ON-LINE NEW EVENT DETECTION AND TRACKING IN A MULTI-RESOURCE ENVIRONMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hakan Kurt

September, 2001

ABSTRACT

ON-LINE NEW EVENT DETECTION AND TRACKING IN A MULTI-RESOURCE ENVIRONMENT

Hakan Kurt
M.S. in Computer Engineering
Supervisor: Prof. Dr. H. Altay Güvenir
September, 2001

As the amount of electronically available information resources increase, the need for information also increases. Today, it is almost impossible for a person to keep track all the information resources and find new events as soon as possible. In this thesis, we present an on-line new event detection and tracking system, which automatically detects new events from multiple news resources and immediately start tracking events as they evolve. Since we implemented the on-line version of event detection approach, the novelty decision about a news story is done before processing the next one. We also present a new threshold, called support threshold, used in detection process to decrease the number of new event alarms, that are caused by informative and one-time-only news. The support threshold can be used to tune the weights of news resources. We implemented the tracking phase as an unsupervised learning process, that is, detected events are automatically tracked by training the system using the first news story of an event. Since events evolve over time, an unsupervised adaptation is used to retrain the tracking system in order to increase the tracking system performance. Adaptation is achieved by adding predicted documents to the training process. From the corpus observations, we conclude that one news story can be associated to more than one event. For this reason, the tracking system can relate a news story to more than one event. The on-line new event detection and tracking system has been tested on the Reuters news feed, available on the Internet. The Reuters news feed, that we used, comprises four independent news resources. The news stories are in Turkish.

Keywords: Event detection, event tracking, information retrieval.

ÖZET

ÇOK KANALLI KAYNAK ORTAMINDA ÇEVRİMİÇİ YENİ OLAY BELİRLEME VE TAKİBİ

Hakan Kurt
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. H. Altay Güvenir
Eylül, 2001

Elektronik ortamdaki bilgi miktarı arttıkça bilgiye olan ihtiyaç da artmaktadır. Bugün, bir kişinin bütün bigi kaynaklarını takip etmesi ve yeni olaylari olabildiğince kısa zamanda bulması hemen hemen imkansızdır. Bu tezde, birden fazla haber kaynağından otomatik olarak yeni olayları belirleyen ve olaylar geliştikçe onları anında takibe başlayan, çevrimiçi yeni olay belirleme ve takip sistemi sunuyoruz. Olay belirleme yaklaşımının çevrimiçi versiyonunu uyguladığımız için, haber metni hakkındaki yenilik kararı bir sonraki haber işleme alınmadan yapılmaktadır. Ayrıca, destek eşiği olarak adlandırdığımız, belirleme işleminde, bilgi verici ve bir seferlik haberler tarafından sebep olunan yeni olay alarmlarının sayısını azaltmak maksadıyla kullanılan yeni bir eşiği de tanıtıyoruz. Destek eşiği, haber kaynaklarının ağırlığını ayarlamak için de kullanılabilir. Takip etme safhasını, denetlemesiz öğrenme metodu şeklinde uyguladık, yani belirlenen olaylar bir olayın ilk haberini kullanarak otomatik olarak takip edilir. Olaylar zaman geçtikce geliştiği için, takip etme sisteminin performansını arttırmak maksadıyla, bir denetlemesiz adaptasyon yöntemi takip sistemini tekrar eğitmek için kullanılır. Adaptasyon tahmin edilen dokümanların eğitim işlemine eklenmesiyle sağlanır. Toplanan haberlerin incelenmesinden, bir haber metninin birden fazla olaydan bahsedebileceği sonucuna vardık. Bu sebeple, takip sistemi bir haber metnini birden fazla olayla ilişkilendirecek şekilde uygulandı. Çevrimiçi yeni olay belirleme ve takip sistemi, İnternet'te mevcut olan, Reuters haber kaynağında testedildi. Kullandığımız Reuters haber kaynağı dört bağımsız haber kaynağını içerir ve haberleri Türkçe'dir.

Anahtar sözcükler: Olay belirleme, olay takibi, bilgi erişimi.

Türk Silahlı Kuvvetleri'ne,
Eşim Nazan'a
Anneme, Babama ve Kardeşime.

Acknowledgement

I am very grateful to my supervisor, Prof. Dr. H. Altay Güvenir for his invaluable support, guidance and motivation.

I also would like to thank my thesis committee members Prof. Dr. Fazlı Can and Assist. Prof. Dr. Attila Gürsoy for their valuable comments to improve this thesis.

I would like to thank to my friends Türker Yılmaz, Gültekin Arabacı, Sezgin Abalı, İlker Yoncacı and Erdoğan Bıkmaz for their valuable help during the preparation phase of this thesis.

I would like to thank to my sister, Ruhan Kurt, and my parents, Makbule and Mehmet Kurt, for their moral support.

Finally, I would like to thank to my wife, Nazan Kurt, for her moral support, patience and love during the first year of our marriage.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

SYMBOLS

f_{ik}	: Term frequency of word i in document k.
M	: Number of words in the corpus.
N	: Number of documents in training corpus
N_t	: Number of training documents for event tracking
$sim(k, q)$	: Similarity between document k and document (cluster) q.
w_{ik}	: Weight of word i in document k.

ABBREVIATIONS

ASR	: Automatic Speech Recognition
BORG	: Best Overall Result Generator
CMU	: Carnegie Mellon University
DARPA	: Defence Advanced Research Project Agency
DET	: Decision Error Trade-off
dtree	: Decision-Tree Induction
IDF	: Inverse Document Frequency
IR	: Information Retrieval
kNN	: k-Nearest Neighbor
NIST	: National Institute of Standards and Technology
RF	: Relevance Feedback
TDT	: Topic Detection and Tracking
TF	: Term Frequency
TIDES	: Translingual Information Detection, Extraction, and Summarization
UMass	: University of Massachusetts, Amherst
UPenn	: University of Pennsylvania

Chapter 1

Introduction

The rapidly-growing amount of electronically available information threatens to overwhelm human attention, raising new challenges for information retrieval technology. Traditional query-based retrieval is useful when one knows more precisely the nature of the events or facts one is seeking, and less useful when one wants specific information but can only formulate a larger category-query sharing few if any terms with the potentially most useful texts. In short, information retrieval based on immediate-content-focused queries is often insufficient for obtaining a variety of relevant stories and tracking the gradual evolution of events through time [26].

It would be desirable for an intelligent system to automatically detect significant events from large volume of news stories with a desired level of abstraction, alert the new events as they happen, and track events as they evolve.

In this thesis, we discuss and evaluate solutions to on-line new event detection from chronologically-ordered streams of news stories coming, from multiple resources, and track events over time.

1.1 Event Analysis

Event means some unique thing that happens at some point in time. The notion of an event differs from a broader category of events both in spatial/temporal localization and in specificity [3]. Specific time and place information differs event from the broader category "topic". For example, *Kocaeli earthquake in 1999* is an event but not a topic, while the term *earthquakes* refers to a topic but not an event.

From a journalist's perspective, a news story about an event will typically specify :

- When the event occurred?
- Who was involved?
- Where it took place?
- How it happened?
- The impact, significance, or consequence of the event on the intended audience.

However, as an event evolves, many of these properties are either not initially known, or be assumed to be known by the audience and therefore are not referenced within the text of documents relating to the same event. As a result, the lack of certain event properties and the appearance of new lexical features within documents relating to the same event should be expected as the event evolves [14, 15].

Several patterns emerged from observations of temporal event distributions [3, 26, 28] and also several properties of events derived:

- Events might be unexpected or expected.
- Events are often associated with news bursts.

- News stories discussing the same event tend to be temporally proximate.
- A time gap between bursts of topically similar stories is often an indication of different events.
- A significant vocabulary shift and rapid changes in term frequency distribution are typical of stories reporting a new event.
- Events are typically reported in a relatively brief time window (e.g. 1-4 weeks) and contain fewer reports than broader topics.

1.2 News Analysis

A detailed investigation of news is also required, since events are told in news stories.

News can be defined as *a new information about a subject of some public interest that is shared with some portion of the public* [19].

News is the unusual. It is also something fresh, something that people have not heard before and, crucially, is of interest to readers. That means not just matters which affect the public or have an impact on public life, but also what is *of* interest to the public [16].

For most consumers, most of the time, news is what is happening locally. Like any business person, the news provider serves his or her basic market first [12]. All the news providers in the world give priority to the events in their country, unless a very important event happens in another part of the world. Also, news providers give information about the local events such as local crimes, theaters, concerts etc., since, people want to learn what is happening around them.

From the definitions given above, it can be concluded that news stories which give information about an event are only the subset of news. This means that we need to find a system, which can select only the news stories that belongs to an event. Since events are often associated with news bursts, waiting for other news stories which support the event can be a solution to the problem. Thus, only the events that are supported by more than one news stories are known by the user, and the others are hidden.

Some properties of news stories make the detection and tracking process difficult:

- A news story might be about one or more events. We have observed many news stories, in the corpus, which give information about more than one event. As a result, we conclude that one news story may give information about one or more than one event.
- When a new event happens, all of the news stories give information about that event only (answers to when, where, how and who type of questions). After a while, the amount of new information about the event decreases. Subsequently, news stories about the event usually turn to the discussions about the event with a broader perspective, that is topic related. For example, after a skyjack event, the news stories usually turn to the discussions about the security considerations of the airports, skyjack events in the past etc.

1.3 Event Detection Methods

New event detection is an unsupervised learning task, sub-divided into two forms. One is *retrospective detection*, which entails the discovery of previously unidentified events in a chronologically-ordered accumulation of stories. The other is *on-line detection*, where the goal is to identify the onset of new events from live news feeds in real-time. Both forms of detection intentionally lack advance knowledge of novel events, but may have access to unlabelled historical news stories for use as contrast sets [26].

The new event detection system has two on-line models of operation: *immediate* and *delayed*. In immediate mode, a strict real-time application is assumed, and the system indicates whether the current document contains or does not contain discussion of a new event before looking at the next document. In delayed mode, classification decisions deferred for a prespecified time interval. For example, the system could collect news throughout the day and provide the user with new events at the end of the day [14].

The detection system developed for this thesis works in immediate mode but the new event alarms are deferred until a userdefined support threshold is exceeded.

1.4 Event Tracking

Event tracking is defined to be the task of associating incoming stories with events known to the system. In the tracking task, a target event is given, and each successive story must be classified as to whether or not it discusses the target event [3]. The objective of event tracking is to correctly classify all of the subsequent stories.

We have the same objective, but a different approach is applied. According to the previous studies, a news story can be related to only one event, since there are only small number of counter examples in their corpus. However, we found that this assumption is too strong for our corpus. As a result of this fact, we implemented our tracking system with a weaker assumption: the first news story of an event cannot be related to the existing events in the past, but the other news stories can be related to more than one event.

1.5 Overview of the Thesis

In this thesis, solutions to the on-line new event detection and tracking problem for multi-resource news feed case is introduced. Our approach to the on-line new event detection and tracking problem is different from the other approaches that we found in the literature. The differences are shown in Table 1.1.

Table 1.1: Comparison of our approach with previous work.

