Topic Detection through Statistical Methods

by

Frederick George Walls

B.S., Massachusetts Institute of Technology (1998)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineeering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999

Author .....
Department of Electrical Engineering and Computer Science
January 29, 1999

Certified by .....
Victor Zue
Associate Director of LCS and Senior Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Topic Detection through Statistical Methods

by

Frederick George Walls

Submitted to the Department of Electrical Engineering and Computer Science
on January 29, 1999, in partial fulfillment of the
requirements for the degree of
Master of Engineeering in Electrical Engineering and Computer Science

Abstract

A system is developed to group news stories together according to topic. Several clustering algorithms can be used to group related stories into clusters. The clustering algorithms used require two types of metrics: metrics that, given a story and a set of clusters, can find the most topical cluster for that story; or metrics that can help decide whether or not a given story is on the same topic as a cluster. These metrics are derived by combining simple similarity metrics that compare stories and groups of stories. Finally, methods are proposed for evaluating the story groupings, and experimental results are reported based on these methods.

Thesis Supervisor: Victor Zue
Title: Associate Director of LCS and Senior Research Scientist

Acknowledgments

I would like to thank Richard Schwartz, Michael Schmidt, and all the people at BBN Technologies for their technical assistance, patience, and support. Their experience and brilliance were invaluable in the completion of this work. Moreover, they were a tremendous bunch of people to work with.

I would also like to thank Victor Zue. His efforts and patience were crucial for me to finish my degrees.

Many thanks to the brothers of Pi Lambda Phi for making my college experience truly outstanding. I will carry the memories with me for not four years but a lifetime.

I would especially like to thank the people whose love and support has kept me going through the years: my mother for her constant faith in me, my dad for his good advice, my brother George and sister Katherine for putting up with me for all those years, and the rest of my relatives. Finally, my profound thanks to Carrie Daake for brightening my day for three years and counting.

This work was supported by the Defense Advanced Research Projects Agency and monitored by Ft. Huachuca under contract No. DABT63-94-C0063 and by the Defense Advanced Research Projects Agency and monitored by NRaD under contract No. N66001-97-D-8501. The views and findings contained in this material are those of the author and do not necessarily reflect the position or policy of the Government and no official endorsement should be inferred.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The recent explosion in textual information has created a need for methods to automatically organize text documents. New technologies such as the World-Wide Web and advances in speech-recognition have fueled a significant growth in the range of textual information. This growth has engendered a drive for techniques to organize and classify large amounts of text. For these reasons, this work addresses the problem of automatically organizing text documents into groups related by topic.

The problem is addressed in the context of the government-sponsored Topic Detection and Tracking (TDT) research effort. TDT is involved with the identification of broadcast news stories (both in text and audio) that share the same topic. TDT consists of three separate tasks: segmentation, tracking, and detection. This work focuses on the *detection* task, which consists of grouping together stories related by topic. Hence, the following is a discussion of the theory and design of a high-performance topic detection system.

This thesis is organized into six chapters. The problem is surveyed and the basic system design is outlined in Chapter 1. The clustering algorithms described in Chapter 2 create clusters of topic-similar stories. Chapter 3 describes metrics for comparing stories with one another and for comparing stories with clusters. Chapter 4 develops the two classes of metrics used in the clustering and shows how to combine the similarity metrics described in Chapter 3 to create clustering metrics. Chapter 5 discusses methods for evaluating the clustering, and then provides the results of

experiments conducted using the designed system. A brief summary and conclusions are provided in Chapter 6.

1.1 Overview

1.1.1 What are TDT and Detection?

The TDT initiative is a performance-driven DARPA-sponsored research effort. It is aimed at advancing the state-of-the-art in technologies that find pieces of information that are on the same topic. Many sites compete in formal evaluations that are intended to compare the performance of various research systems. The evaluations provide a basis for comparing the participating sites' system performance.

The goal of a TDT system is to be able to accurately find *topics* of interest and track their course through news stories. For the purposes of this task, a topic is defined as "a seminal event or activity, along with all directly related events and activities" [16]. The systems that best match the human annotators' judgements about the corpus receive the best scores in the evaluations.

TDT ultimately hopes to address all written and spoken news broadcasts, regardless of language. Therefore, the eventual system will draw from a large body of research on many aspects of speech and language processing. Although the focus is currently limited to English news broadcasts, TDT will incorporate Mandarin and Spanish broadcasts in 1999.

The TDT research initiative is divided into three separately evaluated tasks. The *segmentation* task involves finding the boundaries between stories in the audio newscasts. The *tracking* task requires a system to find the stories that are on the same topic as a small set of training stories (see [10]). Finally, the *detection* task involves the unsupervised grouping of stories on the same topic.

The detection task is constrained in a number of ways. First, each story may only be included in exactly one cluster. This constraint implies that each story contains information primarily only on one topic. Although many stories contain information

on multiple topics, this assumption is still necessary to simplify an otherwise extraordinarily complex problem. Second, the detection must be performed in a causal manner; i.e., the stories must be processed sequentially and clustering decisions must be made within a specified time interval. Therefore, early incorrect decisions can hurt performance more than later mistakes. This also means that the order stories are processed in can significantly affect system performance.

