Event Threading within News Topics

Ramesh Nallapati, Ao Feng, Fuchun Peng, James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{nmramesh,aofeng,fuchun,allan}@cs.umass.edu

ABSTRACT

With the overwhelming volume of online news available today, there is an increasing need for automatic techniques to analyze and present news to the user in a meaningful and efficient manner. Previous research focused only on organizing news stories by their topics into a flat hierarchy. We believe viewing a news topic as a flat collection of stories is too restrictive and inefficient for a user to understand the topic quickly.

In this work, we attempt to capture the rich structure of events and their dependencies in a news topic through our event models. We call the process of recognizing events and their dependencies *event threading*. We believe our perspective of modeling the structure of a topic is more effective in capturing its semantics than a flat list of on-topic stories.

We formally define the novel problem, suggest evaluation metrics and present a few techniques for solving the problem. Besides the standard word based features, our approaches take into account novel features such as temporal locality of stories for event recognition and time-ordering for capturing dependencies. Our experiments on a manually labeled data sets show that our models effectively identify the events and capture dependencies among them.

Categories and Subject Descriptors

H.3.3 **[Information Search and Retrieval]**: Clustering

General Terms

Algorithms, Experimentation, Measurement

Key words

Event, dependency, threading, clustering

1. INTRODUCTION

News forms a major portion of information disseminated in the world everyday. Common people and news analysts alike are very interested in keeping abreast of new things that happen in the news, but it is becoming very difficult to cope with the huge volumes of information that arrives each day. Hence there is an increasing need for automatic techniques to organize news stories in a way that helps users interpret and analyze them quickly. This problem is addressed by a research program called Topic Detection and Tracking (TDT) [3] that runs an open annual competition on standardized tasks of news organization.

One of the shortcomings of current TDT evaluation is its view of news topics as flat collection of stories. For example, the detection task of TDT is to arrange a collection of news stories into clusters of topics. However, a topic in news is more than a mere collection of stories: it is characterized by a definite structure of inter-related events. This is indeed recognized by TDT which defines a topic as '*a set of news stories that are strongly related by some seminal real-world event*' where an event is defined as '*something that happens at a specific time and location*' [3]. For example, when a bomb explodes in a building, that is the seminal event that triggers the topic. Other events in the topic may include the rescue attempts, the search for perpetrators, arrests and trials and so on. We see that there is a pattern of dependencies between pairs of events in the topic. In the above example, the event of rescue attempts is 'influenced' by the event of bombing and so is the event of search for perpetrators.

In this work we investigate methods for modeling the structure of a topic in terms of its events. By structure, we mean not only identifying the events that make up a topic, but also establishing dependencies—generally causal—among them. We call the process of recognizing events and identifying dependencies among them *event threading*, an analogy to email threading that shows connections between related email messages. We refer to the resulting interconnected structure of events as the *event model* of the topic. Although this paper focuses on threading events within an existing news topic, we expect that such event based dependency structure more accurately reflects the structure of news than strictly bounded topics do. From a user's perspective, we believe that our view of a news topic as a set of interconnected events helps him/her get a quick overview of the topic and also allows him/her navigate through the topic faster.

The rest of the paper is organized as follows. In section 2, we discuss related work. In section 3, we define the problem and use an example to illustrate threading of events within a news topic. In section 4, we describe how we built the corpus for our problem. Section 5 presents our evaluation techniques while section 6 describes the techniques we use for modeling event structure. In section 7 we present our experiments and results. Section 8 concludes the paper with a few observations on our results and comments on future work.

CIKM'04, November 8–13, 2004, Washington,DC,USA.

2. RELATED WORK

The process of threading events together is related to threading of electronic mail only by name for the most part. Email usually incorporates a strong structure of referenced messages and consistently formatted subject headings—though information retrieval techniques are useful when the structure breaks down [7]. Email threading captures reference dependencies between messages and does not attempt to reflect any underlying real-world structure of the matter under discussion.

Another area of research that looks at the structure within a topic is hierarchical text classification of topics [9, 6]. The hierarchy within a topic does impose a structure on the topic, but we do not know of an effort to explore the extent to which that structure reflects the underlying event relationships.

Barzilay and Lee [5] proposed a content structure modeling technique where topics within text are learnt using unsupervised methods, and a linear order of these topics is modeled using hidden Markov models. Our work differs from theirs in that we do not constrain the dependency to be linear. Also their algorithms are tuned to work on specific genres of topics such as earthquakes, accidents, etc., while we expect our algorithms to generalize over any topic.

In TDT, researchers have traditionally considered topics as flat-clusters [1]. However, in TDT-2003, a hierarchical structure of topic detection has been proposed and [2] made useful attempts to adopt the new structure. However this structure still did not explicitly model any dependencies between events.

In a work closest to ours, Makkonen [8] suggested modeling news topics in terms of its evolving events. However, the paper stopped short of proposing any models to the problem. Other related work that dealt with analysis within a news topic includes temporal summarization of news topics [4].

3. PROBLEM DEFINITION AND NOTATION

In this work, we have adhered to the definition of event and topic as defined in TDT. We present some definitions (in italics) and our interpretations (regular-faced) below for clarity.