Previous Approaches	**Our Approach**
Divide the problem into two parts; on-line new event detection and event tracking.	On-line new event detection and event tracking are done together.
For every new event found by the system, an alarm is issued.	New event alarm is issued whenever a user-defined supporting document is found.
Resource information of the news stories is NOT used.	Resource information is used in support threshold calculations.
Handle event tracking problem as a supervised learning task.	User interaction is NOT needed. All new events are tracked.
User gives a number of sample stories to train the tracking system (usually 4).	Number of training stories is one that is the first story of an event.
One document can NOT belong to more than one event.	One document can belong to more than one event.
No work done about Turkish news.	Our implementation is done by using Turkish news stories.

We define the on-line new event detection and tracking problem as:

On-line new event detection and tracking is to find the first document that talks about an event which is not previously reported, warn the user about the new event after a given amount of supporting document are observed from one or multiple resources, and track each event without user interaction.

In order to test our approach, we created a corpus which spans the period from January 2,2001 to March 31,2001 with 46,530 Turkish news stories from the Reuters newswire. Corpus consists of four different news resources of Reuters, which are *Anadolu News Agency*, *Dünya News Agency*, *İstanbul Stock Exchange Company News* and *Reuters News* itself. Anadolu News Agency news stories contain political, sports, economical, cultural, local and other types of events. Dünya News Agency news stories usually contain economical and political events, and long comments about economical and political events. İstanbul Stock Exchange

Company News contains only short stock market news stories. Reuters News contains economical news stories in general. From 46,530 news, we selected 15 events which form 1322 (2.8% of the corpus) stories to test our approach.

In the next chapter, information about previous work about on-line new event detection and tracking is given. In Chapter 3, our solutions for on-line new event detection part of the problem with detailed information about the document representation, similarity and threshold calculations are presented. Than, we present our event tracking approach. At the end of the Chapter 3, a complete algorithm of our system is given. Chapter 4 presents the evaluation methodology, effectiveness measures, and results of our experiments respectively. Conclusions of the thesis and plans for future work are discussed in Chapter 5.

Detailed information of our corpus is given in the Appendix.

Chapter 2

Related Work

The main motivation of this thesis is affected from the studies and results of a research called Topic Detection and Tracking (TDT). TDT is a Defence Advanced Research Project Agency (DARPA)-sponsored initiative to investigate the state of the art in finding and following new events in a stream of broadcast news stories.

The basic idea for TDT originated in 1996. A pilot study laid the essential groundwork in 1997, producing a small corpus and establishing feasibility. During 1998 and 1999, TDT research blossomed, with new and more challenging tasks, many more participating sites, and considerably larger multilingual corpora (adding automatic speech recognition (ASR) data in 1998 and Chinese data in 1999) [21].

TDT research is continuing under the new DARPA program known as TIDES (Translingual Information Detection, Extraction, and Summarization) with 28 organizations and universities [13].

For research purposes, TIDES is placing primary emphasis on English, Chinese, and Arabic – three important, challenging, and very different languages. Some other groups work on Korean, Japanese, or Spanish.

The approaches described below are taken from the articles and reports of this project group.

2.1 On-line New Event Detection Approaches

In this section, we will give the approaches to solve the on-line new event detection problem. Only the approaches about immediate mode are given, since our implementation is on immediate mode of on-line event detection[1].

2.1.1 The CMU Approach

The CMU (Carnegie Mellon University) approach uses conventional vector space model to represent the documents and traditional clustering techniques in information retrieval to represent the events [17].

Each document is represented using a vector of weighted terms which can be either words or phrases. In choosing a term weighting system, low weights should be assigned to high-frequency words that occur in many documents of a collection, and high weights to terms that are important in particular documents but unimportant in the remainder of the collection. A well-known term weighting system following that prescription assigns weight w_{ik} to term T_i in document D_k in proportion to the frequency of occurrence of the term in D_k, and in inverse proportion to the number of documents to which the term is assigned [17]. Such a weighting system is known as a TFxIDF (Term Frequency times Inverse Document Frequency) weighting system. To allow a meaningful final retrieval similarity, it is convenient to use a length normalization factor as part of the term weighting formula. Under these considerations, for term weighting, "ltc" (logarithm of the term frequency) version of the TF-IDF schema [1, 7] is used as:

$$w_{ik} = \frac{(1 + \log_2(f_{ik})) * log_2(\frac{N}{n_i})}{\sqrt{\sum_{j=1}^{M}[(1 + \log_2(f_{ik})) * log_2(\frac{N}{n_j})]^2}} \tag{2.1}$$

where w_{ik} is the weight of word i in document k and f_{ik} be the frequency of the word i in document k, N the number of documents in the training corpus, n_i the number of times word i occurs in the training corpus, and M corresponds to the

[1]Small amount of deferral period before a decision is considered as in immediate mode. This amount is taken as less than 10 news stories for TDT implementations.

number of words in the corpus. If the word does not appear in the training corpus ($n_i = 0$ or $n_j = 0$), a default value 1 is used instead of these values.

For story and cluster representation, CMU uses the conventional vector space model. A story is presented as a vector whose dimensions are the stemmed unique terms in the corpus, and whose elements are the term (word or phrase) weights in the story. A cluster is represented using a *prototype vector* (or *centroid*) which is the normalized sum of story vectors in the cluster. The similarity of two stories k and q is defined as the cosine value of the corresponding story vectors:

$$sim(k,q) = \frac{\sum_{j=1}^{M} w_{jk} * w_{jq}}{\sqrt{(\sum_{j=1}^{M} w_{jk}^2) * (\sum_{j=1}^{M} w_{jq}^2)}} \tag{2.2}$$

where $sim(k,q)$ is the cosine similarity value, w_{jk} is the weight of word j in document k, w_{jq} is the weight of word j in document q,M corresponds to the number of words in the corpus, $\sum_{j=1}^{M} w_{jk}^2$ is the sum of squares of the word weights in document k, and $\sum_{j=1}^{M} w_{jq}^2$ is the sum of squares of the word weights in document q. Similarly, the similarity of two cluster is defined as the cosine value of the corresponding prototype vectors (i.e., cluster representatives, or centroids). As a clustering algorithm, incremental (single-pass) clustering algorithm with a time window is used (see Figure 2.1 for detailed information about the algorithm).

As it is described in Figure 2.1, the single-pass clustering algorithm is straightforward. It sequentially processes the input documents, one at a time, and grows clusters incrementally. A new document is absorbed by the most similar cluster in the past if the similarity between the document and the cluster is above a preselected *clustering threshold* (t_c); otherwise, the document is treated as the seed of a new cluster. By adjusting the threshold, one can obtain clusters at different levels of granularity.

CMU applied two types of time penalty functions to the similarity values between the current document x and cluster c. The purpose of the first function is to get the similarity values of the documents in the time window by weighting the similarity scores uniformly:

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

1. The documents are processed sequentially.
2. The representation for the first document becomes the cluster representative of the first cluster.
3. Each subsequent document is matched against all cluster representatives existing at its processing time.
4. A given document is assigned to one cluster (or more if overlap is allowed) according to some similarity measure.
5. When a document is assigned to a cluster, the representative for that cluster is recomputed.
6. If a document fails a certain similarity test, it becomes the cluster representative of a new cluster.

Figure 2.1: Single-pass clustering algorithm

where $sim(x, c)$ is the cosine similarity.

The other one is a linear decaying weight function. The purpose of this function is to decrease the influence of the clusters as the number of documents between a cluster c and the document x increase:

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

where i is the number of documents between the current document x and the most recent member-document in cluster c, and m is the number of documents prior to x in time window.

Novelty decision is made by thresholding the maximum similarity score between the document and any cluster in the time window [3, 26, 28]. The confidence score for the novelty decision is defined as:

$$score(i) = 1 - \arg\max_{c}\{sim(i, c)'\} \tag{2.3}$$

where i is the current document and $sim(i, c)'$ is the similarity value between document i and any cluster in the time window.

2.1.2 The UMass Approach

The new-event detection algorithm in the UMass (University of Massachusetts, Amherst) was implemented by combining the ranked-retrieval mechanisms of Inquery [9], a feature extraction and selection process based on relevance feedback [2], and the routing architecture of InRoute [8].

Events and documents are represented by queries. A modified version of the single-pass clustering algorithm (given in Figure 2.1) is used for new event detection. This algorithm processes each new document on the stream sequentially, as shown in Figure 2.2.

1. Use feature extraction and selection techniques to build a query representation to define the document's content.
2. Determine the query's initial threshold by evaluating the new document with the query.
3. Compare the new document against previous queries in memory.
4. If the document does not trigger any previous query by exceeding its threshold, flag the document as containing a new event.
5. If the document triggers an existing query, flag the document as not containing a new event.
6. (Optional) Add the document to the agglomeration list of queries it triggered.
7. (Optional) Rebuild existing queries using the document.
8. Add new query to memory.

Figure 2.2: On-line new event detection algorithm of UMass [6, 14, 15].

The document representation used in the system is a set of *belief* values corresponding to each feature specified in a query. Belief values are produced by

Inquery's belief function, which is composed of TF-IDF. For any instance of document d and collection c :

$$d_i = belief(q_i, d, c) = 0.4 + 0.6 * tf * idf \tag{2.4}$$

where

$$tf = \frac{t}{t + 0.5 + 1.5 * \frac{dl}{avg\_dl}} \tag{2.5}$$

$$idf = \frac{\log(\frac{|c|+0.5}{df})}{\log(|c| + 1)} \tag{2.6}$$

and t is the number of times feature q_i occurs in the document, *df* (document frequency) is the number of documents in which the feature appears in the auxiliary corpus, *dl* is the document's length, *avg_dl* is the average document length in the auxiliary corpus, and $|c|$ is the number of documents in the auxiliary corpus.

An automatic process creates a classifier from single or multiple documents. The classifier formulation process has three main steps: feature selection, weight assignment, and threshold estimation. The selected features and weights are used to construct a classifier using Inquery's query syntax.

The feature selection begins with collecting statistics from the words appearing in the training documents. The words that do not play a role in the categorization of a news document are called *stopwords*. Stopwords are removed from the documents, than the remaining words are sorted by the following measure:

$$\frac{r}{R} - \frac{nr}{NR} > 0 \tag{2.7}$$

where R is the number of relevant documents and NR is the number of non-relevant documents in the training sample. The values r and nr are the number of documents in the corresponding relevant and non-relevant training sample containing the word. The top n words are used for weight assignment as:

$$q_{i,k} = c_1 * tf_{rel} - c_2 * tf_{nonrel} \tag{2.8}$$

where tf_{rel} is average *tf* score (Equation 2.5) for the word in relevant documents, and tf_{nonrel} is the average *tf* score for the word in non-relevant documents. The constants c_1 and c_2 are determined empirically, and it is found that setting $c_1 = c_2 = 0.5$ works well.

A similarity value is calculated when comparing a document d to a query q using the #WSUM operator of Inquery:

$$sim(q_i, d_j) = \frac{\sum_{i=1}^{N} q_{i,k}.d_{j,k}}{\sum_{i=1}^{N} q_{i,k}} \tag{2.9}$$

If a classifier is created at time i, that is, when the last relevant training document arrives, then the resulting classifier's threshold for a document arriving at a later time j is computed as:

$$threshold(q_i, d_j) = 0.4 + \theta * (sim(q_i, d_i)) - 0.4) + \beta * (date_j - date_i) \tag{2.10}$$

where $sim(q_i, d_i)$ is the similarity value between the classifier and the document from which it was formulated (Equation 2.9), and 0.4 is an Inquery constant. The value of $(date_j - date_i)$ is the number of days between the arrival of document d_j and the formulation of the classifier q_i. The values for θ and β control the new event classification decisions.