1.1.2 Motivation

Topic detection is an important area of research for several reasons. First, as the sheer amount of information grows in today's society, the need for improved methods to organize and browse that information expands. Topic detection produces a particularly useful model for grouping related information. Once the information is grouped, it is much easier to index and retrieve specific data items.

The applications for topic detection technology are far-reaching. Efficient information retrieval systems use similar clustering methods to narrow the search space for large data sets [6]. Clustering also provides a means for producing related documents even if a query is poorly or abstractly formulated. In other words, if a user is interested in a particular document, it is likely that the user would also be interested in other documents from the same cluster. This concept is also known as the *clustering hypothesis* [23].

Topic detection is useful for finding novel events or topics — it can initiate a response if news "breaks". New topics are indicated simply by a new cluster in the system. Finding novel topics can be important if the document stream contains a large amount of redundant information. A detection system could alert a human analyst to a new topic that has surfaced. The analyst could then determine whether or not the new topic is important and act accordingly.

Finally, topic detection can be seen as an efficiency tool. For example, if an analyst is not given prior knowledge about which stories are interesting, he could save effort by discarding entire clusters of stories by judging a few stories from that cluster to be uninteresting. Therefore, a detection system saves the labor cost of looking at

redundant or similar information.

1.1.3 Differences from Document Clustering

Although it may be tempting to dismiss topic detection as a trivial extension to traditional document clustering, there are a few key differences between the two. First, a topic detection system utilizes source-specific information external to typical document clustering schemes. For example, systems are privy to information regarding what news source produced the story, as well as the time and date that the story ran. This information can be used extensively to augment the performance of a detection system.

Secondly, the constraints of the problem are somewhat different than that of the traditional clustering problem. Ordinary document clustering generally involves two steps: clustering documents already in the system in a retrospective manner and perhaps incrementally adding new documents to the existing structure in such a way as to minimally disrupt the rest of the clusters. The TDT detection task clusters all stories in a causal way — the clusters are constantly changing and detection decisions must be made immediately.

Perhaps the most important difference between topic detection and document clustering is the relationship between clustered documents. Because the definition of a topic in TDT includes "directly related events and activities," different threads in a TDT topic may contain very different material. In document clustering, the goal is generally to group documents together that would be produced by a single query. This goal differs from that of grouping documents together that share the same topic, because different queries may be required for topics in news stories that tend to change over time. For example, traditional document clustering might cluster stories regarding the subpoena of Secret Service agents and stories about the impeachment of President Clinton into separate clusters. Detection, on the other hand, is expected to treat those as one topic regarding the Monica Lewinsky scandal. This requires detecting the common elements, if there are any in the text, or tracking and adapting to the diverging threads as they evolve.

1.2 Goals

The research effort is undertaken with two main goals in mind. One goal of the research is to produce a system that performs well in the TDT evaluation. Secondly, the designed system should be useful as a component technology, and it should perform well under a variety of circumstances.

1.2.1 Win the Evaluation

First and foremost, the object of the game is to win. While this is not in itself a particularly useful way of metering the success or failure of this venture, strong performance in the evaluation is evidence that the technologies developed for detection are effective.

1.2.2 Design a Useful System

Another important goal of this work is to generate a usable and useful system that could be used for the applications described above. Although our goal is to create a component technology whose performance can be measured independently of the application, good performance in a number of settings would show that the technology is more broadly applicable. A good system should perform well under a variety of circumstances, although this is not a requirement for TDT.

1.3 System Design

A block diagram of the topic detection system is given in figure 1-1. This shows the main components of the system: the clustering algorithm, preprocessing, similarity metrics, and metric combinations for selection and thresholding.

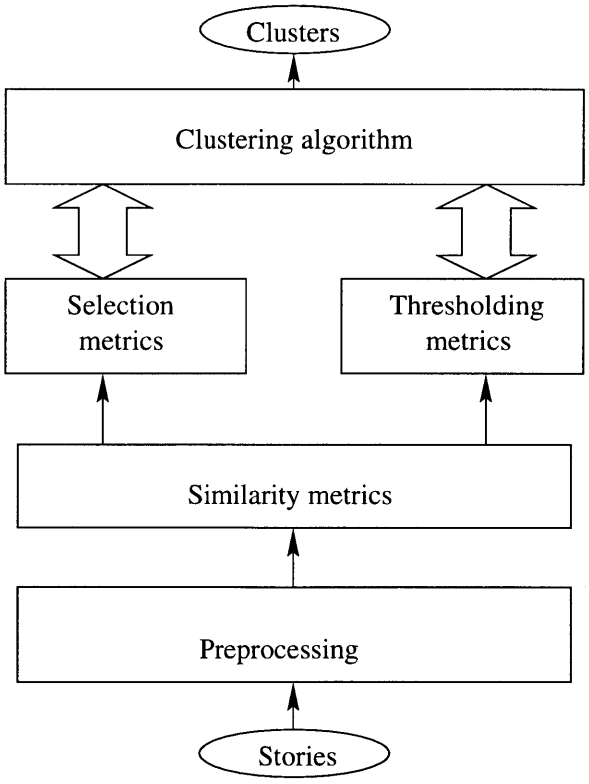


Figure 1-1: Block diagram of the detection system

1.3.1 Clustering

The clustering process uses clustering metrics to produce clusters of similar-topic stories. The clusters can then be compared with new stories to determine which cluster the story best fits within, if any. Clustering, by definition, enforces mutually exclusive clusters as required by the task constraints; therefore, it is the most appropriate framework for building groups of similar-topic stories.