1. **Story**: *A story is a news article delivering some information to users.* In TDT, a story is assumed to refer to only a single topic. In this work, we also assume that each story discusses a single event. In other words, a story is the smallest atomic unit in the hierarchy (topic $\rightarrow$ event $\rightarrow$ story). Clearly, both the assumptions are not necessarily true in reality, but we accept them for simplicity in modeling.

2. **Event**: *An event is something that happens at some specific time and place [10].* In our work, we represent an event by a set of stories that discuss it. Following the assumption of atomicity of a story, this means that any set of distinct events can be represented by a set of non-overlapping clusters of news stories.

3. **Topic**: *A set of news stories strongly connected by a seminal event.* We expand on this definition and interpret a topic as a series of related events. Thus a topic can be represented by clusters of stories each representing an event and a set of (directed or undirected) edges between pairs of these clusters representing the dependencies between these events. We will describe this representation of a topic in more detail in the next section.

4. **Topic detection and tracking (TDT)** :*Topic detection detects clusters of stories that discuss the same topic; Topic tracking detects stories that discuss a previously known topic [3].* Thus TDT concerns itself mainly with clustering stories into topics that discuss them.

5. **Event threading**: *Event threading detects events within in a topic, and also captures the dependencies among the events.* Thus the main difference between event threading and TDT is that we focus our modeling effort on microscopic events rather than larger topics. Additionally event threading models the relatedness or dependencies between pairs of events in a topic while TDT models topics as unrelated clusters of stories.

We first define our problem and representation of our model formally and then illustrate with the help of an example. We are given a set of n news stories $\mathbf{S} = \{s_1, \cdots, s_n\}$ on a given topic $\mathcal{T}$ and their time of publication. We define a set of events $\mathcal{E} = \{\mathcal{E}_1, \cdots, \mathcal{E}_m\}$ with the following constraints:

$$\forall\, i \qquad \mathcal{E}_i \in 2^{\mathbf{S}} \tag{1}$$

$$\forall\, i, j \qquad s.t.\ i \neq j,\ \mathcal{E}_i \cap \mathcal{E}_j = \{\} \tag{2}$$

$$\forall s_i \quad \exists \quad \mathcal{E}_k \in \mathcal{E}\ s.t.\ s_i \in \mathcal{E}_k \tag{3}$$

While the first constraint says that each event is an element in the power set of $\mathbf{S}$, the second constraint ensures that each story can belong to at most one event. The last constraint tells us that every story belongs to one of the events in $\mathcal{E}$. In fact this allows us to define a mapping function f from stories to events as follows:

$$f(s_i) = \mathcal{E}_k \text{ iff } s_i \in \mathcal{E}_k \tag{4}$$

Further, we also define a set of directed edges $\mathbf{E} = \{(\mathcal{E}_i, \mathcal{E}_j)\}$ which denote dependencies between events. It is important to explain what we mean by this directional dependency: While the existence of an edge itself represents relatedness of two events, the direction could imply causality or temporal-ordering. By causal dependency we mean that the occurrence of event B is related to and is a consequence of the occurrence of event A. By temporal ordering, we mean that event B happened after event A and is related to A but is not necessarily a consequence of A. For example, consider the following two events: 'plane crash' (event A) and 'subsequent investigations' (event B) in a topic on a plane crash incident. Clearly, the investigations are a result of the crash. Hence an arrow from A to B falls under the category of causal dependency. Now consider the pair of events 'Pope arrives in Cuba'(event A) and 'Pope meets Castro'(event B) in a topic that discusses Pope's visit to Cuba. Now events A and B are closely related through their association with the Pope and Cuba but event B is not necessarily a consequence of the occurrence of event A. An arrow in such scenario captures what we call time ordering. In this work, we do not make an attempt to distinguish between these two kinds of dependencies and our models treats them as identical. A simpler (and hence less controversial) choice would be to ignore direction in the dependencies altogether and consider only undirected edges. This choice definitely makes sense as a first step but we chose the former since we believe directional edges make more sense to the user as they provide a more illustrative flow-chart perspective to the topic.

To make the idea of event threading more concrete, consider the example of TDT3 topic 30005, titled 'Osama bin Laden's Indictment' (in the 1998 news). This topic has 23 stories which form 5 events. An event model of this topic can be represented as in figure 1. Each box in the figure indicates an event in the topic of Osama's indictment. The occurrence of event 2, namely 'Trial and Indictment of Osama' is dependent on the event of 'evidence gathered by CIA', i.e., event 1. Similarly, event 2 influences the occurrences of events 3, 4 and 5, namely 'Threats from Militants', 'Reactions

from Muslim World' and 'announcement of reward'. Thus all the dependencies in the example are causal.

Extending our notation further, we call an event A a parent of B and B the child of A, if $(A, B) \in \mathbf{E}$. We define an event model $M = (\mathcal{E}, \mathbf{E})$ to be a tuple of the set of events and set of dependencies.

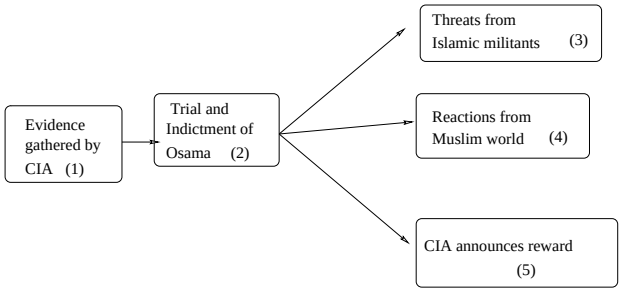

Figure 1: An event model of TDT topic 'Osama bin Laden's indictment'.