When deciding whether a new event has arrived, decision scores are used as:

$$decision(q_i, d_j) = sim(q_i, d_j) - threshold(q_i, d_j) \tag{2.11}$$

Decision scores greater than 0 imply that documents d_i and d_j are similar in content, and thus document d_j does not discuss a new event [3, 6, 14, 15].

2.1.3 Dragon Approach

Dragon used the k-means algorithm to solve to on-line detection problem, by executing only the first pass of the algorithm [3]. Following this procedure, the first story in the corpus defines an initial cluster. The remaining stories in the corpus are processed sequentially; for each one, the "distance" to each of the existing clusters is computed. A story is inserted into the closest cluster unless this distance is greater than a threshold, in which case a new cluster is created.

The decision to create a new cluster is equivalent to declaring the appearance of a new event.

Dragon used a new measure to compute the distances between a given story and existing clusters by smoothing the cluster distribution with a background distribution, and then preventing the cluster from being "dragged" by the story distribution. Two improvements were also made: a story-background distance was subtracted from the story-cluster distance (to compensate for the fact that small clusters tend to look a lot like background after smoothing), and a decay term was introduced to cause clusters to have a limited duration in time. This term is just a decay parameter times the distance between the number of the story represented by the distribution of the story count for a word and the number midway between the first and last stories in the cluster.

By adjusting the decay parameter and the overall threshold the on-line detection system can be tuned.

2.1.4 UPenn Approach

UPenn (University of Pennsylvania) used the *idf*-weighted cosine coefficient [18]. It is a document similarity metric where documents are represented as vectors of an n-dimensional space, where n is the number of unique terms in the database. For their implementation, they weighted only the topic vector (vector which represents the event) by *idf* and left the story vector under test unchanged. The resulting calculation for the similarity measure becomes:

$$sim(k,q) = \frac{\sum_{j=1}^{M} tf_{jk} * tf_{jq} * idf(w)}{\sqrt{(\sum_{j=1}^{M} tf_{jk}^2) * (\sum_{j=1}^{M} tf_{jq}^2)}} \tag{2.12}$$

where $sim(k,q)$ is the cosine similarity value between documents k and q, tf_{jk} is the term frequency of word j in document k, tf_{jq} is the term frequency of word j in document q, $\sum_{j=1}^{M} tf_{jk}^2$ is the sum of squares of the term frequencies in document k, and $\sum_{j=1}^{M} tf_{jq}^2$ is the sum of squares of the term frequencies in document q.

Of the feature selection methods, they selected not the best but a simpler method between the approaches that they implemented: For each story, word counts (*tf*) are sorted, than n most frequent ones are kept. They set the number of selected features to 50.

As a system parameter, a deferral period is defined to be the number of files (each containing multiple stories) the system is allowed to process before it associates an event with the stories contained in the files.

The UPenn approach uses the single-linkage (nearest neighbor) clustering method to represent each event. This method begins with all stories in their own singleton clusters. Two clusters are merged if the similarity between any story of the first cluster and any story of the second cluster exceeds a threshold.

To implement the clustering, the UPenn approach takes the stories of each deferral period and created an inverted index. Then each story, in turn, is compared with all preceding stories (including those from previous deferral periods). When the similarity metric for two stories exceeds a threshold their clusters are merged. The clusters of earlier deferral periods cannot merge since they have already been reported. If a story cannot be merged with an existing cluster, it becomes a new cluster which means a new event.

2.1.5 BBN Technologies' Approach

The BBN approach uses an incremental k-means algorithm for clustering stories. For comparing stories, it utilizes a probabilistic document similarity metric and a traditional vector-space metric.

Although it is similar, the clustering algorithm they used is not precisely a k-means algorithm, because the number of clusters k is not given beforehand. This algorithm involves an iteration through the data that the system is permitted to modify and making appropriate changes during each iteration (see Figure 2.3).

As a similarity metric, they utilize a probabilistic metric called the BBN topic

1. Use the incremental clustering algorithm (see Figure 2.1 for details) to process stories up to the end of the current modifiable window.
2. Compare each story in the modifiable window with the old clusters to determine whether each story should be merged with that cluster or used as a seed for a new cluster.
3. Modify all the clusters at once according to the new assignments.
4. Iterate steps (2)-(3) until the clustering does not change.
5. Look at the next few stories and go to (1).

Figure 2.3: On-line new event detection algorithm of BBN Technologies

spotting metric which is derived from Bayes' Rule:

$$p(C|S) \approx p(C) \cdot \prod_{n} \frac{p'(s_n|C)}{p(s_n)} \tag{2.13}$$

where $p(C|S)$ is the probability that the given story belongs to the cluster C, $p(C)$ is the *a priori* probability that any new story will be relevant to cluster C, $p(s_n)$ is the occurrence probability of a story word s_n, and $p'(s_n|C)$ is the smoothed probability that a word in a story on the topic represented by cluster C would be s_n. Smoothing is done because of the zero probability of unobserved words for the topic as:

$$p'(s_n|C) = \alpha \cdot p(s_n|C) + (1 - \alpha) \cdot p(s_n) \tag{2.14}$$

The BBN approach models $p(s_n|C)$ with a two-state mixture model, where one state is a distribution of the words in all of the stories in the group, and the other state is a distribution from the whole corpus.

There are two types of metrics that are useful for the clustering algorithm: selection metric, which is the maximum probability value of the BBN topic spotting metric and thresholding metric, which is the binary decision metric to combine the story with a cluster. A score normalization method is used to produce improved scores [20].

2.1.6 Summary of New Event Detection Approaches

Summary of the new event detection approaches are shown in Figure 2.4, where N/A means no information available.

	CMU	UMass	Dragon	UPenn	BBN
Words are weighted by using	ltc version of TF-IDF	TF-IDF	N/A	TF-IDF	probabilistic
Documents are represented by	vector space model	query representation	N/A	vector space model	vector space model
Events are represented by clusters using	single-pass clustering	single-pass clustering	k-means clustering	nearest neighbor clustering	incremental k-means clustering
Similarity between a given document to a cluster is calculated by	cosine similarity	previous queries run on new document	distance between document to clusters	cosine similarity	probabilistic similarity
Time windowing	used	used	used	N/A	N/A

Figure 2.4: Summary of New Event Detection Approaches

2.2 Event Tracking Approaches

Event tracking, according to the TDT, can be considered as a text categorization problem subject to the following constraints [25]:

- Each event of interest is defined by a set of positive instances (documents) that are *manually* identified before tracking starts; no other knowledge is available.
- As soon as a new document arrives, a binary (YES/NO) decision must made by the tracking system with respect to each defined event.
- Any document preceding the document being evaluated may be used as training data. However, only the previously-identified positive instances are labelled; the rest of the documents are not, although some of these unlabelled documents may actually be positive instances.
- When training on an event, relevance judgements for other events are assumed to be unknown.

As a result, event tracking for TDT is a supervised learning task. A target event is given by N_t number of training documents, and each successive story must be classified as to whether or not it discusses the target event. The tracking task is to correctly classify all of the incoming stories [3].

The official event tracking evaluation of TDT restricted the number (N_t) of positive training examples per event to be 1,2,4,8 an 16, respectively [26].

2.2.1 CMU Approach

Researchers in CMU developed three methods for tracking events: a *k-Nearest Neighbor* (kNN) classifier, a *Decision-Tree Induction* (dtree) classifier and Rocchio [3, 24, 25, 26].

2.2.1.1 K-Nearest Neighbor Classification (kNN)

This method adapted conventional M-way classification kNN to the 2-way classification problem of event tracking by constructing a 2-way kNN for each event. The system converts an input story into a vector as it arrives and compares it to the training stories, and select the k nearest neighbors based on the cosine similarity between the input story and the training stories. The documents, given by the user to track an event, are used as the positive training stories, and the other stories in the corpus are used as negative training documents. The *confidence score* for a YES prediction on the input story is computed by summing the similarity scores for the positive and negative stories respectively in the k-neighborhood, and taking the difference between the two sums:

$$s_1(x, k, D) = \sum_{y \in P_k} \cos(x, y) - \sum_{z \in Q_k} \cos(x, z) \tag{2.15}$$

where x is the test document; y (z) is a positive (negative) training document; D is the training set of documents; k is the number of nearest neighbors of x in D, which the system uses to compute the score; P_k (Q_k) is the set of positive (negative) instances among the k nearest neighbors of x in D.

Binary decisions are obtained by thresholding locally on the confidence scores generated by each event-specific classifier.

Also, two modified version of kNN are used. In the first modified version, they take $kp(\leq k)$ nearest positive examples and $kn(\leq k)$ nearest negative examples from the k-neighborhood, and average the similarity scores of the two subsets respectively. The confidence score for the YES prediction on the input story is defined to be:

$$s_2(x, kp, kn, D) = \frac{1}{|U_{kp}|} \sum_{y \in U_{kp}} \cos(x, y) - \frac{1}{|V_{kn}|} \sum_{z \in V_{kn}} \cos(x, z) \tag{2.16}$$

where U_{kp} consists of the kp nearest neighbors of x among the positive documents in the training set; and V_{kn} consists of the kn nearest neighbors of x among the negative documents in the training set. By introducing the parameters kp and kn in addition to k, and by suitably choosing the parameter values, behavior of the tracking system can be adjusted effectively.

The second version is almost the same as Equation 2.15 except using the score averages instead of score sums as:

$$s_3(x,k,D) = \frac{1}{|P_k|}\sum_{y\in P_k}\cos(x,y) - \frac{1}{|Q_k|}\sum_{z\in Q_k}\cos(x,z) \tag{2.17}$$

where x is the test document; y(z) is a positive (negative) training document; D is the training set of documents; k is the number of nearest neighbors of x in D, which the system uses to compute the score; P_k (Q_k) is the set of positive (negative) instances among the k nearest neighbors of x in D. This modification prevents negative examples to dominate.

2.2.1.2 Decision Trees

In the CMU approach, decision trees are constructed by selecting the feature with maximal information gain as the root node, and dividing the training data according to the values of this feature; then for each branch finding the feature which maximizes information gain over the training instances for that branch, and so on recursively. One potential disadvantage of decision trees is that, unlike kNN, they cannot generate a continuously varying tradeoff between miss and false alarms, or recall and precision. The parameters are tuned by cross-validation [26].

2.2.1.3 Rocchio

Rocchio is a classic information retrieval method for query expansion using relevance judgements on documents. It has been applied to text categorization in a modified form:

$$c(D,\gamma) = \frac{1}{|R|}\sum_{y\in R} y + \gamma\frac{1}{|S_n|}\sum_{z\in S_n} z \tag{2.18}$$

where $c(D,\gamma)$ is the *prototype* or the *centroid* of a category and Rocchio's representation of the event. D is a training set, $R \in D$ consists of the training documents relevant to the query,γ is the weight of the component, summation $S_n \in D - R$ consists of the n most-similar (as measured by cosine similarity) negative instances to the positive centroid.