We examine a number of useful clustering algorithms for causally clustering stories, as well as those that allow the system to look ahead at future stories. These clustering algorithms are discussed extensively in Chapter 2.

1.3.2 Preprocessing and Similarity Metrics

For clustering, we require a method for determining whether or not two stories are related by topic. Hence, similarity metrics can be used as the basic building blocks for clustering metrics. Although similarity metrics differ significantly, all require the critical first step of preprocessing.

Preprocessing

Preprocessing involves converting all the different possible sources of data to a common format that subsequent processing steps can deal with. The system preprocesses each source differently. This is discussed in detail in Chapter 3.

Audio sources require the most amount of preprocessing. First, a speech recognizer is used on the data to acquire a text transcript of the stories along with confidence scores for each word. The confidence scores can be used to filter or deemphasize words that the speech recognizer is unsure of. Segmentation is also required to determine the boundaries between stories in audio, although the system described herein uses the high-quality manual segmentation. As an alternative to speech-recognized text, the manually transcribed closed-captioned text associated with the source may be used.

Similarity Metrics

One could conceive of a vast number of metrics to compare a story with another single story or with a cluster of stories. In Chapter 3, we describe two types of similarity metrics. First, we discuss vector-space metrics, which are based on modelling the stories as vectors in a large-dimensional space. We also consider probabilistic similarity metrics, which formulate the similarity problem in terms of probability models.

1.3.3 Combining Similiarity Measures

The clustering algorithm used requires specific types of metrics that can be generated by combining similarity metrics and other information in different ways. We are concerned with finding appropriate metrics for two main clustering tasks. The first, *selection*, is concerned with finding the most similar cluster to a story. The second task is *thresholding*, which is the more complicated decision of whether a story is relevant to a given cluster. Metric combinations are discussed in detail in Chapter 4.

1.4 The Evaluation

Once a system generates a group of clusters, its performance must be evaluated. Unfortunately, there is no obvious way to evaluate a clustering without a clear notion of how the clusters may eventually be used. Therefore, we shall consider a number of methods for evaluating partitions including the official TDT evaluation metric, a metric based on detecting the first story in a topic, and a metric based on the utility of clustering to a human analyst. We provide the results of using different metrics and justify the effectiveness of our topic detection system.

Chapter 2

Clustering

One major component of the system is the clustering algorithm. The clustering algorithm uses the metrics developed in Chapter 4 to form clusters of stories related by topic. Methods for evaluating the clusters generated by the algorithm are presented in Chapter 5.

First, a brief survey of clustering algorithms is presented, including those used by others for topic detection. Then the goals and constraints for a TDT clustering algorithms are outlined. A simple, causal clustering algorithm known as *incremental clustering* is described, followed by methods to extend incremental clustering to look ahead into future stories as allowed by the TDT evaluation.

2.1 Clustering Background

The following section contains some background information about clustering. First, different methods are surveyed for the unsupervised clustering of data into groups. This is followed by a discussion of clustering algorithms currently used for topic detection.

2.1.1 Clustering Data into Groups

Much has been written on the subject of clustering data into groups (e.g., [7], [11], and [15]). Types of algorithms include single-pass clustering, agglomerative clustering, divisive clustering, fuzzy clustering, and neural network clustering. Single-pass clustering involves processing the set of data items once and dealing with each item one at a time. Agglomerative (bottom-up) clustering involves recursively merging close clusters. Divisive (top-down) clustering involves recursively splitting the data to generate clusters. Fuzzy clustering assigns data items to clusters in a probabilistic way. Finally, neural network clustering involves using a neural network architecture to group data items together [14].

2.1.2 Popular Detection Clustering Algorithms

Several clustering algorithms have been used for the TDT detection task. Incremental clustering is used by a number of participating sites [2]. It is a single-pass algorithm that produces mutually exclusive clusters. Incremental clustering is described in more detail in section 2.3.

Hierarchical agglomerative clustering techniques such as group average clustering (GAC) are also used for topic detection [2]. The algorithm involves recursively merging the closest clusters, where singleton stories are treated as clusters. This algorithm and its application to detection are described in section 2.4.1.

Another algorithm in common use is the incremental k-means clustering algorithm [2]. This algorithm involves assigning stories to clusters in an iterative fashion. The k-means clustering algorithm is outlined in section 2.4.2.