Event threading is strongly related to topic detection and tracking, but also different from it significantly. It goes beyond topics, and models the relationships between events. Thus, event threading can be considered as a further extension of topic detection and tracking and is more challenging due to at least the following difficulties.

1. The number of events is unknown.
2. The granularity of events is hard to define.
3. The dependencies among events are hard to model.
4. Since it is a brand new research area, no standard evaluation metrics and benchmark data is available.

In the next few sections, we will describe our attempts to tackle these problems.

4. LABELED DATA

We picked 28 topics from the TDT2 corpus and 25 topics from the TDT3 corpus. The criterion we used for selecting a topic is that it should contain at least 15 on-topic stories from CNN headline news. If the topic contained more than 30 CNN stories, we picked only the first 30 stories to keep the topic short enough for annotators. The reason for choosing only CNN as the source is that the stories from this source tend to be short and precise and do not tend to digress or drift too far away from the central theme. We believe modeling such stories would be a useful first step before dealing with more complex data sets.

We hired an annotator to create truth data. Annotation includes defining the event membership for each story and also the dependencies. We supervised the annotator on a set of three topics that we did our own annotations on and then asked her to annotate the 28 topics from TDT2 and 25 topics from TDT3.

In identifying events in a topic, the annotator was asked to broadly follow the TDT definition of an event, i.e., 'something that happens at a specific time and location'. The annotator was encouraged to merge two events A and B into a single event C if any of the stories discusses both A and B. This is to satisfy our assumption that each story corresponds to a unique event. The annotator was also encouraged to avoid singleton events, events that contain a single news story, if possible. We realized from our own experience that people differ in their perception of an event especially when the number of stories in that event is small. As part of the guidelines, we instructed the annotator to assign titles to all the events in each topic. We believe that this would help make her understanding of the events more concrete. We however, do not use or model these titles in our algorithms.

In defining dependencies between events, we imposed no restrictions on the graph structure. Each event could have single, multiple or no parents. Further, the graph could have cycles or orphan-nodes. The annotator was however instructed to assign a dependency from event A to event B if and only if the occurrence of B is 'either causally influenced by A or is closely related to A and follows A in time'.

From the annotated topics, we created a training set of 26 topics and a test set of 27 topics by merging the 28 topics from TDT2 and 25 from TDT3 and splitting them randomly. Table 1 shows that the training and test sets have fairly similar statistics.

Feature	Training set	Test set
Num. topics	26	27
Avg. Num. Stories/Topic	28.69	26.74
Avg. Doc. Len.	64.60	64.04
Avg. Num. Stories/Event	5.65	6.22
Avg. Num. Events/Topic	5.07	4.29
Avg. Num. Dependencies/Topic	3.07	2.92
Avg. Num. Dependencies/Event	0.61	0.68
Avg. Num. Days/Topic	30.65	34.48

Table 1: Statistics of annotated data

5. EVALUATION

A system can generate some event model $M' = (\mathcal{E}', \mathbf{E}')$ using certain algorithms, which is usually different from the truth model $M = (\mathcal{E}, \mathbf{E})$ (we assume the annotator did not make any mistake). Comparing a system event model M' with the true model M requires comparing the entire event models including their dependency structure. And different event granularities may bring huge discrepancy between M' and M. This is certainly non-trivial as even testing whether two graphs are isomorphic has no known polynomial time solution. Hence instead of comparing the actual structure we examine a pair of stories at a time and verify if the system and true labels agree on their event-memberships and dependencies. Specifically, we compare two kinds of story pairs:

- **Cluster pairs ($C(M)$)**: These are the complete set of *unordered* pairs (s_i, s_j) of stories s_i and s_j that fall within the same event given a model M. Formally,

$$C(M) = \{(s_i, s_j) | s_i, s_j \in S \cap f(s_i) = f(s_j)\} \quad (5)$$

  where f is the function in M that maps stories to events as defined in equation 4.

- **Dependency pairs ($D(M)$)**: These are the set of all *ordered* pairs of stories (s_i, s_j) such that there is a dependency from the event of s_i to the event of s_j in the model M.

$$D(M) = \{(s_i, s_j) | (f(s_i), f(s_j)) \in \mathbf{E}\} \quad (6)$$

  Note the story pair is ordered here, so (s_i, s_j) is not equivalent to (s_j, s_i). In our evaluation, a correct pair with wrong

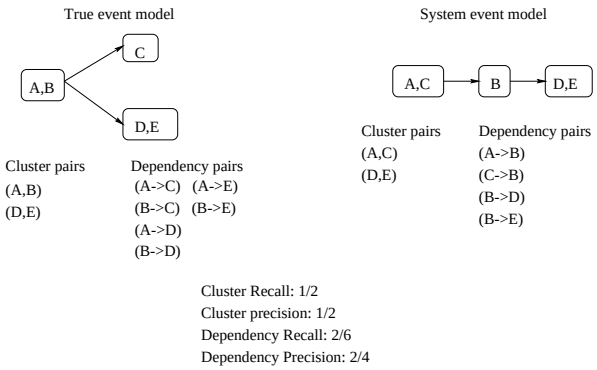

Figure 2: Evaluation measures

direction will be considered a mistake. As we mentioned earlier in section 3, ignoring the direction may make the problem simpler, but we will lose the expressiveness of our representation.

