topic, we solicited two human annotators to create an event evolution graph for each topic independently and the reliability of agreement between two annotators were tested to ensure the validity of the event evolution graphs. Before the annotators started to construct event evolution graphs, we used the news stories of another topic as a sample to illustrate the granularity of events and the criteria for annotating event evolution relationships (i.e., the three conditions discussed in Section III). This training session continued until both annotators were familiar with the concept of events and event evolution relationships. Subsequently, the annotators were asked to read the news stories of each topic several times to form a general picture on its development. In the next step, each annotator was asked to identify the events for each topic, assigned each story to an event, and annotated evolution relationships between events independently. The two annotators then met together, reviewed the event evolution graphs constructed individually for each topic, and revised them to arrive at a "consensus" event evolution graph for the topic. The annotators took a half day, ten days, and five days in the training session, generating the first set of event evolution graphs, and reviewing jointly the event evolution graphs, respectively. It took 15.5 days in total to generate the whole data set.

Table II shows the statistics of our evaluation data set. The major difference between our data set and the TDT corpus

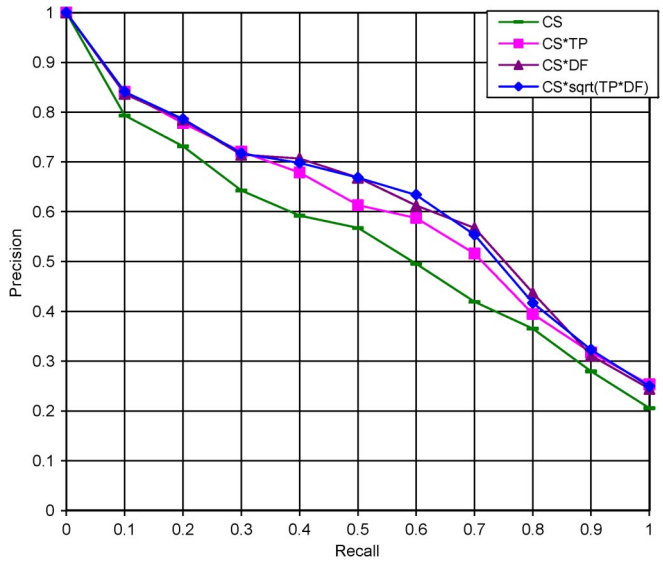


Fig. 11. Comparison of different event evolution scoring functions with the static thresholding pruning method. Only TF is used in the content similarity function. Precision and recall rates are calculated using the average across all topics and then interpolated to the standard 11-level.

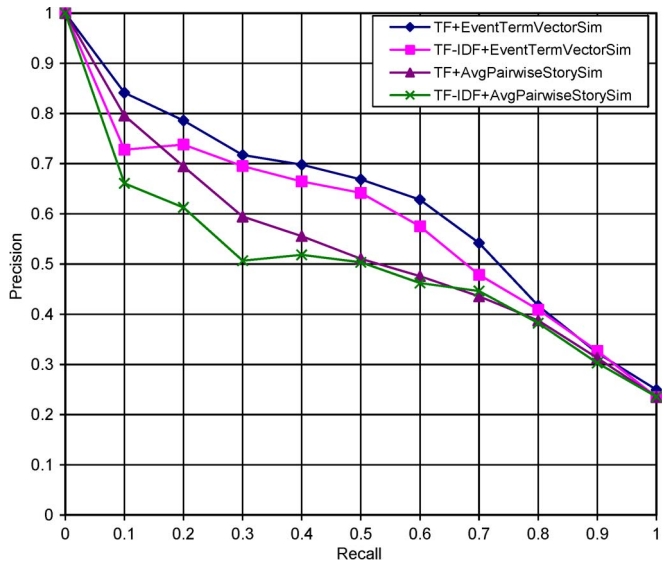


Fig. 12. Comparison of different event content similarity measures. We use $CS^*sqrt(TP^*DF)$ as our event evolution scoring function and the *static thresholding* pruning method.

adopted in the event threading study by Nallapati *et al.* [16] is that the stories in our data set are online news articles and their average length is much longer. The durations of most topics also span a much longer interval. Each topic includes significantly more events and stories in our data set. Therefore, the experimental results analyzed from this data set should be more persuasive in real-world practice.