One other unique clustering algorithm that is used is a novelty detection and tracking approach [2]. The algorithm finds the novel stories (those containing new words or phrases) and tracks their course through time. Although this algorithm has some interesting properties, it does not enforce mutually exclusive clusters. We did not consider this algorithm in our system design.

2.2 Goals and Constraints

Not every clustering algorithm can be used for topic detection. Because of the large number of algorithms that have been developed for the purpose of clustering, they must be judged based on some standards. This section is divided into two parts: first, the goals for the algorithms are discussed; then, the constraints of the problem are given.

Clustering Goals

First, we would like to have a clustering algorithm that does not require a great deal of computation. Many clustering algorithms are computationally expensive, especially when many items need to be clustered. Because the corpus is large, computationally expensive algorithms can consume an impractical amount of resources. Therefore, the space and processing requirements of a clustering algorithm should be reduced to a practical level.

One goal is that the algorithm adapt clusters over time. When a story is deemed to be relevant to a cluster, the cluster can be adapted to account for the inclusion of the new story. This important property can create a very powerful representation for topics in the system, because it helps incorporate changes in the topic focus over time.

The clustering algorithm should be stable. In other words, the clusters should not migrate to different topics when new stories are added to the clusters. The stability must be balanced with the the tendency to adapt, however, because these properties compete with one another.

Clustering Constraints

First, according to the rules of TDT, the algorithm must be causal with a short look-ahead — decisions about a particular story must be finalized using past stories and a small number of later stories. This constraint eliminates algorithms that require random access to the corpus.

Second, the algorithm must be able to deal with a lack of prior knowledge about the final clusters. For example, the system is unaware of which topics the annotators have selected or the final number of clusters in a corpus. This constraint forces the system to discover the size and number of clusters. This is important given that human-annotated clusters can vary from one story to several hundred in size.

2.3 Incremental Clustering

One of the simplest clustering algorithms is the *incremental clustering* algorithm. This algorithm processes stories one at a time and sequentially, and for each story it executes a two-step process (shown in figure 2-1):

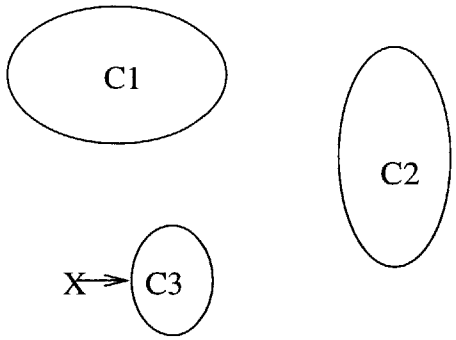
1. Selection: The most similar system cluster to the story is selected.
2. Thresholding: That story is compared to the cluster, and the system decides whether to merge the story with the cluster or to start a new cluster.

Although the algorithm is simple, it is within the constraints of the topic detection problem. It is a causal algorithm, as decisions are made once and in order. It represents clusters in a flat way, and the quantity of clusters and their sizes are determined dynamically as the corpus is processed.

There are also a number of drawbacks to this approach. Decisions can only be made once, so early mistakes based on little information can be costly. Secondly, the computational requirement grows as the stories are processed. At the end of the corpus, the system may have several thousand clusters to compare each story with.

2.4 Utilizing Look-ahead

Although a simple causal algorithm such as incremental clustering is easy to implement and describe, the evaluation allows systems to look ahead at a given number of stories before making decisions. We shall now examine various methods for exploiting the look-ahead allowed in the evaluation.



Step 1: Find closest cluster
Step 2: Decide whether to merge

Figure 2-1: The two-step incremental clustering process (X = story, Cn = clusters)

2.4.1 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering is a method generally used for clustering with infinte look-ahead. Although the algorithm is computationally expensive, it performs better than any other algorithm in many applications. Agglomerative clustering performs the following steps (see figure 2-2):

1. The stories are all initially treated as singleton clusters.
2. Find the minimum distance between any two clusters subject to the constraint that at least one must be within the look-ahead period. Let this minimum distance be between clusters i and j.
3. If the distance between i and j is greater than some threshold, stop clustering.
4. Merge cluster i with cluster j.
5. Goto (2).

One interesting characteristic of this particular algorithm is that the resulting clusters are hierarchical in nature — stories that are essentially the same tend to appear at the bottom of the hierarchy, and clusters on entirely separate topics tend to be split among the top leaves in the hierarchy. This characteristic allows for an