This formula allows to selectively use the negative examples that lie in the neighborhood of the positive centroid. A test document can be scored by computing the cosine similarity between the test document x and the prototype of the event $c(D, \gamma)$:

$$r(x, c(D, \gamma)) = \cos(x, c(D, \gamma)) \tag{2.19}$$

A binary decision is obtained by thresholding on this score.

2.2.2 UMass Approach

The same text representation for event detection is also applied in event tracking. A classifier is formulated automatically from the lexical features of the training set of documents using the operators from Inquery's query language. An *event* discussed in a set of relevant training documents is represented with a classifier comprising a query syntax and threshold. A separate threshold is estimated for each classifier, and documents on the stream that have similarity exceeding the threshold are classified as positive instances of an event, that is, the contents of the document are assumed to discuss the same event as the relevant document(s) with which the classifier was formulated.

Similarity scores from Equation 2.9 are needed to be normalized to produce good decision scores for tracking. They found that the most effective way was to normalize similarity scores using a standard normal transformation. The general form of the transformation is

$$decision\_score(q_i, d_j) = \frac{sim(q_i, d_j) - \mu}{\sigma} \tag{2.20}$$

where $sim(q_i, d_j)$ is the similarity between classifier q_i and document d_j. μ is the mean, and σ is the standard deviation of a distribution of similarity values using Equation 2.9.

To increase the ability of the tracking system, they implemented the adaptive version of their tracking system. The adaptive version is needed since the discussion of an event changes over time. This idea is a form of unsupervised learning. An adaptive version of the tracking system can rebuild the query after it "tracks" a news story on a given event. Experiments show that adaptive tracking works well if the number of training documents are high ($N_t \geq 4$) [3, 5, 14].

2.2.3 Dragon Approach

Dragon Systems uses statistical approaches in their tracking implementation. Event models were build from the words in the N_t training stories, after stopwords were removed with some appropriate smoothing. In order to provide a more accurate smoothing for the event model, the mixture of the background topic models that best approximates the unsmoothed event model taken as the backoff distribution. Therefore, there is a different backoff model for every event and every value of N_t.

A score for each story against its set of background models , as well as against the event model is computed, and the score difference between the best background model and the event model is reported. A threshold is applied to this difference to determine whether a story is about the event or not. This threshold can be adjusted to tune the tracker's output characteristics [3].

2.2.4 UPenn Approach

The UPenn approach uses the same weighting, comparison and future selection methods which are described in section 2.1.4 for event tacking. Results indicate that very simple feature selection with no normalization of topic scores performed best.

2.2.5 BBN Technologies' Approach

The BBN approach uses the same approach that they used in event detection (see Section 2.1.5 for details). In addition to the topic spotting metric, they calculated two additional metrics called *information retrieval* (IR) metric and *relevance feedback* (RF) metric.

The IR metric looks at the problem in exactly the positive way. Given a query Q, the probability that any new story S is relevant to the query is calculated. In this case, it is assumed that the query was generated by a model estimated from the story.

$$p(SisR|Q) \approx p(SisR) \cdot \prod_{n} p(q_n|S) \tag{2.21}$$

Again, the BBN approach uses a two-state model, where one state is a unigram distribution estimated from the story S, and the other is the unigram distribution from the hole corpus.

The RF measure is similar to the IR measure. Instead of using all of the words in the relevant stories, only those words that are common to at least two of the irrelevant stories are used.

Besides that, they also implemented causal unsupervised adaptation for their training documents, since more relevant stories usually lead to better models. Their unsupervised adaptation algorithm looks for a test story with very high score, adds it as a relevant story and re-train the system before working on the next test story.

In order to achieve optimum system performance and comparable scores, a statistical hypothesis test method is used to normalize scores.

Since different systems focus on different features of the stories, it seems reasonable to combine the probability scores from any tracking system. A linear combination of the log scores from the above three systems and the time decay method are used to increase system performance [10, 20].

2.2.6 Combining Multiple Learning Strategies

This approach hypothesize that, by combining the output scores of classifiers whose errors tend to be uncorrelated, the resulting system will have much less cross-collection and cross-event performance variance than those of the individual classifiers. This system is called as Best Overall Result Generator (BORG) [24].

BORG limits its empirical validation to Rocchio (see Section 2.2.1.3), the two variants of kNN (see Section 2.2.1.1) and BBN topic spotting language modelling (see Section 2.2.5) with a more careful analysis on the behavior of these classifiers and the conditions under which it combines them.

Decision Error Trade-off curve [11] of the classifiers is used as the primary means of analyzing the error patterns each produces. Given a validation set, following procedure shown in Figure 2.5 is used to generate a BORG system.

1. Run each classifier with different parameter settings, resulting in a set of system generated scores and a DET curve per run.
2. Select the runs whose DET curves are either globally optimal, or significantly better than other runs in a local region.
3. Combine the system output of selected runs by first normalizing the scores of each system and then compute the sum of the scores of multiple runs per test document. The normalization formula is

$$x' = \frac{x - \mu}{s.d.} \tag{2.22}$$

   where x is the original score, μ is the mean of the scores for the run, and $s.d.$ is their standard deviation. This results in a set of scores of BORG; re-normalize these scores in the same way.
4. Find the optimal threshold for BORG on the validation set.

Figure 2.5: Best Overall Result Generator (BORG) algorithm

2.2.7 Summary of Event Tracking Approaches

Summary of the event tracking approaches, are shown in Figure 2.6.

Approach	Summary
CMU	Training documents are used as positive, other documents are used as negative training documents. Three methods are used to make a decision about the given document. • kNN classifier • Decision trees induction • Rocchio
UMass	Training documents are represented by a query and a threshold. Similarity to the given document is found by running the query on it. If the similarity exceeds the threshold, we conclude that the given document is about given event.
Dragon	Event models are produced by using training documents. Distance between event models and given model is calculated with a statistical method. A threshold is applied to obtain the binary decision.
UPenn	UPenn uses the similarity values, found in detection process. Binary decision is obtained by thresholding these values.
BBN	BBN uses the same probabilistic method. Two additional metrics: IR metric and RF metric are also calculated. Binary decision is obtained by combining these three metrics.
BORG	Combines the output scores of four classifiers: • Rocchio • Two variants of kNN classifier • BBN topic spotting model

Figure 2.6: Summary of Event Tracking Approaches

Chapter 3

On-line New Event Detection and Tracking

A solution to on-line new event detection and tracking problem is presented in this chapter. On-line new event detection phase of the implementation is discussed first, since, the novelty decision about a new document is done first. After the event detection section, our solution to event tracking problem is given. The complete algorithm is given in the last section.

3.1 On-line New Event Detection

Our approach for on-line new event detection is based on the difference between an event and a news. Since every news item does not necessarily give information about an event, we need not to be informed about all the events found by the system. As a result, our detection system suggests that new event detections should be alarmed after a user-defined amount of supporting documents found.

3.1.1 Preprocessing

Preprocessing is the first step in our implementation as in any other text categorization implementations. In this step, we transform the documents, which are typically strings of characters, into a representation convenient for detection algorithm. This process is done by deeper investigation of the documents. Frequent words that carry no information, called **stopwords**, (i.e. pronouns, prepositions, conjunctions etc.) are needed to be removed. Also removing suffixes from the word roots, called **word stemming**, are needed to increase the system performance [1]. As a result, preprocessing phase includes:

- Removing tags (i.e., time, resource info, etc.).
- Removing stopwords.
- Performing word stemming.
- Performing simple corrections.

In our implementation, deep lexical analysis of the words was not done, instead, a simple list of words that contains the word stems was prepared, and the words that start with any of the word in this list by its word stem were replaced. Also, the corpus that we study has a lot of incorrectly written words. In order to reduce the effect of these words, a one-to-one matching list of important names is used to correct them.

3.1.2 Document Representation

We implemented a commonly used document representation technique so called vector space model. In the vector space model, documents (stories) are represented using a vector of weighted terms. There are several ways of determining the weight of a word in a document, but most of the approaches are based on two empirical observations regarding text:

- The more times a word occurs in a document, the more relevant it is to the topic of the document.
- The more times the word occurs throughout all documents in the collection, the more poorly it discriminates between documents.

According to the criteria listed above, terms in a document vector are statistically weighted using the term frequency (TF) and the Inverse Document Frequency (IDF). As a weighting schema, we selected *tfc (term frequency component) weighting*, which take into account both the frequency of the word in a document and the frequency of the word throughout all documents in the collection. Furthermore, it takes into account that documents may be of different lengths. Tfc weights are calculated using the Equation below:

$$w_{ik} = \frac{f_{ik} * log(\frac{N}{n_i})}{\sqrt{\sum_{j=1}^{M}[f_{jk} * log(\frac{N}{n_j})]^2}} \tag{3.1}$$

where w_{ik} is the weight of word i in document k and f_{ik} is the frequency of the word i in document k, N is the number of documents in the collection, n_i is the total number of times word i that occurs in the hole collection, and M corresponds to the number of words in the auxiliary corpus. If the word does not appear in the auxiliary corpus ($n_i = 0$ or $n_j = 0$), a default value 1 is used instead of these values.

In order to use static IDF values, we used the incremental IDF. Since new stories arrive continuously, the new vocabulary from incoming documents should be inserted into the IDF calculations [28]. The incremental version of the IDF is defined to be:

$$IDF_{(t,p)} = log(\frac{N_p}{n_{(t,p)}}) \tag{3.2}$$

where p is the current time, t is a word, N_p is the number of accumulated documents up to the current point in time, and $n_{(t,p)}$ is the number of documents that contains the word t up to the current point.

As a result, the word weighting formula becomes:

$$w_{ik} = \frac{f_{ik} * log(\frac{N_p}{n_{(t,i)}})}{\sqrt{\sum_{j=1}^{M}[f_{jk} * log(\frac{N_p}{n_{(t,j)}})]^2}} \tag{3.3}$$

3.1.3 Document Comparison

We did not grow any cluster, to represent events, in the implementation. Keeping individual documents without clustering makes the *novelty test* more difficult for the current story to pass, because this story must be sufficiently different from *all* of the past stories, a stronger condition compared to being different from an *average* of past stories [26]. As a result, the current story with all the existing stories in the time window is computed.

A similarity value is calculated while comparing two documents. The similarity of two documents is defined as the cosine value of the corresponding document vectors:

$$sim(k,q) = \frac{\sum_{j=1}^{M} w_{jk} * w_{jq}}{\sqrt{(\sum_{j=1}^{M} w_{jk}^2) * (\sum_{j=1}^{M} w_{jq}^2)}} \tag{3.4}$$

where $sim(k,q)$ is the cosine similarity value between documents k and q, w_{jk} is the weight of word j in document k, w_{jq} is the weight of word j in document q, $\sum_{j=1}^{M} w_{jk}^2$ is the sum of squares of the word weights in document k, and $\sum_{j=1}^{M} w_{jq}^2$ is the sum of squares of the word weights in document q.