Given these two sets of story pairs corresponding to the true event model M and the system event model M', we define recall and precision for each category as follows.

- **Cluster Precision (CP)**: It is the probability that two randomly selected stories s_i and s_j are in the same true-event given that they are in the same system event.

$$\begin{aligned} CP &= P(f(s_i) = f(s_j) | f'(s_i) = f'(s_j)) \\ &= \frac{|C(M) \cap C(M')|}{|C(M')|} \end{aligned} \quad (7)$$

where f' is the story-event mapping function corresponding to the model M'.

- **Cluster Recall(CR)**: It is the probability that two randomly selected stories s_i and s_j are in the same system-event given that they are in the same true event.

$$\begin{aligned} CR &= P(f'(s_i) = f'(s_j) | f(s_i) = f(s_j)) \\ &= \frac{|C(M) \cap C(M')|}{|C(M)|} \end{aligned} \quad (8)$$

- **Dependency Precision(DP)**: It is the probability that there is a dependency between the events of two randomly selected stories s_i and s_j in the true model M given that they have a dependency in the system model M'. Note that the direction of dependency is important in comparison.

$$\begin{aligned} DP &= P((f(s_i), f(s_j)) \in E | (f'(s_i), f'(s_j)) \in E') \\ &= \frac{|D(M) \cap D(M')|}{|D(M')|} \end{aligned} \quad (9)$$

- **Dependency Recall(DR)**: It is the probability that there is a dependency between the events of two randomly selected stories s_i and s_j in the system model M' given that they have a dependency in the true model M. Again, the direction of dependency is taken into consideration.

$$\begin{aligned} DR &= P((f'(s_i), f'(s_j)) \in E' | (f(s_i), f(s_j)) \in E) \\ &= \frac{|D(M) \cap D(M')|}{|D(M)|} \end{aligned} \quad (10)$$

The measures are illustrated by an example in figure 2. We also combine these measures using the well known F1-measure commonly used in text classification and other research areas as shown below.

$$\begin{aligned} CF &= \frac{2 \times CP \times CR}{CP + CR} \\ DF &= \frac{2 \times DP \times DR}{DP + DR} \\ JF &= \frac{2 \times CF \times DF}{CF + DF} \end{aligned} \quad (11)$$

where CF and DF are the cluster and dependency F1-measures respectively and JF is the Joint F1-measure (JF) that we use to measure the overall performance.

6. TECHNIQUES

The task of event modeling can be split into two parts: clustering the stories into unique events in the topic and constructing dependencies among them. In the following subsections, we describe techniques we developed for each of these sub-tasks.

6.1 Clustering

Each topic is composed of multiple events, so stories must be clustered into events before we can model the dependencies among them. For simplicity, all stories in the same topic are assumed to be available at one time, rather than coming in a text stream. This task is similar to traditional clustering but features other than word distributions may also be critical in our application.

In many text clustering systems, the similarity between two stories is the inner product of their *tf-idf* vectors, hence we use it as one of our features. Stories in the same event tend to follow temporal locality, so the time stamp of each story can be a useful feature. Additionally, named-entities such as person and location names are another obvious feature when forming events. Stories in the same event tend to be related to the same person(s) and locations(s).

In this subsection, we present an agglomerative clustering algorithm that combines all these features. In our experiments, however, we study the effect of each feature on the performance separately using modified versions of this algorithm.

6.1.1 Agglomerative clustering with time decay (ACDT)

We initialize our events to singleton events (clusters), i.e., each cluster contains exactly one story. So the similarity between two events, to start with, is exactly the similarity between the corresponding stories. The similarity $wsum(s_1, s_2)$ between two stories s_1 and s_2 is given by the following formula:

$$wsum(s_1, s_2) = \omega_1 cos(s_1, s_2) + \omega_2 Loc(s_1, s_2) + \omega_3 Per(s_1, s_2) \quad (12)$$

Here $\omega_1, \omega_2, \omega_3$ are the weights on different features. In this work, we determined them empirically, but in the future, one can consider more sophisticated learning techniques to determine them. $cos(s_1, s_2)$ is the cosine similarity of the term vectors. $Loc(s_1, s_2)$ is 1 if there is some location that appears in both stories, otherwise it is 0. $Per(s_1, s_2)$ is similarly defined for person name.

We use time decay when calculating similarity of story pairs, i.e., the larger time difference between two stories, the smaller their similarities. The time period of each topic differs a lot, from a few days to a few months. So we normalize the time difference using the whole duration of that topic. The time decay adjusted similarity

$sim(s_1, s_2)$ is given by

$$sim(s_1, s_2) = wsum(s_1, s_2)e^{-\frac{\alpha|t_1 - t_2|}{T}} \quad (13)$$

where t_1 and t_2 are the time stamps for story 1 and 2 respectively. T is the time difference between the earliest and the latest story in the given topic. α is the time decay factor.