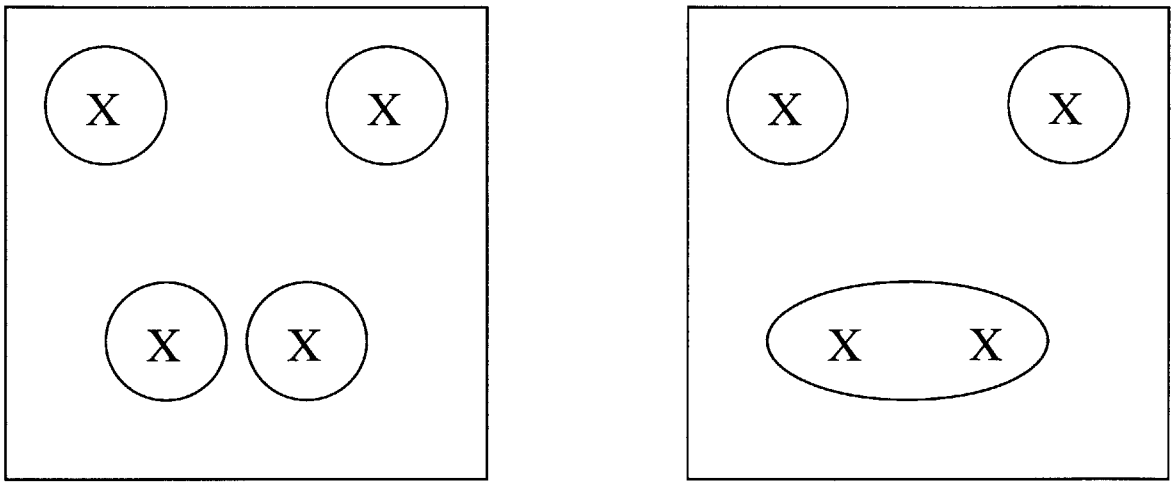


Figure 2-2: Hierarchical agglomerative clustering algorithm: find two closest clusters, then merge them together

intuitive representation of *scope* in the clustering. Scope is a qualititave assessment of the limit of how much information topics encompass.

Because systems are unaware of the scope of topics, conventional clustering methods encounter difficulties discerning whether the annotators worked on broad topics with lots of tangentially related events (e.g., Asian economic crisis, Monica Lewinsky scandal) or narrower topics with a very specific focus (e.g., retirement of Marcus Allen, a specific set of raids by Israel on Palestine). Because of this, the system attempts to group stories according to roughly the same scope. Unfortunately, it often fails on very broad and very narrow topics.

Generating a hierarchy is not very useful in the current TDT evaluation. To be advantageous, it requires a postprocessing step to cut the tree in an appropriate way, which would violate the evaluation condition. However, if a clustering threshold is used to stop the clustering, this algorithm is an effective but expensive method for exploiting look-ahead.

2.4.2 Incremental k-means Algorithm

Although it is similar, the following algorithm is not precisely a k-means algorithm because the number of clusters k is not given beforehand. This algorithm involves

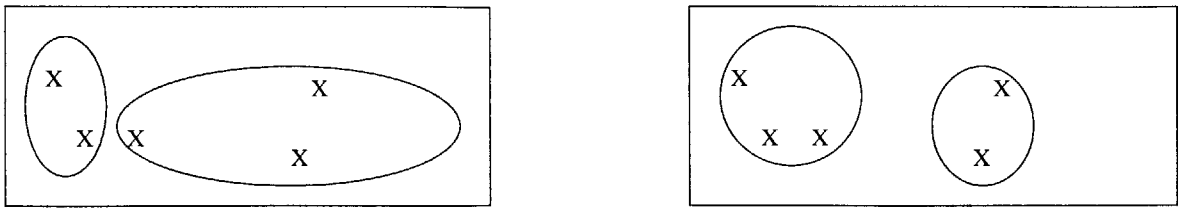


Figure 2-3: k-means incremental clustering: poor initial clusters can be corrected

iterating through the data that the system is permitted to modify and making appropriate changes during each iteration. More specifically:

1. Use the incremental clustering algorithm to process stories up to the end of the currently modifiable window.

2. Compare each story in the modifiable window with the old clusters to determine whether each 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 goto (1).

This algorithm (shown in figure 2-3) is able to restructure poor initial clusters but still process the corpus in a causal fashion with look-ahead. Because all the clusters are modified at once in step (3), the algorithm tends to be fairly stable. This algorithm also allows the number of clusters k to be a free parameter. The computational requirement is less imposing than the agglomerative clustering algorithm, especially for a larger look-ahead.

2.5 Discussion of Clustering Algorithms

Incremental clustering is very simple and fits within the constraints of the TDT evaluation. Because it is the least computationally intensive of the all the methods

described above, it is most appropriate under limits of computing resources. Fortunately, the TDT evaluation specification does not impose any computational constraint for the algorithms used. In addition, incremental clustering has the significant drawback of order dependence; i.e., early decisions are more difficult because there are fewer clusters to compare with. This can hurt the system performance dramatically, because early incorrect decisions are generally costly.

Hierarchical agglomerative clustering performs slightly better than incremental clustering; unfortunately, it requires a much larger amount of computation and is unfeasible for a look-ahead of greater than a few dozen stories. Even for a very small look-ahead, agglomerative clustering takes much more computation than simple incremental clustering. Because of this problem, we reject agglomerative clustering for the system design.

The k-means algorithm also performs better than incremental clustering, although most of the improvement is gained from looking ahead a small number of stories. With a greater look-ahead, the performance changes very little. For the system described herein, we utilize the incremental k-means algorithm, because it requires significantly less computation than agglomerative clustering, but still prevents early incorrect decisions.