B. Experimental Results and Analysis

Our measure of event evolution score considers three components: event content similarity, temporal proximity, and document distributional proximity (as (14) defines). Thus, in our first experiment, we attempt to compare the effectiveness of different event evolution scoring functions, including the following.

1) CS: Event content similarity only (i.e., without temporal proximity and document distributional proximity). This is equivalent to setting $\alpha = 0$ and $\beta = 0$ in our proposed event evolution scoring function in (14).
2) CS^*TP: Event content similarity multiplied by temporal proximity. This is equivalent to setting $\alpha = 1$ and $\beta = 0$.
3) CS^*DF: Event content similarity multiplied by document distributional proximity. This is equivalent to setting $\alpha = 0$ and $\beta = 1$.
4) $CS^*sqrt(TP^*DF)$: Event content similarity multiplied by the square root of the product of temporal proximity and document distributional proximity. This is equivalent to setting $\alpha = 1/2$ and $\beta = 1/2$.

Fig. 11 shows the evaluation results of the static thresholding pruning method across different values for λ. We observe that both the CS^*TP and CS^*DF scoring functions substantially outperform the CS scoring function. This suggests that temporal proximity and document distributional proximity are helpful for evaluating event evolution relationships. It is also interesting to observe that the $CS^*sqrt(TP^*DF)$ scoring function which incorporates both temporal proximity and document distributional proximity does not significantly outperform the CS^*DF function. Such comparable performance may be due to the overlapping effect of the temporal proximity and the document distributional proximity. When both are included, they cancel the effect of each other to some degree.

In our proposed event content similarity measure, we first compute the event term vectors (with the TF or TF-IDF representation) of two events using the average of the term vectors of stories that belong to each event and then take the cosine similarity of the two event term vectors to estimate the content similarity between the two events. In contrast, Nallapati *et al.* [16] employed a different event content similarity measure, estimated as the average of the similarities of all pairs of stories between two events. Therefore, the objective of the second experiment is to compare the effectiveness of different event content similarity measures. Specifically, we investigate the following four event content similarity measures.

1) *TF+EventTermVectorSim:* TF weight for story term vectors, cosine similarity of event term vectors. IDF factor is excluded.
2) *TF-IDF+EventTermVectorSim:* TF-IDF weight for story term vectors, cosine similarity of event term vectors. IDF factor is included.
3) *TF+AvgPairwiseStorySim:* TF weight for story term vectors, average of the pairwise cosine similarities of story term vectors. IDF factor is excluded.
4) *TF-IDF+AvgPairwiseStorySim:* TF-IDF weight for story term vectors, average of the pairwise cosine similarities of story term vectors. IDF factor is included. This is the measure proposed by Nallapati *et al.* [16].

As Fig. 12 shows, use of event term vectors for measuring event content similarity (i.e., *TF+EventTermVectorSim* and *TF-IDF+EventTermVectorSim*) substantially outperform