In each iteration, we find the most similar event pair and merge them. We have three different ways to compute the similarity between two events $\mathcal{E}_u$ and $\mathcal{E}_v$:

- Average link: In this case the similarity is the average of the similarities of all pairs of stories between $\mathcal{E}_u$ and $\mathcal{E}_v$ as shown below:

$$sim(\mathcal{E}_u, \mathcal{E}_v) = \frac{\sum_{s_u \in \mathcal{E}_u} \sum_{s_v \in \mathcal{E}_v} sim(s_u, s_v)}{|\mathcal{E}_u||\mathcal{E}_v|} \quad (14)$$

- Complete link: The similarity between two events is given by the smallest of the pair-wise similarities.

$$sim(\mathcal{E}_u, \mathcal{E}_v) = \min_{s_u \in \mathcal{E}_u, s_v \in \mathcal{E}_v} sim(s_u, s_v) \quad (15)$$

- Single link: Here the similarity is given by the best similarity between all pairs of stories.

$$sim(\mathcal{E}_u, \mathcal{E}_v) = \max_{s_u \in \mathcal{E}_u, s_v \in \mathcal{E}_v} sim(s_u, s_v) \quad (16)$$

This process continues until the maximum similarity falls below the threshold or the number of clusters is smaller than a given number.

6.2 Dependency modeling

Capturing dependencies is an extremely hard problem because it may require a 'deeper understanding' of the events in question. A human annotator decides on dependencies not just based on the information in the events but also based on his/her vast repertoire of domain-knowledge and general understanding of how things operate in the world. For example, in Figure 1 a human knows 'Trial and indictment of Osama' is influenced by 'Evidence gathered by CIA' because he/she understands the process of law in general.

We believe a robust model should incorporate such domain knowledge in capturing dependencies, but in this work, as a first step, we will rely on surface-features such as time-ordering of news stories and word distributions to model them. Our experiments in later sections demonstrate that such features are indeed useful in capturing dependencies to a large extent.

In this subsection, we describe the models we considered for capturing dependencies. In the rest of the discussion in this subsection, we assume that we are already given the mapping $f' : \mathbf{S} \rightarrow \mathcal{E}$ and we focus only on modeling the edges $\mathbf{E}'$. First we define a couple of features that the following models will employ.

First we define a 1-1 time-ordering function $t : S \rightarrow \{1, \cdots, n\}$ that sorts stories in ascending order by their time of publication. Now, the event-time-ordering function t_e is defined as follows.

$$\begin{aligned} t_e : \quad & \mathcal{E} \rightarrow \{1, \cdots, m\} \; s.t. \\ \forall \mathcal{E}_u, \mathcal{E}_v \in \mathcal{E} \quad & t_e(\mathcal{E}_u) < t_e(\mathcal{E}_v) \iff \min_{s_u \in \mathcal{E}_u} t(s_u) < \min_{s_v \in \mathcal{E}_v} t(s_v) \end{aligned} \quad (17)$$

In other words, t_e time-orders events based on the time-ordering of their respective first stories.

We will also use average cosine similarity between two events as a feature and it is defined as follows.

$$AvgSim(\mathcal{E}_u, \mathcal{E}_v) = \frac{\sum_{s_u \in \mathcal{E}_u} \sum_{s_v \in \mathcal{E}_v} cos(s_u, s_v)}{|\mathcal{E}_u||\mathcal{E}_v|} \quad (18)$$

6.2.1 Complete-Link model

In this model, we assume that there are dependencies between all pairs of events. The direction of dependency is determined by the time-ordering of the first stories in the respective events. Formally, the system edges are defined as follows.

$$\mathbf{E}' = \{(\mathcal{E}_u, \mathcal{E}_v) \mid t_e(\mathcal{E}_u) < t_e(\mathcal{E}_v)\} \quad (19)$$

where t_e is the event-time-ordering function. In other words, the dependency edge is directed from event $\mathcal{E}_u$ to event $\mathcal{E}_v$, if the first story in event $\mathcal{E}_u$ is earlier than the first story in event $\mathcal{E}_v$. We point out that this is not to be confused with the complete-link algorithm in clustering. Although we use the same names, it will be clear from the context which one we refer to.

6.2.2 Simple Thresholding

This model is an extension of the complete link model with an additional constraint that there is a dependency between any two events $\mathcal{E}_u$ and $\mathcal{E}_v$ only if the average cosine similarity between event $\mathcal{E}_u$ and event $\mathcal{E}_v$ is greater than a threshold T. Formally,

$$\begin{aligned} \mathbf{E}' = \{&(\mathcal{E}_u, \mathcal{E}_v) \mid AvgSim(\mathcal{E}_u, \mathcal{E}_v) > T \\ &\cap \; t_e(\mathcal{E}_u) < t_e(\mathcal{E}_v)\} \end{aligned} \quad (20)$$

6.2.3 Nearest Parent Model

In this model, we assume that each event can have at most one parent. We define the set of dependencies as follows.

$$\begin{aligned} \mathbf{E}' = \{&(\mathcal{E}_u, \mathcal{E}_v) \mid AvgSim(\mathcal{E}_u, \mathcal{E}_v) > T \\ &\cap \; t_e(\mathcal{E}_v) = t_e(\mathcal{E}_u) + 1\} \end{aligned} \quad (21)$$

Thus, for each event $\mathcal{E}_v$, the nearest parent model considers only the event preceding it as defined by t_e as a potential candidate. The candidate is assigned as the parent only if the average similarity exceeds a pre-defined threshold T.