2.6 Chapter Summary

Although there are many clustering algorithms in common use, only a few are appealing to use for topic detection. Incremental clustering is computationally cheap and yields very good performance; unfortunately, early incorrect decisions can hurt performance substantially. Agglomerative clustering is an effective but expensive clustering algorithm that can be used where computational limitations are not imposed and research time is not an issue. Finally, the incremental k-means algorithm is somewhat more computationally intensive than incremental clustering. This algorithm performs well, and is a good balance of complexity versus accuracy.

Chapter 3

Similarity Measures

This chapter discusses various methods for finding the topic similarity between different stories and groups of stories. The measures described herein are used as the basis for the clustering metrics described in Chapter 4. Once measures of similarity between stories have been developed, these measures can be combined into thresholding and selection metrics useful for the clustering system. Therefore, the discussion of the strengths and weaknesses of each metric is deferred until Chapter 4.

First, the basic preprocessing necessary for the stories to be represented consistently in the system is discussed. Then, some vector-space methods for comparing stories with other stories are described. Finally, clusters are defined, and basic probabilistic metrics for comparing stories to clusters are outlined.

3.1 Story Preprocessing

Before similarities are calculated between stories, it is necessary to convert the stories into a consistent representation. This representation needs to incorporate prior knowledge about the corpus to eliminate information that is irrelevant to the story's topic. The representation also needs to keep as much relevant information as possible. We shall refer to these representations as *term vectors* or *feature vectors.*

3.1.1 Audio Conversion

Much information regarding a news story's topic can be gained by looking at the story's words. Therefore, audio news is transcribed to generate a text representation. Although methods exist to compare the similarity of untranscribed audio segments (see [17], [18]), they are beyond the scope of our exploration. Transcription allows the system to use a generic method to handle both text and audio sources. In TDT, transcripts are generated from audio in one of two ways: extraction of the story's closed-captioned text (CCAP) or automatic speech recognition (ASR). Closed-captioned text can easily be extracted from the television news programs that provide it, although the text contains a significant number of errors. The speech-recognized text is generated using a Large-Vocabulary Continuous Speech Recognition (LVCSR) system and is provided by Dragon Systems for the 1998 TDT data [8]. The speech recognition word error rate is approximately 23/

All transcripts are required to be segmented into stories. The story boundaries can be found automatically or manually; however, we assume that the story boundaries are provided manually. The automatic segmentation is currently not very accurate, and it is the focus of the segmentation task of TDT.

3.1.2 Term Vectors

The system requires a method for consistently representing the story texts. This section outlines options the system could use to represent text stories. These include counting words, word stemming, removing stop words, associating synonymous words, and finding the names of entities.

One convenient representation for stories consists of the counts for each different word, regardless of the order in which the words occur. Although this representation eliminates syntax, grammar, and word order, it maintains important defining details about a topic (e.g., nouns, actions). However, it also counts equally the occurrence of topic-irrelevant words within a story.

Often, simply counting words is insufficient to accurately capture the relationships

between words. An improved method for generating word counts involves techniques such as stemming, removing stop words, and using a thesaurus. Stemming is the removal of suffixes from a word leaving the word stem. Stemming helps associate different forms of semantically related words. For example, the words "bombing", "bomber", and "bombs" all have the same stem "bomb". The stemming process requires some additional processing beyond suffix removal, because certain words do not stem properly when the suffix is removed. For example, the stem of "hoping" is "hope", but simply removing the suffix "-ing" would generate "hop". Therefore, several rules are typically used by a word stemmer [19]:

1. Restore the silent "e" after removing the suffixes from certain words.

2. Delete certain doubled consonants after suffix removal. For example, "stemming" is transformed to "stemm" by suffix removal, then to "stem" by removing the doubled consonant.

3. Substitute a "y" for "i" in the final position for certain words. For example, "speedier" turns into "speedi" after suffix removal, then "speedy" after applying this rule.

4. Use a dictionary for exceptions to the stemming rules given above.

Stop word removal involves ignoring words that usually offer no discriminating power because they are common to most news stories (e.g., "the", "so", "because"). Stop word removal helps performance, because it eliminates words that are not helpful in determining a story's topic. This is important because the occurrence of stop words can vary substantially between different stories. This variation effectively adds a great deal of "noise" to the story comparisons. To alleviate this problem, stop words can simply be taken out based on a predefined list of words common to all topics [19].

Finally, a thesaurus can be used to broaden terms that are too narrow by combining terms that are synonymous or closely related (e.g., "U.S.", "United States"; "bravery", "courage") [20]. A thesaurus can help factor out a story author's word

choices. Although this technique is used successfully in other systems, the detection system presented here does not use a thesaurus.