Since we are using tfc weighting schema (Equation 3.3), the sum of squares of the word weights for a document is equal to 1, and the denominator of the similarity equation becomes 1. So, we can simplify Equation 3.4 as:

$$sim(k,q) = \sum_{j=1}^{M} w_{jk} * w_{jq} \tag{3.5}$$

3.1.4 Time Window

We also added the *time penalty* functionality to document similarity calculations [3, 26, 28]. We used *day-based* windowing with an exponential smoothing. Given the current document k in day $date_k$ in the input stream, we impose a *time window* of m days prior to day d, and define the modified similarity between k and any document q in the time window in day $date_q$:

$$sim(k,q)' = \begin{cases} sim(k,q) & \text{if } (date_k - date_q) < 1 \\ (date_k - date_q)^{-\alpha} * sim(k,q) & \text{if } (date_k - date_q) \geq 1 \end{cases}$$

If a document's date does not fall into the time window, the similarity between that document and the current document is not computed. That is $sim(k,q)'$ is assumed to be 0.

In order to apply the time penalty function, a two-phased method is used. In the first phase, we keep the k most similar documents in the time window without applying the smoothing function. This method allows us to find the k most similar documents in the time window. In the second phase, we apply the smoothing function to them. This process prevents the domination of the recent documents even if they are not similar to the older ones and increases the importance of the recent ones within the neighbors.

3.1.5 Novelty Decision

An additional threshold called *novelty threshold* (t_n) was used [26]. If the maximal similarity score between the current document and any document in the past are below the novelty threshold, then this document is labelled as "NEW", meaning that it is the first story of a new event; otherwise a flag of "OLD" is issued. By tuning the novelty threshold, one can adjust the sensitivity to novelty in on-line detection. We formulate this approach with a decision score:

$$decision(d_i) = \arg\max_c \{sim(i,c)'\} - threshold \tag{3.6}$$

where d_i is the current document, $\arg\max_c\{sim(i,c)'\}$ is the maximum similarity value between document i and any document in the time window, and *threshold* is the user-defined novelty threshold.

If the decision score is *positive*, then we assume that the document does not discuss a new event. On the contrary, negative decision score is the indication of the new event.

In addition to the binary (New or Old) prediction, a confidence score is also computed for each incoming document, indicating how *new* this document is as measured by the system [26, 28]. This score is defined to be:

$$score(i) = 1 - \arg\max_{c}\{sim(i,c)'\} \tag{3.7}$$

where i is the current document and $\max\{sim(i,c)'\}$ is the maximum similarity value between document i and any cluster in the time window.

3.1.6 Support Threshold

One of the main drawback of the previous on-line new event detection implementations is that they focus on the first story detection and did not focus on the number of alarms. We observed from the experiments that the number of *new event* alarms are so high that user of the system must look at the new event alarms many times in a day.

In order to decrease the number of new event alarms, we added a new user-defined threshold called *support threshold* (t_s). By the help of this threshold, the number of new event alarms can be decremented enormously.

Our approach is originated from the difference of the event and news (see Section 1.1 and 1.2 for details). As it is described in related section, news can be informative. We have observed many news stories that are talking about local events, local activities, results of the sportive activities etc. These type of one-time-only informative news stories should be eliminated and hidden from the user.

Our hypothesis is simple:

> If a new event is worth for alarming, it should be supported by up-coming news in a short time.

Since we assume a multi-resource environment, supporting news stories come rapidly. Because, all the news agencies rapidly react to new and important events. The more important the event is, the faster the reaction of the agencies will be.

Also, the user can define the support threshold easily. It is defined as the number of news stories needed from each resource before an alarm is issued. First, it is expressed by words. Then it is formulated by defining a weight for each resource, a test condition and an overall threshold. The resulting representation of the support threshold is also easy and understandable. User can change this definition easily.

Support threshold is checked when a new document is inserted into an event. An example of a support threshold can be defined as: "wait one supporting news story from any news resource including itself" This definition can be formulated as:

$$src_1 + src_2 + src_3 + src_4 > 1$$

where src_i is the number of documents, in a given event, which belongs to resource number i. We call the left-hand-side of the equation as *support value*. First, support value of an event is calculated. Then, the threshold is checked according to the condition. If the solution of the equation is *true*, then new event alarm for the event is issued. If the solution of the equation is *false*, nothing is done.

As a result, we make *new event decision* immediately, however we delay *new event alarm* until a support threshold is exceeded. This approach decreases the number of new event alarms dramatically, even if the support threshold is kept very small such as "wait one supporting news from any news resource including itself".

3.2 Event Tracking

We implemented event tracking as an unsupervised learning task. The tracking process starts immediately after detection of a new event. Therefore, our tracking system is trained with only one document ($N_t = 1$), which is the news story of the first document of an event. It is clear that as the number of training documents increases, the system performance also increases. Furthermore, events evolve over time. So it is always natural to add more training documents to a given event. This process is called as *unsupervised adaptation* [10], and we applied it by adding each tracked event to the training documents. We also propose that a news story can talk about more than one event, so it can belong to more then one event. In this chapter, we give our solutions to the event tracking problem according to considerations given above.

Our event tracking approach uses the similarity scores obtained from the new event detection calculations by keeping the k most similar documents and their similarity scores to the given document. If the decision for the given story in event detection phase is “old”, these k similarity information is used in the k-Nearest Neighbor (kNN) classification method to find the related events of the given document.

3.2.1 K-Nearest Neighbor Classification

Since it is found as a robust approach to text categorization, ranking among the top-performing classifiers in cross-method evaluations on benchmark collections, the kNN classification method is used for event tracking [23, 25, 27].

The kNN algorithm is quite simple [27]: given a test document, the system finds the k nearest neighbors among the training documents, and uses the categories of the k neighbors to weight the category candidates. The similarity score of each neighbor document to the test document is used as the weight of the categories of the neighbor document. If several of the k nearest neighbors share a category, then the per-neighbor weights of that category are added together, and the resulting weighted sum is used as the likelihood score of that category with respect to the test document. By sorting the scores of candidate categories, a ranked list is obtained for the test document. By thresholding on these scores, binary category assignments are obtained.

Yang et al. adapted conventional M-way classification kNN to the 2-way kNN classification problem of event tracking by constructing a 2-way kNN for each event (see Section 2.2.1.1 on Page 20 for details).

The performance of kNN depends on the choice of the value of k. Optimizing the value of k in event tracking is a problem, due to the very small number of positive examples in the training set. Using a large value of k will retrieve many negative examples whose sum could easily exceed the sum of the positive examples, even if each negative example is very dissimilar to the test document and thus has a small similarity score. Using a small value of k will cause the system to retrieve only negative examples for a test document unless it is very close to a positive training example. To solve this problem, two alternatives of Formula 2.15 were used to calculate the confidence score for kNN algorithm. One of them is the Formula 2.16, and the other is the one which we used in our tracking implementation as:

$$s(x,k,D) = \frac{1}{|P_k|}\sum_{y\in P_k}\cos(x,y) - \frac{1}{|Q_k|}\sum_{z\in Q_k}\cos(x,z) \tag{3.8}$$

where x is the test document; y(z) is a positive (negative) training document; D is the training set of documents; k is the number of nearest neighbors of x in D, which the system use to compute the score; P_k (Q_k) is the set of positive (negative) instances among the k nearest neighbors of x in D.

Empirical results show that both of the new variants of kNN can significantly improve the tracking performance of the original kNN [25].

3.2.2 Event Tracking Method

Event tracking starts with k nearest neighbors of a given document which are found in detection phase. Next, the events that at least one neighbor exists in their event lists are found. Than, we calculate the Equation 3.8 for each event. While we are calculating the kNN scores for a given event, the documents which are in the event list of a given event are used as the positive training documents, and the others are used as negative training documents. Decision is made by thresholding the results of Equation 3.8:

$$decision(x) = s(x, k, D) - threshold \tag{3.9}$$

We add the document information to the events which has a positive decision scores.

3.3 On-line New Event Detection and Tracking Algorithm

With the considerations described in this chapter, our on-line new event detection and tracking algorithm processes each new document sequentially, as shown in Figure 3.1.

```
1. Prepare a vector space model of the document (Equation 3.3).
2. Remove the old documents that exceed the time window according to the
   date of new document.
3. Calculate the similarities (Equation 3.5) between the new document and
   existing documents in the time window. Keep $k$ maximum similarity values
   and information about them.
4. Apply time penalty function to $k$ maximum similarity values.
5. Calculate the decision score (Equation 3.6) for the new document.
6. If the decision score for the current document does not result in a positive
   value, flag the document as containing a new event. Calculate support value
   for the new event (Section 3.1.6).
7. If the decision score for the current document results a positive value, flag
   the document as not containing a new event. Then:
   (a) Find all events that contain at least one of the neighbors.
   (b) For each event, found in previous step:
         i. Calculate the kNN score (Equation 3.8).
        ii. Calculate the kNN decision score(Equation 3.9).
       iii. If the decision score does not result a positive score, ignore it.
        iv. If the decision score results a positive value, add the document
            to the event list and recalculate support values for those events
            (Section 3.1.6).
8. If the support value of any event exceeds the support threshold, perform
   new event alarm process.
9. Adjust document counts to calculate the idf value of the next document.
10. Add the new document to the time window.
```

Figure 3.1: On-line new event detection and tracking algorithm.

Chapter 4

Experimental Design and Results

Effectiveness measures of our system were evaluated using the stories related to the 15 selected events (about 2.8% of the entire corpus). Although, the detection and tracking system were run on the entire corpus. The results were obtained on a personal computer with Intel Celeron MMX[1] - 400 Mhz. CPU and 64 MB of main memory. The process time of a document, including the preprocessing phase, changes between 1 to 15 seconds with an average of 3 seconds. Length of the document, window size, and kNN threshold effects the processing time. Window size has the maximum effect among them.

4.1 Evaluation Methodology

In this section, we discuss the effectiveness measures that were used to evaluate our experiments.

[1]Celeron is a registered trademark of Intel Corporation.

4.1.1 Effectiveness Measures

To evaluate the effectiveness of the on-line detection results, a two-by-two contingency table is used for each event, as shown in Table 4.1, where a, b, c and d are document counts in the corresponding cases.

Table 4.1: Contingency table for on-line detection

	NEW is true	OLD is true
System-predicted NEW	a	b
System-predicted OLD	c	d

Since there are only 15 events defined, and each event has only one first story, the total number of true *New* stories is 15 for the entire corpus. This is a small number for a statistically reliable estimation of performance. To improve the reliability, an 11-pass evaluation was conducted [3, 26]. The first pass used the entire corpus; the second pass used the modified corpus after moving ("skipping") the first story of each event; the third pass used the modified corpus after moving the first two stories of each event, and so on. The eleven passes are labelled as $N_{skip} = 0, 1, \ldots, 10$. A contingency table was computed for each value of N_{skip}; a global contingency table then was obtained by summing the corresponding cells in the per-N_{skip} contingency table.

Tracking effectiveness is calculated by using the contingency table, which is similar to Table 4.1.

Table 4.2: Contingency table for tracking

	YES is true	NO is true
System-predicted YES	a	b
System-predicted NO	c	d

4.1.2 Performance Measures

Six performance measures of effectiveness are also defined by using the contingency table values. These measures are given in Table 4.3.

Table 4.3: Performance measures for on-line detection and tracking.