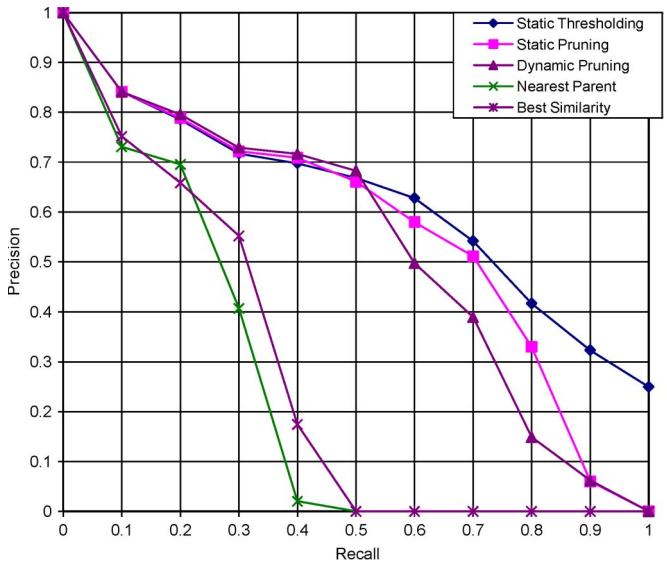


Fig. 13. Comparison of different pruning methods. We use the $CS^{*}sqrt(TP^{*}DF)$ event evolution scoring function and *TF+EventTermVectorSim* as the event content similarity measure.

the average pairwise similarity measures (i.e., *TF+AvgPairwiseStorySim* and *TF-IDF+AvgPairwiseStorySim*). Besides, the exclusion of the IDF factor when representing story term vectors generally improves the effectiveness of event evolution relationship identification in both the event term vector similarity measure and the average of the pairwise similarities of story term vectors measure. The superiority of the TF representation scheme for event evolution relationship identification differs from the findings commonly observed in information retrieval research (i.e., TF-IDF generally outperforms TF). Such inconsistent results may be resulted from differences of the target corpora managed in the two contexts. Particularly, the number of stories within a topic is much less than the number of documents in a typical information retrieval scenario. In addition, the content diversity of stories within a topic is significantly limited as compared to the diversified topics of documents for information retrieval.

The objective of our third experiment is to evaluate the effectiveness of our proposed pruning methods (including static thresholding, static pruning, and dynamic pruning), using the methods proposed by Nallapati *et al.* [16] as performance benchmarks. Nallapati *et al.* proposed several methods for modeling event dependencies, including *Nearest Parent* and *Best Similarity*. The nearest parent method assigns a parent event to an event e_i only if it is the most recent event preceding the event e_i. On the contrary, the best similarity parent method assigns a parent event to an event e_i only if the similarity between these two events is higher than the similarity between any other parent events and the event e_i. For our static pruning method, we set $N_o = 6$ and $N_i = 4$ empirically. For the dynamic pruning method, we set $\theta_o = 0.6$ and $\theta_i = 0.4$.

As Fig. 13 shows, our proposed pruning methods significantly outperform the *Nearest Parent* and *Best Similarity* methods employed by Nallapati *et al.* [16]. This result supports that our proposed technique is more effective than the previous models in capturing event evolution relationships and constructing event evolution graphs. In addition, the *static thresholding* method generally achieves higher effectiveness than the static pruning and dynamic pruning methods do. That is, it is not effective to apply a uniform upper bound on the degree of event threading and event joining to limit the number of edges per event. One possible explanation is that the degrees of event threading and joining vary substantially according to the importance of events. For example, some events are more important and critical and, hence, will have larger numbers of linkages to other events, but the remaining ones will not have many strong associations with other events.

VIII. Conclusion and Future Work

There is a large volume of news stories reporting ongoing incidents on the Web. In order to capture the development of the events in these incidents efficiently and effectively, we develop the event evolution identification technique to automatically identify event evolution relationships and represent the underlying structure as an event evolution graph. Contrary to the traditional view of topics as flat hierarchies in document clustering and categorizing tasks [1] or tree structure in event threading [16], we view news topic as a DAG with events as its nodes and event evolution relationships as its directed edges. We have introduced and employed the concept of event timestamp for measuring the temporal proximity of events. We have utilized the temporal proximity and document distributional proximity as decaying functions in addition to the cosine similarity of event term vectors for measuring event content similarities. We have investigated several graph-pruning methods to construct event evolution relationships for news topics effectively. According to our evaluation experiments, the performance of our proposed technique is promising and significantly outperforms our benchmark methods.