Stories can also be represented by the names of the entities mentioned within. For many topics, names uniquely determine what stories belong (e.g., a story containing "Timothy MacVeigh" and "Oklahoma City" would likely discuss the topic of the 1995 bombing of the Murrah Federal Building in Oklahoma City). Once the names of story entities have been found, they can be used to either group sets of words together (e.g., "United Nations", "James Earl Ray", "Cambridge, Massachusetts") or simply to emphasize the terms corresponding to names. The subject of how to find named entities is beyond the scope of this dissertation; however, in section 5.3.7 we present the results of some experiments using the BBN IdentiFinder, a Hidden Markov Model (HMM) based named-entity extractor [4].

3.1.3 Term Weighting

One method for improving the story representations is to emphasize terms that are better for distinguishing a story's topic. Larger weights should be associated with terms that are more important, whereas smaller weights should de-emphasize those terms that are likely to offer very little discriminating power. This section surveys a number of methods for term weighting: confidence score weighting, inverse document frequency (IDF) weighting, and time-based weighting.

A simple weight associated with ASR text involves the use of confidence scores to emphasize terms that the speech recognizer believes it recognized correctly. A confidence score is a number between 0 and 1 that reflects the belief that the speech recognizer correctly identified the occurrence of each word. By de-emphasizing words that were more likely to have been misrecognized, the system incorporates incorrect words less into the story representations. In doing so, comparisons between stories become more accurate.

Another term weighting scheme involves using an *IDF weight* [19]. The IDF (inverse document frequency) weight is given by:

$$IDF_i = \log \frac{N}{df_i}, \tag{3.1}$$

where N is the number of documents, and df_i is the number of documents that the term i appears in. The counts in the term vector can now be weighted by IDF_i. Terms that appear in most or all of the documents in a collection are then strongly de-emphasized, whereas terms that appear in only a few stories are emphasized. This technique is interesting because it is based on the story statistics rather than individual word statistics.

Techniques specific to broadcast news can also be used for term weighting. For example, when several stories are combined into a term vector, terms that occur in old stories can be de-emphaszied in favor of terms from newer stories. This allows the topic focus to change over time.

3.2 Comparing Stories to Stories

A vast number of measures for computing the similarity of a story to another story have been developed. We shall reflect on a number of these measures and adopt the ones that perform well. There are a number of traditional IR measures that are used to measure the similarity between stories. These are based on traditional information retrieval (IR) vector-space models: a cosine distance and an IDF-weighted cosine distance.

3.2.1 The Cosine Distance

One class of IR metrics, based on a vector-space model for stories, has been used in many topic detection systems that have been developed [2]. These metrics treat the stories as vectors and attempt to compute the similarity between two stories as the cosine of the angle between the two vectors. The vector components generally correspond to the terms in the stories. More explicitly, let $\vec{q_i}$ be a vector with dimensionality of the number of words in the corpus, and let $[\vec{q_i}]_j$ be the jth component of

vector $\vec{q}_i$. Then, we let

$$[\vec{q}_i]_j = \log(tf_{ij} + 1), \tag{3.2}$$

where tf_{ij} is the term frequency (i.e., word count) corresponding to term j in story i. A simple cosine distance between story k and story l can be expressed as

$$D_{kl} = \cos(\angle(\vec{q}_k, \vec{q}_l)) = \frac{\vec{q}_k \cdot \vec{q}_l}{\|\vec{q}_k\|\|\vec{q}_l\|}. \tag{3.3}$$

This is one formulation for a cosine distance. The log function *smooths* the story vectors, so that single words that occur a large number of times do not dominate the score D_{kl}. Other research systems use different formulations for $\vec{q}_i$, though they all involve finding the cosine of the angle between two smoothed term frequency vectors ([1], [2]).

3.2.2 IDF-weighted Cosine Distance

The cosine distance can be combined with the IDF weight. First, let

$$\left[\vec{q'}_i\right]_j = \log(tf_{ij} + 1) \cdot \log \frac{N}{df_j}, \tag{3.4}$$

where df_j is the number of documents in the corpus containing the term j and N is the total number of documents in the corpus. The vector $\vec{q'}_i$ is the IDF-weighted term vector. The following two formulations can then be used:

$$D_{kl} = \cos[\angle(\vec{q'}_k, \vec{q'}_l)], \tag{3.5}$$

which is the cosine of the angle between two IDF-weighted term vectors; or

$$D_{kl} = \frac{\cos[\angle(\vec{q'}_k, \vec{q}_l)] + \cos[\angle(\vec{q}_k, \vec{q'}_l)]}{2}, \tag{3.6}$$

which is the symmetrized cosine of the angle between one regular and one IDF-weighted term vector. Equation 3.5 counts the IDF weight twice, and therefore the

score tends to be dominated by the weight. The distance in equation 3.6 counts the IDF weight only once; therefore, the system utilizes equation 3.6 when the cosine distance metric is called for. Note that we could have also found the cosine of the angle between two vectors weighted by the square root of the IDF weight, although this measure is very similar to the symmetrized measure described above.

3.2.3 Discussion of Story-Story Measures

The cosine distance metrics have a number of desirable properties. First, they are inherently normalized — the score is guaranteed to be between zero and one. A score of zero indicates that the stories have no words in common; a score of one indicates that the stories are identical. The metrics are also symmetric; therefore, it does not matter whether story A is compared to story B or vice versa.