Name	Formula	Condition
Miss	$m = \frac{c}{a+c}$	if $a + c > 0$, otherwise undefined
False Alarm	$f = \frac{b}{b+d}$	if $b + d > 0$, otherwise undefined
Recall	$r = \frac{a}{a+c}$	if $a + c > 0$, otherwise undefined
Precision	$p = \frac{a}{a+b}$	if $a + b > 0$, otherwise undefined
F_1	$F_1 = \frac{2rp}{(r+p)} = \frac{2a}{(2a+b+c)}$	if $(2a + b + c) > 0$, otherwise undefined
Cost	$C = \alpha_1 \frac{b}{n} + \alpha_2 \frac{c}{n}$	where $n = a + b + c + d$

F_1 score equally weights the recall and precision values, which is a desired condition [17]. Cost of detection C_{det} and cost of tracking C_{trk} can be calculated by selecting different values for the parameters α_1 and α_2. In our implementation, we selected $\alpha_1 = 0.2$ and $\alpha_2 = 0.98$ for detection task cost calculations [22], and $\alpha_1 = 0.1$ and $\alpha_2 = 1.0$ for tracking task cost calculations [25].

Global performance over all events is evaluated using two methods: the *micro average*, obtained by first summing the corresponding cells in the contingency tables of the individual events and computing the global performance scores from the combined table, and the *macro average*, obtained by computing per-event performance first and averaging them. The micro-average introduces a scoring bias towards frequently-reported events, and the macro-average towards less-reported events; both are informative [25].

4.2 Experimental Results

In order to test our implementation, we run the algorithm in 4 different window sizes: 15, 10, 7 and 4 days, including the current date of the document which is tested. The previous works [26, 28] suggest that the window sizes between 1 to 6 weeks provide good performance. But we have also seen that the data sizes are too small in previous implementations. For example, Yang et al. [28] suggest a window size of 2500 documents which cover about 6 weeks, but we have reached that average document count in 4 days of window size, because the average number of documents per day is about 600 in weekdays, and about 300

Table 4.4: Average number of documents in experiment windows.

Window Size	**Average Number of Documents**
4 days	2089
7 days	3628
10 days	5165
15 days	7701

on weekends (see Figure A.2). The average document counts, for each window sizes, are given in Table 4.4. As it is seen from Table 4.4, the document counts for window sizes of 7, 10, and 15 days are relatively higher than the previous works.

4.2.1 Time Windowing Parameter Selection

For the time penalty function parameter α, in Section 3.1.4, we selected $\alpha = 0.25$. This value is selected to keep the penalty function value greater than 0.5 for a considerable time period.

Events are typically reported within 4-week time window (see Section 1.1), so we selected the half of this period (14 days) as a suitable date difference for decreasing the similarity of a document by 0.5. As it can be seen from Figure 4.1, the function value becomes 0.5 when the time difference is equal to 16 days.

4.2.2 11-pass Test Results

We calculated the performance measures for each window size, in order to find the optimum threshold for new event detection process. The optimum detection threshold is selected according to the following considerations:

- Around the crossing point of recall and precision values. That is, we tried to keep the recall and precision values equal. When a choice must be made between them, the former is generally preferable [17].
- Which maximizes the F_1 score [26, 28].

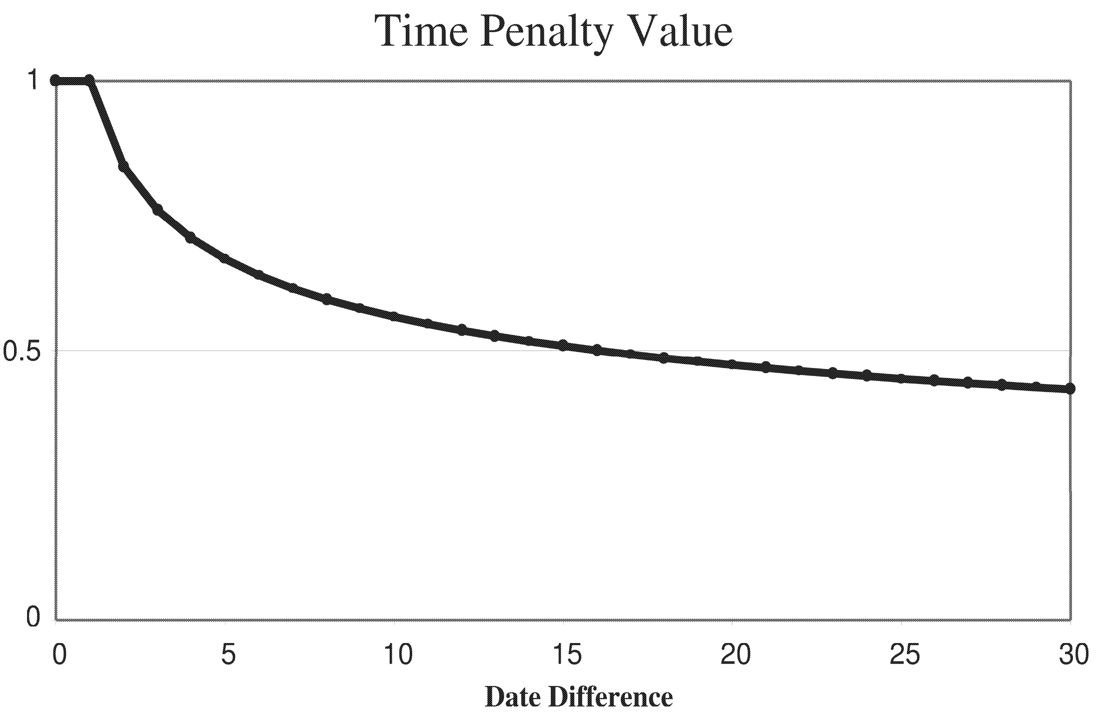

Figure 4.1: Value of time penalty function for $\alpha = 0.25$.

- Which minimizes both miss and false alarm rates [4].
- Which minimizes the C_{det}.

Since F_1 score equally weights the recall and precision values, maximum value of the F_1 score must be around the crossing point of recall and precision values. Likewise, miss and false alarm rates directly effect the cost function. Minimizing the cost minimizes the miss and false alarm rates. As a result, optimum threshold is selected by plotting the F_1 and C_{det} scores to a chart, and then by selecting the optimum point accordingly. An example of such chart is given for micro-averaged values of 15 day window size in Figure 4.2. Since at that point F_1 score has a local maximum, and C_{det} value is very close to the minimum, the optimum threshold was selected as 0.1493.

Optimum thresholds for all window sizes are selected by using the same methodology. Results are given in Table 4.5. According to the results obtained from the optimum performance measures, optimum thresholds are selected as

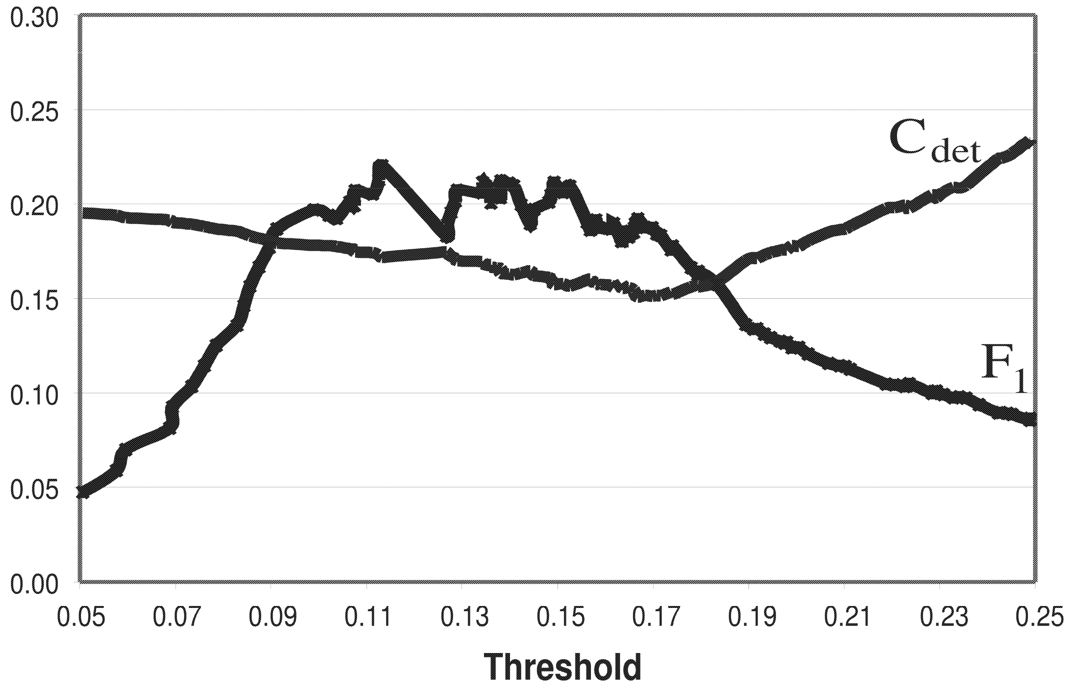

Figure 4.2: On-line detection F_1 and C_{det} curves for micro-averaged values of 15 day window size.

Table 4.5: Optimum threshold values(m,f,r,p values are given in percentages).

window size	type	threshold	m	f	r	p	F_1	C_{det}
15 days	micro	0.1493	69.1%	1.96%	30.9%	16.1%	0.212	0.157
15 days	macro	0.1493	69.1%	1.97%	30.9%	15.8%	0.209	0.157
10 days	micro	0.1381	72.1%	1.42%	27.9%	19.3%	0.228	0.158
10 days	macro	0.1381	72.1%	1.42%	27.9%	19.1%	0.226	0.158
7 days	micro	0.1398	71.5%	1.58%	28.5%	18.0%	0.221	0.159
7 days	macro	0.1398	71.5%	1.58%	28.5%	17.7%	0.218	0.159
4 days	micro	0.1381	69.7%	2.15%	30.3%	14.7%	0.198	0.160
4 days	macro	0.1381	69.7%	2.15%	30.3%	14.5%	0.196	0.160

0.15 for 15 day window size and 0.14 for the other window sizes. We tested our on-line new event detection and tracking implementation by using these threshold values.

We can observe from Table 4.5 that macro-averaged and micro-averaged values are almost the same. This result is normal, since we used only the documents of events as an entire corpus in performance calculations. While we were calculating the macro-average measures, the summation of a,b,c, and d values are equal to the number of documents of a given event. As a result, macro-averaged values are calculated independently for each event. Since there is no relation among the event calculations, the micro-averaged and micro-averaged values should be very close to each other.

4.2.3 Event Tracking Results

In order to test the effectiveness of the tracking system, we run the tracking process with optimal detection threshold and 10 different decision threshold values of the Equation 3.9. This process helps us to understand the effect of the kNN decision thresholds.