The event evolution graphs are useful to present the underlying structure of the events extracted for a topic. It helps to understand how the events evolve along the timeline. In our future work, we plan to develop information visualization tools that support users in conducting interactive browsing, extracting the main story line from the event evolution graph and extracting summary automatically for a specific path in the evolution graph. Given such visualization tools, users are able to comprehend the development of the events that are of their interests among all the relevant and less relevant events and stories. Using the summarization tools, users are also able to find the specific information to satisfy their information needs, and know when it happens and how it relates to the preceding and following events.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for on-demand classification of evolving data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 5, pp. 577–589, May 2006.

[2] J. Allan, "Detection as multi-topic tracking," *Inf. Retr.*, vol. 5, no. 2/3, pp. 139–157, Apr. 2002.

[3] J. Allan, *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA: Kluwer, 2000.

[4] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study: Final report," in *Proc. DARPA Broadcast News Transcription Understanding Workshop*, 1998, pp. 194–218.

[5] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval*, Melbourne, Australia, 1998, pp. 37–45.
[6] J. Allan, R. Gupta, and V. Khandelwal, "Temporal summaries of new topics," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval*, New Orleans, LA, 2001, pp. 10–18.
[7] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.
[8] J. Carthy, "Lexical chains for topic detection," Dept. Comput. Sci., Univ. College Dublin—National Univ. Ireland, Dublin, Ireland, 2002. Tech. Rep.
[9] C. C. Chen, Y. Chen, and M. C. Chen, "An aging theory for event life-cycle modeling," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 2, pp. 237–248, Mar. 2007.
[10] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, p. 29.
[11] C. Eden, "Analyzing cognitive maps to help structure issues or problems," *Eur. J. Oper. Res.*, vol. 159, no. 3, pp. 673–686, Dec. 2004.
[12] K. Y. Kwahk and Y. G. Kim, "Supporting business process redesign using cognitive maps," *Decis. Support Syst.*, vol. 25, no. 2, pp. 155–178, Mar. 1999.
[13] Z. Li, B. Wang, M. Li, and W. Ma, "A probabilistic model for retrospective news event detection," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval*, Salvador, Brazil, 2005, pp. 106–113.
[14] J. Makkonen, "Investigations on event evolution in TDT," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Language Technol., HLT-NAACL Student Research Workshop*, Edmonton, AB, Canada, 2003, pp. 43–48.
[15] Q. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: An exploration of temporal text mining," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, IL, 2005, pp. 198–207.
[16] R. Nallapati, A. Feng, F. Peng, and J. Allan, "Event threading within news topics," in *Proc. 13th ACM Int. Conf. Inf. Knowl. Management*, Washington, DC, 2004, pp. 446–453.
[17] A. Sun and E. Lim, "Hierarchical text classification and evaluation," in *Proc. 1st IEEE Int. Conf. Data Mining*, 2001, pp. 521–528.
[18] C. Wei and Y. Chang, "Discovering event evolution patterns from document sequences," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 2, pp. 273–283, Mar. 2007.
[19] C. Xu, Y. Zhang, G. Zhu, Y. Rui, H. Lu, and Q. Huang, "Using webcast text for semantic event detection in broadcast sports video," *IEEE Trans. Multimedia*, vol. 10, no. 7, pp. 1342–1355, Nov. 2008.
[20] C. C. Yang and X. Shi, "Discovering event evolution graphs from newswires," in *Proc. 15th Int. WWW Conf.*, Edinburgh, U.K., 2006, pp. 945–946.
[21] C. C. Yang and F. L. Wang, "An information delivery system with automatic summarization for mobile commerce," *Decis. Support Syst.*, vol. 43, no. 1, pp. 46–61, Feb. 2007.
[22] Y. Yang, T. Pierce, and J. Carbonell, "A study on retrospective and on-line event detection," in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval*, Melbourne, Australia, 1998, pp. 28–36.
[23] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, and X. Liu, "Learning approaches for detecting and tracking news events," *IEEE Intell. Syst.*, vol. 14, no. 4, pp. 32–43, Jul./Aug. 1999.
[24] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer, "Improving text categorization methods for event tracking," in *Proc. 23rd Annu. Int. ACM SIGIR Conf. Res. Development Inf. Retrieval*, Athens, Greece, 2000, pp. 65–72.
[25] Y. Yang, J. Carbonell, R. Brown, J. Lafferty, T. Pierce, and T. Ault, "Multi-strategy learning for TDT," in *Topic Detection and Tracking: Event-Based Information Organization*, J. Allan, Ed. Norwell, MA: Kluwer, 2002, pp. 85–114.
[26] Y. Yang, J. Zhang, J. Carbonell, and C. Jin, "Topic-conditioned novelty detection," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Edmonton, AB, Canada, 2002, pp. 688–693.
[27] M. Yeh, B. Dai, and M. Chen, "Clustering over multiple evolving streams by events and correlation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 10, pp. 1349–1362, Oct. 2007.