6.2.4 Best Similarity Model

This model also assumes that each event can have at most one parent. An event $\mathcal{E}_v$ is assigned a parent $\mathcal{E}_u$ if and only if $\mathcal{E}_u$ is the most similar *earlier* event to $\mathcal{E}_v$ and the similarity exceeds a threshold T. Mathematically, this can be expressed as:

$$\begin{aligned} \mathbf{E}' = \{&(\mathcal{E}_u, \mathcal{E}_v) \mid AvgSim(\mathcal{E}_u, \mathcal{E}_v) > T \\ &\cap \; \mathcal{E}_u = \arg \max_{\mathcal{E}_w : t_e(\mathcal{E}_w) < t_e(\mathcal{E}_v)} AvgSim(\mathcal{E}_w, \mathcal{E}_v)\} \end{aligned} \quad (22)$$

6.2.5 Maximum Spanning Tree model

In this model, we first build a maximum spanning tree (MST) using a greedy algorithm on the following fully connected *weighted, undirected* graph whose vertices are the events and whose edges $\hat{\mathbf{E}}$ are defined as follows:

$$\hat{\mathbf{E}} = \{(\mathcal{E}_u, \mathcal{E}_v)\} \cap \; w(\mathcal{E}_u, \mathcal{E}_v) = AvgSim(\mathcal{E}_u, \mathcal{E}_v) \quad (23)$$

Let $MST(\hat{\mathbf{E}})$ be the set of edges in the maximum spanning tree of $\mathbf{E}'$. Now our *directed* dependency edges $\mathbf{E}$ are defined as follows.

$$\begin{aligned} \mathbf{E}' = \{&(\mathcal{E}_u, \mathcal{E}_v) | (\mathcal{E}_u, \mathcal{E}_v) \in MST(\hat{\mathbf{E}}) \cap \; t_e(\mathcal{E}_u) < t_e(\mathcal{E}_v) \\ &\cap \; AvgSim(\mathcal{E}_u, \mathcal{E}_v) > T\} \end{aligned} \quad (24)$$

Thus in this model, we assign dependencies between the most similar events in the topic.

7. EXPERIMENTS

Our experiments consists of three parts. First we modeled only the event clustering part (defining the mapping function f') using clustering algorithms described in section 6.1. Then we modeled only the dependencies by providing to the system the true clusters and running only the dependency algorithms of section 6.2. Finally, we experimented with combinations of clustering and dependency algorithms to produce the complete event model. This way of experimentation allows us to compare the performance of our algorithms in isolation and in association with other components. The following subsections present the three parts of our experimentation.

7.1 Clustering

We have tried several variations of the $ACDT$ algorithm to study the effects of various features on the clustering performance. All the parameters are learned by tuning on the training set. We also tested the algorithms on the test set with parameters fixed at their optimal values learned from training. We used agglomerative clustering based on only cosine similarity as our clustering baseline. The results on the training and test sets are in Table 2 and 3 respectively. We use the Cluster F1-measure (CF) averaged over all topics as our evaluation criterion.

Model	best T	CP	CR	CF	P-value
cos+1-lnk	0.15	0.41	0.56	0.43	-
cos+all-lnk	0.00	0.40	0.62	0.45	-
cos+Loc+avg-lnk	0.07	0.37	0.74	0.45	-
cos+Per+avg-lnk	0.07	0.39	0.70	0.46	-
cos+TD+avg-lnk	0.04	0.45	0.70	0.53	2.9e-4*
cos+N(T)+avg-lnk	-	0.41	0.62	0.48	7.5e-2
cos+N(T)+T+avg-lnk	0.03	0.42	0.62	0.49	2.4e-2*
cos+TD+N(T)+avg-lnk	-	0.44	0.66	0.52	7.0e-3*
cos+TD+N(T)+T+avg-lnk	0.03	0.47	0.64	0.53	1.1e-3*
Baseline(cos+avg-lnk)	0.05	0.39	0.67	0.46	-

Table 2: Comparison of agglomerative clustering algorithms (training set)

Model	CP	CR	CF	P-value
cos+1-lnk	0.43	0.49	0.39	-
cos+all-lnk	0.43	0.62	0.47	-
cos+Loc+avg-lnk	0.37	0.73	0.45	-
cos+Per+avg-lnk	0.44	0.62	0.45	-
cos+TD+avg-lnk	0.48	0.70	0.54	0.014*
cos+N(T)+avg-lnk	0.41	0.71	0.51	0.31
cos+N(T)+T+avg-lnk	0.43	0.69*	0.52	0.14
cos+TD+N(T)+avg-lnk	0.43	0.76	0.54	0.025*
cos+TD+N(T)+T+avg-lnk	0.47	0.69	0.54	0.0095*
Baseline(cos+avg-lnk)	0.44	0.67	0.50	-

Table 3: Comparison of agglomerative clustering algorithms (test set)

P-value marked with a $*$ means that it is a statistically significant improvement over the baseline (95% confidence level, one tailed T-test). The methods shown in table 2 and 3 are:

- *Baseline*: *tf-idf* vector weight, cosine similarity, average link in clustering. In equation 12, $\omega_1 = 1$, $\omega_2 = \omega_3 = 0$. And $\alpha = 0$ in equation 13. This F-value is the maximum obtained by tuning the threshold.
- *cos+1-lnk*: Single link comparison (see equation 16) is used where similarity of two clusters is the maximum of all story pairs, other configurations are the same as the baseline run.
- *cos+all-lnk*: Complete link algorithm of equation 15 is used. Similar to single link but it takes the minimum similarity of all story pairs.
- *cos+Loc+avg-lnk*: Location names are used when calculating similarity. $\omega_2 = 0.05$ in equation 12. All algorithms starting from this one use average link (equation 14), since single link and complete link do not show any improvement of performance.
- *cos+Per+avg-lnk*: $\omega_3 = 0.05$ in equation 12, i.e., we put some weight on person names in the similarity.
- *cos+TD+avg-lnk*: Time Decay coefficient $\alpha = 1$ in equation 13, which means the similarity between two stories will be decayed to $1/e$ if they are at different ends of the topic.
- *cos+N(T)+avg-lnk*: Use the number of true events to control the agglomerative clustering algorithm. When the number of clusters is fewer than that of truth events, stop merging clusters.
- *cos+N(T)+T+avg-lnk*: similar to *N(T)* but also stop agglomeration if the maximal similarity is below the threshold T.
- *cos+TD:+N(T)+avg-lnk*: similar to *N(T)* but the similarities are decayed, $\alpha = 1$ in equation 13.
- *cos+TD+N(T)+T+avg-lnk*: similar to *TD+N(Truth)* but calculation halts when the maximal similarity is smaller than the threshold T.

Our experiments demonstrate that single link and complete link similarities perform worse than average link, which is reasonable since average link is less sensitive to one or two story pairs. We had expected locations and person names to improve the result, but it is not the case. Analysis of topics shows that many on-topic stories share the same locations or persons irrespective of the event they belong to, so these features may be more useful in identifying topics rather than events. Time decay is successful because events are temporally localized, i.e., stories discussing the same event tend to be adjacent to each other in terms of time. Also we noticed that providing the number of true events improves the performance since it guides the clustering algorithm to get correct granularity. However, for most applications, it is not available. We used it only as a "cheat" experiment for comparison with other algorithms. On the whole, time decay proved to the most powerful feature besides cosine similarity on both training and test sets.

7.2 Dependencies

In this subsection, our goal is to model only dependencies. We use the true mapping function f and by implication the true events V. We build our dependency structure E' using all the five models described in section 6.2. We first train our models on the 26 training topics. Training involves learning the best threshold T for each of the models. We then test the performances of all the trained models on the 27 test topics. We evaluate our performance

using the average values of Dependency Precision (DP), Dependency Recall (DR) and Dependency F-measure (DF). We consider the complete-link model to be our baseline since for each event, it trivially considers all earlier events to be parents.

Table 4 lists the results on the training set. We see that while all the algorithms except MST outperform the baseline complete-link algorithm , the nearest Parent algorithm is statistically significant from the baseline in terms of its DF-value using a one-tailed paired T-test at 95% confidence level.

Model	best T	DP	DR	DF	P-value
Nearest Parent	0.025	0.55	0.62	0.56	0.04*
Best Similarity	0.02	0.51	0.62	0.53	0.24
MST	0.0	0.46	0.58	0.48	-
Simple Thresh.	0.045	0.45	0.76	0.52	0.14
Complete-link	-	0.36	0.93	0.48	-

Table 4: Results on the training set: Best T is the optimal value of the threshold T. * indicates the corresponding model is statistically significant compared to the baseline using a one-tailed, paired T-test at 95% confidence level.

In table 5 we present the comparison of the models on the test set. Here, we do not use any tuning but set the threshold to the corresponding optimal values learned from the training set. The results throw some surprises: The nearest parent model, which was significantly better than the baseline on training set, turns out to be worse than the baseline on the test set. However all the other models are better than the baseline including the best similarity which is statistically significant. Notice that all the models that perform better than the baseline in terms of DF, actually sacrifice their recall performance compared to the baseline, but improve on their precision substantially thereby improving their performance on the DF-measure.

We notice that both simple-thresholding and best similarity are better than the baseline on both training and test sets although the improvement is not significant. On the whole, we observe that the surface-level features we used capture the dependencies to a reasonable level achieving a best value of 0.72 *DF* on the test set. Although there is a lot of room for improvement, we believe this is a good first step.

Model	DP	DR	DF	P-value
Nearest Parent	0.61	0.60	0.60	-
Best Similarity	0.71	0.74	0.72	0.04*
MST	0.70	0.68	0.69	0.22
Simple Thresh.	0.57	0.75	0.64	0.24
Baseline (Complete-link)	0.50	0.94	0.63	-

Table 5: Results on the test set

7.3 Combining Clustering and Dependencies

Now that we have studied the clustering and dependency algorithms in isolation, we combine the best performing algorithms and build the entire event model. Since none of the dependency algorithms has been shown to be consistently and significantly better than the others, we use all of them in our experimentation. From the clustering techniques, we choose the best performing *Cos+TD*. As a baseline, we use a combination of the baselines in each components, i.e., *cos* for clustering and *complete-link* for dependencies.