The selected kNN decision threshold values and the macro averaged tracking performance results of 15 day window size, according to the optimum detection threshold 0.15 is given in Table 4.6. In that table, kNN threshold means kNN decision threshold value which is shown in Equation 3.9. Macro averaged C_{trk} is

Table 4.6: Macro averaged tracking results for 0.15 detection threshold with 15 days window size(m,f,r,p values are given in percentages).

kNN threshold	m	f	r	p	C_{trk}	F_1
0.200	78.1%	0.0007%	21.9%	94.4%	0.00172	0.32
0.100	70.5%	0.0170%	29.5%	73.1%	0.00165	0.34
0.090	66.5%	0.0126%	33.5%	74.9%	0.00158	0.38
0.080	67.6%	0.0203%	32.4%	68.2%	0.00155	0.37
0.075	66.4%	0.0287%	33.6%	64.2%	0.00153	0.36
0.070	67.3%	0.0401%	32.7%	57.2%	0.00153	0.33
0.065	67.0%	0.0474%	33.0%	51.0%	0.00152	0.32
0.060	64.9%	0.1687%	35.1%	35.1%	0.00162	0.29
0.055	60.3%	0.2644%	39.7%	33.1%	0.00169	0.28
0.050	57.0%	0.5658%	43.0%	29.2%	0.00192	0.28

Table 4.7: Optimum decision thresholds for tracking system.

window size	kNN threshold
15 days	0.065
10 days	0.065
7 days	0.070
4 days	0.055

used as primary measure [25], and 0.065 is selected as an optimum kNN threshold.

By applying the same process for the other window sizes, optimum decision thresholds are selected. The optimum decision thresholds are shown in Table 4.7.

Since performance measures are calculated by using contingency table, the comparison between our tracking method and simple kNN method (described is Section 2.2.1.1) is done by using the results found in the contingency tables of each method. The comparison values for 15 day window size is given in Table 4.8. kNN_1 means the previous method, while kNN_n means our proposed method.

As it can be seen from Table 4.8, we have achieved better a and c values, and worse b value. In fact, these results can be expected. Because our tracking approach has the ability to relate a document to more than one event. This ability decreases the amount of c value, but can increase the amount of b value.

Table 4.8: Comparison of the contingency table results.

kNN threshold	kNN_n a	kNN_1 a	kNN_n b	kNN_1 b	kNN_n c	kNN_1 c	kNN_n d	kNN_1 d
0.200	119	116	5	1	1203	1206	45203	45207
0.100	180	179	118	39	1142	1143	45090	45169
0.090	228	208	88	51	1094	1114	45120	45157
0.080	257	235	141	74	1065	1087	45067	45134
0.075	271	250	200	107	1051	1072	45008	45101
0.070	284	235	279	129	1038	1087	44929	45079
0.065	297	247	330	125	1025	1075	44878	45083
0.060	309	245	1172	142	1013	1077	44036	45066
0.055	325	249	1837	191	997	1073	43371	45017
0.050	374	261	3935	225	948	1061	41273	44983

It should not be forgotten that while we were calculating the primary measure C_{trc}, b is multiplied by α_1 which is taken as 0.1, and c is multiplied by α_2 which is taken as 1.0. This means that the effect of c on the tracking cost is 10 times greater than the effect of b.

4.2.4 The Effect of Support Threshold

Our main contribution to on-line new event detection problem is adding one additional threshold, called *support threshold* by using the resource information of the news stories. Our aim is to reduce the number of new event alarms according to the user needs. In order to check the effect of the support threshold, we selected 13 different functions as shown in Table 4.9. In this table, Src_i means the amount of documents, from resource number i, Src_a means the amount of documents from the resource of the first document of a given event. First ten methods can be called as simple methods, since alarm is issued after a determined amount of supporting documents are found, regardless of their resource. Method 11 looks for a supporting document which is different from the resource of the first document of a given event. All of the resources must support the event in method 12. Method 13 is the most important one. It shows the priority or the confidence of the user. As it can be seen from the definition in Table 4.9, some weighting values are given to the resources in method 13. If a document

Table 4.9: Support threshold evaluation methods.

Method #	**Alarm if the result is true**
1-10	$Src_1 + Src_2 + Src_3 + Src_4 > 1$
2	$Src_1 + Src_2 + Src_3 + Src_4 > 2$
3	$Src_1 + Src_2 + Src_3 + Src_4 > 3$
4	$Src_1 + Src_2 + Src_3 + Src_4 > 4$
5	$Src_1 + Src_2 + Src_3 + Src_4 > 5$
6	$Src_1 + Src_2 + Src_3 + Src_4 > 6$
7	$Src_1 + Src_2 + Src_3 + Src_4 > 7$
8	$Src_1 + Src_2 + Src_3 + Src_4 > 8$
9	$Src_1 + Src_2 + Src_3 + Src_4 > 9$
10	$Src_1 + Src_2 + Src_3 + Src_4 > 10$
11	$Src_1 + Src_2 + Src_3 + Src_4 - Src_a > 1$
12	$((Src_1 > 0) \wedge (Src_2 > 0) \wedge (Src_3 > 0) \wedge (Src_4 > 0)) = true$
13	$Src_1 + (2 * Src_2) + (8 * Src_3) + (12 * Src_4) > 11$

Table 4.10: Number of new event alarms for the methods 1-10.

Window Size	total#	1	2	3	4	5
15 days	11343	5975	4423	3617	3134	2760
10 days	11978	6297	4628	3739	3210	2785
7 days	13789	6893	4902	3897	3297	2819
4 days	11691	6028	4403	3652	3161	2840

Window Size	total#	6	7	8	9	10
15 days	11343	2482	2253	2048	1883	1735
10 days	11978	2495	2271	2056	1890	1731
7 days	13789	2507	2235	1997	1809	1655
4 days	11691	2568	2362	2203	2074	1956

belongs to the resource 4, then new event alarm is issued immediately, otherwise the system waits until the summation exceeds the given amount.

The results of the evaluation of the first 10 methods, for each window size at their optimum thresholds, are given in Table 4.10, where total# means the number of total new event alarms issued without using the support threshold, and the numbers at the top of each column indicate the method number. We can easily conclude that, even with a simple usage of support threshold, the number of new event alarms can be decreased dramatically.

The results of the evaluation of more complex but also more useful support

Table 4.11: The analysis of the number of new event alarms for the methods 11-13.

	window size	15 days	10 days	7 days	4 days
Method #	total#	11343	11978	13789	11691
11	NoA	3097	3265	3420	3380
	min	2	2	2	2
	max	177	164	105	263
	avg	5.7	5.4	4.6	7.2
12	NoA	224	199	141	225
	min	4	4	4	4
	max	290	298	107	626
	avg	33.7	27.6	16.1	57.9
13	NoA	3291	3394	3525	3408
	min	1	1	1	1
	max	8	8	8	8
	avg	5.1	5.0	4.9	5.1

threshold methods are investigated with a deeper analysis, so the result of the analysis is given in a different table (Table 4.11), where *NoA* means the number of alarms, *min* is the minimum, *max* is the maximum and *avg* is the average number of documents tracked by the system before an alarm issued. As it can be derived from the table, the number of documents needed to issue an alarm, are different for each event. For example, in method 13, an alarm can be issued immediately when a new event detected from a desired news resource Src_4. The maximum values of methods 11 and 12 shows that waiting for a support from more than one resource with a uniform weighting can delay the alarm time very much, but as it can be derived from the maximum value of method 13, nonuniform weighting works better.

Chapter 5

Conclusion and Future Work

This thesis presents new approaches to the on-line new event detection and tracking problem.

Different than the previous studies, we assume a multi resource environment. By using resource information of news stories, obtained from this environment, we define a new concept, called support threshold, as the number of news stories needed from each resource before an alarm is issued. Support threshold is used to decrease the number of new event alarms and it can be adjusted by the user. Furthermore, user can define the priority or the weights of the news resources by using the support threshold. Experimental results showed that the support threshold significantly decreases the number of new event alarms.

Another new approach that we implemented is relating a news story to more than one event. This approach is also promising some improvements, but it needs further investigation. As a future work, unsupervised adaptation part of the process can be implemented by using another threshold, which is greater than the detection threshold, to increase the system performance and decrease the effect of the incorrectly tracked events.

Unfortunately, the relationship between events and topics is not so simple and well defined in real life. Most of the time events consist of sub-events and topics consist of broader topics. Moreover one event can cause to a new event. This close and usually uncertain relation causes difficulties in implementation and increases the false alarm and/or miss rate of the system. But level of abstraction for topics and events can be defined and adjusted by the user with the adjustments of the thresholds.

From a statistical learning point-of-view, it is more difficult to identify documents as instances of small, *living* events than of large, *stable* topics. The small size of an event also presents special difficulties in tuning the parameters of the tracking system to produce optimal results [25]. Consequently, an effective detection and tracking system can be implemented by detecting the topic of the event and than testing for the novelty of the document. This process can easily incorporated to our implementation.

Bibliography

[1] K. Aas and L. Eikvil. Text categorisation: A survey. Technical report, Norwegian Computing Center, June 1999.

[2] J. Allan, L. Ballesteros, J. Callan, and W. Croft. Recent experiments with inquery. In *Proceedings of TREC-4*, pages 49–63, 1995.

[3] 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*, 1998.

[4] J. Allan, V. Lavrenco, and H. Jin. First story detection in TDT is hard. In *Proceedings of the ninth international conference on Information knowledge management CIKM 2000*, pages 374 – 381, 2000.

[5] J. Allan, V. Lavrenko, and R. Papka. Event tracking. Technical Report CIIR Technical Report IR-128, UMASS Computer Science Department, 1998.

[6] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45, 1998.

[7] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *Proceedings of the 3rd Text Retrieval Conference.* NIST, 1994.

[8] J. Callan. Document filtering with inference networks. In *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, 1996.

[9] J. Callan, W. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing & Management*, 31(3):327–343, 1994.

[10] H. Jin, R. Schwartz, S. Sista, and F. Walls. Topic tracking for radio, TV broadcast, and newswire. In *Proceedings of the DARPA Broadcast News Workshop*, Feb 28-Mar 3 1999.

[11] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proc. Eurospeech '97*, pages 1895–1898, Rhodes, Greece, 1997.

[12] M. Mayer. *Making News.* Harvard Business School Press, Boston, Massachusetts, 1993.

[13] NIST. Topic detection and tracking research group web site. In `http://www.nist.gov/speech/tests/tdt/index.htm`. National Institute of Standards and Technology, August 31, 2001.

[14] R. Papka. *On-line New Event Detection, Clustering, and Tracking.* PhD thesis, University of Massachusetts Amherst, September 1999.

[15] R. Papka and J. Allan. On-line new event detection using single pass clustering. Technical Report UMASS Computer Science Technical Report 1998-02, Intelligent Information Retrieval Department of Computer Science University of Massachusetts Amherst, MA 01003, 1998.

[16] D. Randall. *The Universal Journalist.* Pluto Press, second edition, 2000.

[17] G. Salton. *Automatic Text Processing.* Addison-Wesley, Pennsylvania, 1989.

[18] J. Schultz and M. Liberman. Topic detection and tracking using idf-weighted cosine coefficient. In *Proceedings of the DARPA Broadcast News Workshop*, pages 189–192, 1999.

[19] M. Stephens. *A History of News.* Harcourt Brace College Publishers, 1997.

[20] F. Walls, H. Jin, S. Sista, and R. Schwartz. Topic detection in broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, Feb 28-Mar 3 1999.

[21] C. L. Wayne. Multilingual topic detection and tracking: Successful research enabled by corpora and evaluation. *Language Resources and Evaluation Conference (LREC)*, pages 1487–1494, 2000.

[22] C. L. Wayne. Topic detection and tracking in English and Chinese. In *Proceedings of the 5th International Workshop Information Retrieval with Asian Languages*, 2000.