Christopher C. Yang received his Ph.D. degree in computer engineering from the University of Arizona in 1997.

He is currently an Associate Professor with the College of Information Science and Technology, Drexel University, Philadelphia, PA. He has also been a faculty member with the Department of Systems Engineering and Engineering Management and the Director of the Digital Library Laboratory, The Chinese University of Hong Kong, Shatin, Hong Kong, a faculty member in the Department of Computer Science, The University of Hong Kong. He has published over 150 referred journal and conference papers in the *Journal of the American Society for Information Science and Technology (JASIST), Decision Support Systems (DSS)*, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, IEEE COMPUTER, *Information Processing and Management (IPM), Journal of Information Science, Graphical Models and Image Processing, Optical Engineering, Pattern Recognition, International Journal of Electronic Commerce, Applied Artificial Intelligence, IWWWC, SIGIR, ICIS, CIKM*, and more. His recent research interests include Web search and mining, social media analytics, cross-lingual information retrieval and knowledge management, security informatics, text summarization, information visualization, digital libraries, and electronic commerce.

Dr. Yang has edited several special issues on multilingual information systems, knowledge management, and Web mining in *JASIST*, *IPM*, and *DSS*. He has chaired and served in many international conferences and workshops such as the IEEE International Conference on Intelligence and Security Informatics, International Conference on Electronic Commerce, and the ACM Conference on Information and Knowledge Management.

Xiaodong Shi received the Master of Philosophy degree from The Chinese University of Hong Kong, Shatin, Hong Kong.

He is currently a Research Assistant under the supervision of Christopher C. Yang with the Digital Library Laboratory, The Chinese University of Hong Kong. His publications have appeared in the *Journal of the American Society for Information Science and Technology* and International World Wide Web Conference.

Chih-Ping Wei received the B.S. degree in management science from the National Chiao-Tung University, Hsinchu, Taiwan, in 1987 and the M.S. and Ph.D. degrees in management information systems from the University of Arizona, Tucson, in 1991 and 1996, respectively.

He is currently a Professor with the Institute of Service Science, National Tsing Hua University, Hsinchu. Prior to joining the National Tsing Hua University in 2005, he was a faculty member with the Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan, since 1996 and a Visiting Scholar with the University of Illinois at Urbana–Champaign in Fall 2001 and The Chinese University of Hong Kong in Summer 2006 and 2007. His papers have appeared in the *Journal of Management Information Systems, Decision Support Systems (DSS)*, IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, IEEE SOFTWARE, IEEE INTELLIGENT SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, *European Journal of Information Systems, Journal of Database Management, Information Processing and Management, Journal of Organizational Computing and Electronic Commerce*, etc. His current research interests include knowledge discovery and data mining, information retrieval and text mining, knowledge management, data warehouse design, and patent informatics. He has edited special issues of DSS, *International Journal of Electronic Commerce*, and *Electronic Commerce Research and Applications*.