Note that we need to retrain all the algorithms on the training set because our objective function to optimize is now *JF*, the joint F-measure. For each algorithm, we need to optimize both the clustering threshold and the dependency threshold. We did this empirically on the training set and the optimal values are listed in table 6.

The results on the training set, also presented in table 6, indicate that *cos+TD+Simple-Thresholding* is significantly better than the baseline in terms of the joint F-value JF, using a one-tailed paired T-test at 95% confidence level. On the whole, we notice that while the clustering performance is comparable to the experiments in section 7.1, the overall performance is undermined by the low dependency performance. Unlike our experiments in section 7.2 where we had provided the true clusters to the system, in this case, the system has to deal with deterioration in the cluster quality. Hence the performance of the dependency algorithms has suffered substantially thereby lowering the overall performance.

The results on the test set present a very similar story as shown in table 7. We also notice a fair amount of consistency in the performance of the combination algorithms. *cos+TD+Simple-Thresholding* outperforms the baseline significantly. The test set results also point to the fact that the clustering component remains a bottleneck in achieving an overall good performance.

8. DISCUSSION AND CONCLUSIONS

In this paper, we have presented a new perspective of modeling news topics. Contrary to the TDT view of topics as flat collection of news stories, we view a news topic as a relational structure of events interconnected by dependencies. In this paper, we also proposed a few approaches for both clustering stories into events and constructing dependencies among them. We developed a time-decay based clustering approach that takes advantage of temporal-localization of news stories on the same event and showed that it performs significantly better than the baseline approach based on cosine similarity. Our experiments also show that we can do fairly well on dependencies using only surface-features such as cosine-similarity and time-stamps of news stories as long as true events are provided to the system. However, the performance deteriorates rapidly if the system has to discover the events by itself. Despite that discouraging result, we have shown that our combined algorithms perform significantly better than the baselines.

Our results indicate modeling dependencies can be a very hard problem especially when the clustering performance is below ideal level. Errors in clustering have a magnifying effect on errors in dependencies as we have seen in our experiments. Hence, we should focus not only on improving dependencies but also on clustering at the same time.

As part of our future work, we plan to investigate further into the data and discover new features that influence clustering as well as dependencies. And for modeling dependencies, a probabilistic framework should be a better choice since there is no definite answer of yes/no for the causal relations among some events. We also hope to devise an iterative algorithm which can improve clustering and dependency performance alternately as suggested by one of the reviewers. We also hope to expand our labeled corpus further to include more diverse news sources and larger and more complex event structures.

Acknowledgments

We would like to thank the three anonymous reviewers for their valuable comments. This work was supported in part by the Center

Model	Cluster T	Dep. T	CP	CR	CF	DP	DR	DF	JF	P-value
cos+TD+Nearest-Parent	0.055	0.02	0.51	0.53	0.49	0.21	0.19	0.19	0.27	-
cos+TD+Best-Similarity	0.04	0.02	0.45	0.70	0.53	0.21	0.33	0.23	0.32	-
cos+TD+MST	0.04	0.00	0.45	0.70	0.53	0.22	0.35	0.25	0.33	-
cos+TD+Simple-Thresholding	0.065	0.02	0.56	0.47	0.48	0.23	0.61	0.32	0.38	0.0004*
Baseline (cos+Complete-link)	0.10	-	0.58	0.31	0.38	0.20	0.67	0.30	0.33	-

Table 6: Combined results on the training set

Model	CP	CR	CF	DP	DR	DF	JF	P-value
cos+TD+Nearest Parent	0.57	0.50	0.50	0.27	0.19	0.21	0.30	-
cos+TD+Best Similarity	0.48	0.70	0.54	0.31	0.27	0.26	0.35	-
cos+TD+MST	0.48	0.70	0.54	0.31	0.30	0.28	0.37	-
cos+TD+Simple Thresholding	0.60	0.39	0.44	0.32	0.66	0.42	0.43	0.0081*
Baseline (cos+Complete-link)	0.66	0.27	0.36	0.30	0.72	0.43	0.39	-

Table 7: Combined results on the test set

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.

9. 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, A. Feng, and A. Bolivar. Flexible intrinsic evaluation of hierarchical clustering for tdt. volume In the Proc. of the ACM Twelfth International Conference on Information and Knowledge Management, pages 263–270, Nov 2003.

[3] James Allan, editor. *Topic Detection and Tracking:Event based Information Organization*. Kluwer Academic Publishers, 2000.

[4] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 10–18. ACM Press, 2001.

[5] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics(HLT-NAACL)*, pages 113–120, 2004.

[6] D. Lawrie and W. B. Croft. Discovering and comparing topic hierarchies. In *Proceedings of RIAO 2000 Conference*, pages 314–330, 1999.

[7] David D. Lewis and Kimberly A. Knowles. Threading electronic mail: a preliminary study. *Inf. Process. Manage.*, 33(2):209–217, 1997.

[8] Juha Makkonen. Investigations on event evolution in tdt. In *Proceedings of HLT-NAACL 2003 Student Workshop*, pages 43–48, 2004.

[9] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 521–528. IEEE Computer Society, 2001.

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