[23] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

[24] Y. Yang, T. Ault, and T. Pierce. Combining multiple learning strategies for effective cross validation. In *Proc. 17th International Conf. on Machine Learning*, pages 1167–1174. Morgan Kaufmann, San Francisco, CA, 2000.

[25] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving text categorization methods for event tracking. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–72, Athens, GR, 2000. ACM Press, New York, US.

[26] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[27] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

[28] Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 28–36, 1998.

Appendix A

Corpus Information

In order to test our approach, we created a corpus which spans the period from January 2, 2001 to March 31, 2001 with 46,530 Turkish news stories from the Reuters newswire. Corpus consists of four independent news resources of Reuters, which are :

1. Anadolu News Agency

2. Reuters News

3. Dünya News Agency

4. İstanbul Stock Exchange Company News

Anadolu News Agency news stories contain political, sports, economical, cultural, local and other types of events. Reuters News usually contain economical news stories. Dünya News Agency news stories usually contain economical and political events, and long comments about economical and political events. İstanbul Stock Exchange Company News contain only stock market news stories which are usually very short.

The majority of the corpus documents (69%) belongs to the Anadolu News Agency and the minority of the documents (2%) are come from İstanbul Stock

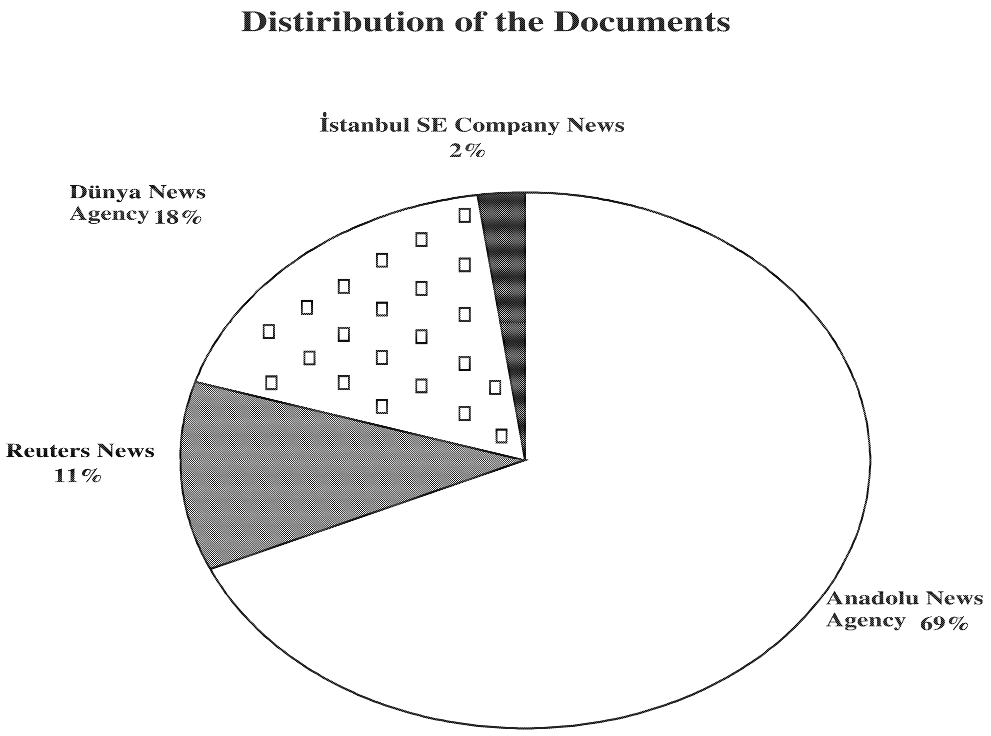

Figure A.1: Distribution of documents in the corpus.

Exchange Company News. The distribution of the documents in the corpus is shown in Figure A.1.

We can expect to have more news stories in weekdays with respect to the weekends. Figure A.2 shows that the average number of weekday documents are almost twice the size of weekend documents. The overall average number of weekday documents is 611 while the average number of weekend documents is 300.

From the 46,530 news, we selected 15 events which form 1322 (2.8%) stories to test our approach. The distribution of events across the corpus is shown in Figure A.3. The points in the figure means that there exists at least one news story in that day.

The events used for evaluation are listed with the number of event documents about the event in Table A.1.

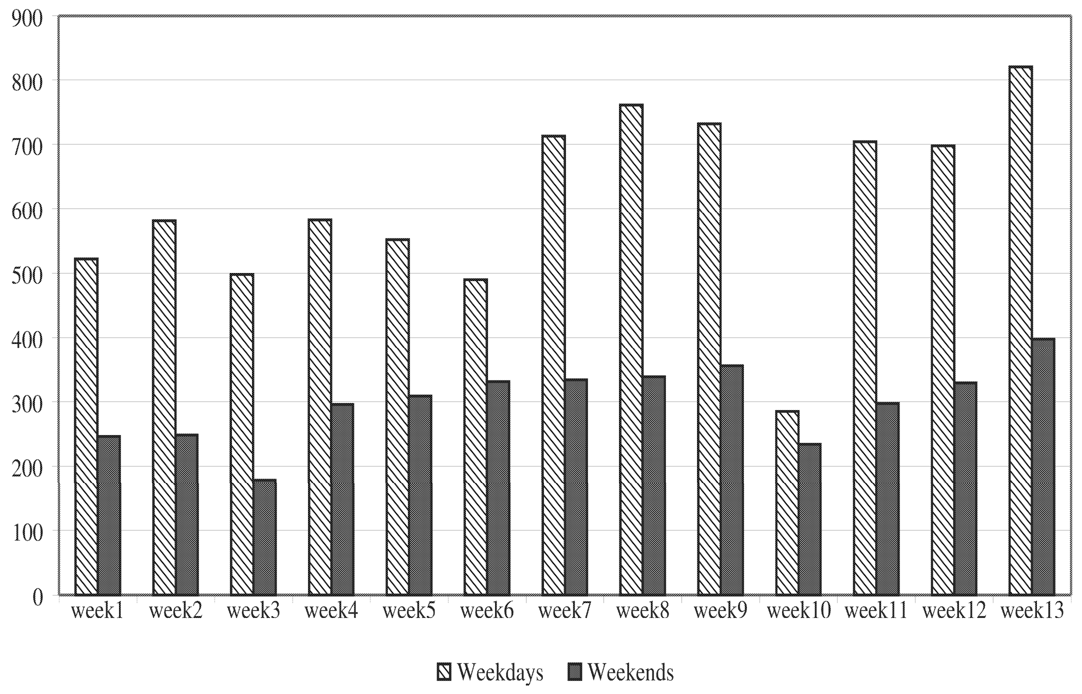

Figure A.2: Average number of documents per day, according to the weeks of the corpus.

Table A.1: Selected events from the corpus.

No	Event	# of Docs.
1	Ship sink in Kemer	40
2	Şişli police station bombing	33
3	Launching of Türksat 2A	31
4	Operation “White Energy”	174
5	Prof. Zekeriya Beyaz assault	65
6	Operation “First Curtain”	23
7	Armenian Assimilation Law in France	358
8	Gaffar Okkan assassination	222
9	Death of the representative Fevzi Şıhanlıoğlu	98
10	MGK Crisis	182
11	Skyjack of Russian Airplane	32
12	Cancellation of immunity of representative Mustafa Bayram	22
13	MGK March meeting	14
14	Fire in Traffic Hospital	11
15	İsmail Cem in the USA	17

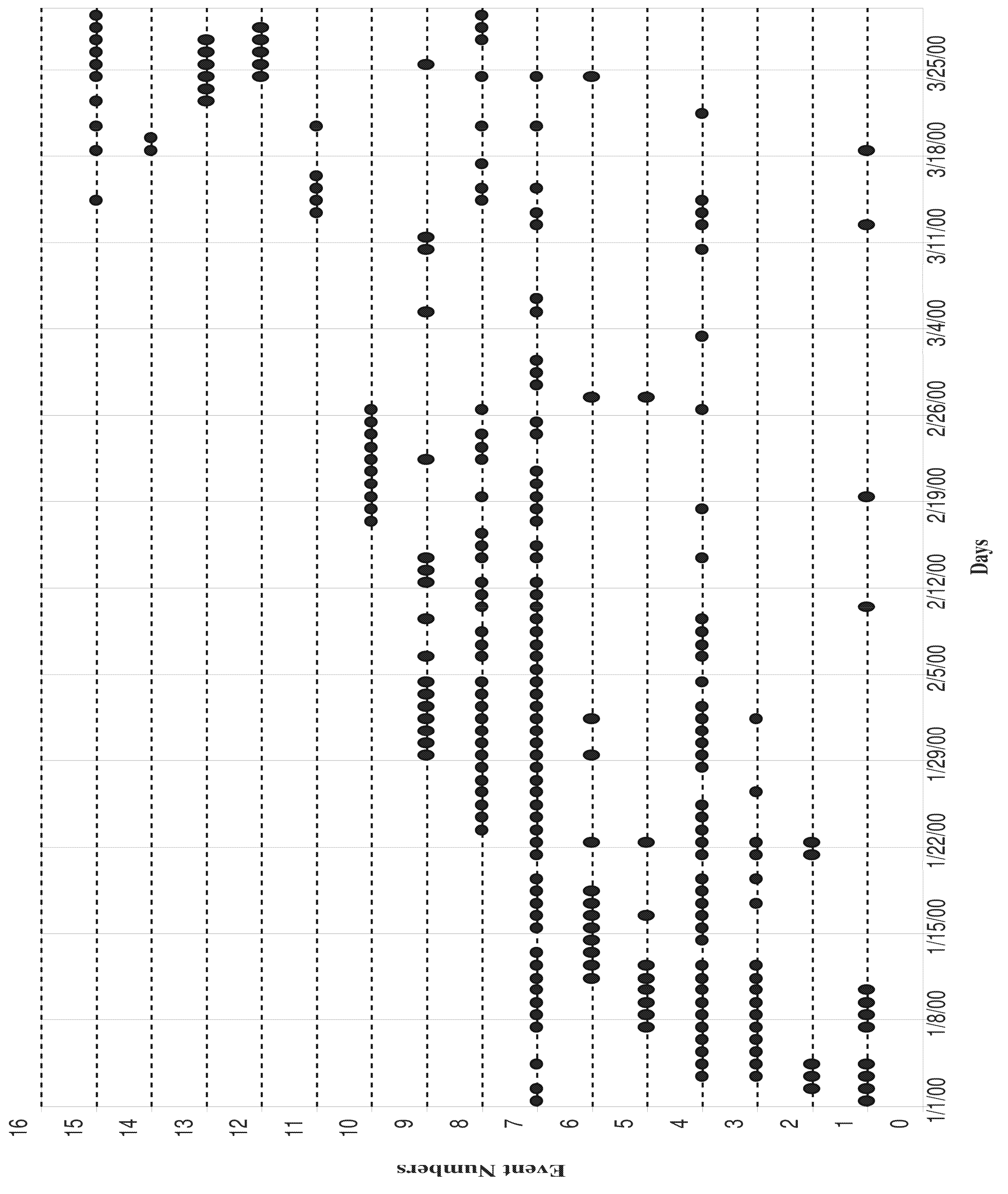

Figure A.3: Distribution of events across the corpus.