Unfortunately, the cosine distance also requires the story words to overlap significantly to produce a good score. Therefore, it is not an effective metric for finding stories that are tangentially related to a topic.

3.3 Clusters

Before we can consider measures that compare stories to clusters, we first show why we should treat stories and clusters differently. The following section describes the motivation for and implementation of story-cluster measures.

3.3.1 What is a Cluster?

A cluster is a collection of stories that the system believes share the same topic. There are a few important notes about clusters:

1. Clusters are defined in a mutually exclusive way. Each story can belong to one an only one cluster. This assumption is a simplifying one and in general does not hold true, although it reduces the complexity of an otherwise unsolvable problem. For example, a story regarding an election campaign might contain

bits from a candidate's speech on a particular crime or disaster. Because the clusters must be mutually exclusive, the system can group the story with one and only one of the covered topics.

2. Clusters must be generated as the data is processed — the system has no *a priori* knowledge of what the clusters should be. Because of this, the system must adapt clusters over time. This is accomplished by merging a story with its related cluster as soon as it is deemed relevant.

Because clusters and stories are fundamentally different, it is worthwhile to consider a metric where clusters and stories are treated differently. In contrast, the cosine distance metric and other similar metrics treat stories and clusters the same. For example, a story could be compared to a cluster simply by substituting for a story vector a vector created from the concatenated cluster stories (see section 3.4. However, this symmetry seems unsatisfying in light of the fundamental differences between stories and clusters.

3.3.2 Previous Work in TDT

Comparing stories to clusters directly is not a new idea. One metric currently in use for detection is a variant of the Kullback-Liebler (KL) distance between the story and cluster distributions [2]. Given story S_k and cluster C_l, the distance metric D_{kl} is the sum of the KL distance between S_k and $S_k + C_l$ and the KL distance between C_L and $S_k + C_l$. This formulation is improved by smoothing the cluster distribution with a corpus background distribution, subtracting a story-background distance, and adding a time decay term. This new metric is given by

$$D'_{kl} = \sum_n \frac{s_{kn}}{|S_k|} \log \frac{u_n/|U|}{c'_{ln}/|C_l|} + decay, \tag{3.7}$$

where s_{kn} is the story word count for word n, c'_{ln} is the smoothed cluster count, u_n is the background model word count, $|S_k| = \sum_n s_{kn}$, $|C_l| = \sum_n c'_{ln}$, and $|U| = \sum_n u_n$.

3.3.3 Probabilistic Measures

The cosine distance, while traditionally effective for IR, is somewhat *ad hoc* and not obviously extensible. Probabilistic models, on the other hand, offer a formal way of expressing the quantities computed. Moreover, as shown in section 3.3.4, the probalistic models we propose show a satisfying similarity with the traditional vector-space approach. Hence, the class of metrics we consider for comparing stories with clusters is based on probability theory.

We consider two classes of probabilistic measures for comparing story S with cluster C:

1. Estimate a model for the cluster, then find the probability that the story S was generated from the cluster model C.

2. Estimate a model for the story, then find the probability that the cluster C was generated from the story model S.

In the first case, we are trying to calculate $p(C|S)$ where S is the story and C represents the cluster. In the second, we calculate $p(S \text{ is } R|C)$, which is the probability that S is relevant given the topic model.

The next sections discuss two probabilistic metrics: the BBN topic spotting metric and the BBN IR metric [24].

BBN Topic Spotting Metric

A useful set of metrics for topic detection is the class of metrics that calculate $P(C|S)$. We shall analyze one particular example of such a metric, the BBN topic spotting metric.

The BBN topic spotting metric is derived from Bayes' Rule:

$$p(C|S) = \frac{p(C) \cdot p(S|C)}{p(S)}, \tag{3.8}$$

where $p(C)$ is the *a priori* probability that any new story will be relevant to cluster C. If we assume that the story words s_n are conditionally independent, we get:

$$p(C|S) \approx p(C) \cdot \prod_n \frac{p(s_n|C)}{p(s_n)}, \tag{3.9}$$

where $p(s_n|C)$ is the probability that a word in a story on the topic represented by cluster C would be s_n.

We model $p(s_n|C)$ with a two-state mixture model shown in figure 3-1, 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. That is, we have a generative model for the words in the new story.

To calculate the distributions of the states, we use the Maximum Likelihood (ML) estimate, which is the number of occurrences of s_n among the topic stories divided by the number of words in topic stories. This estimate must be corrected for the weakness that the unobserved words for the topic have zero probability. Therefore, the model can be smoothed with a "back-off" to the General English model:

$$p'(s_n|C) = \alpha \cdot p(s_n|C) + (1 - \alpha) \cdot p(s_n) \tag{3.10}$$

The estimates for the general English state distribution and topic state distributions can be refined using the Expectation-Maximization (EM) algorithm [21]. This process allows new words to be added to the distributions and emphasizes topic-specific words. Therefore, the EM algorithm automatically assigns higher probabilities to words that are specific to the topic.

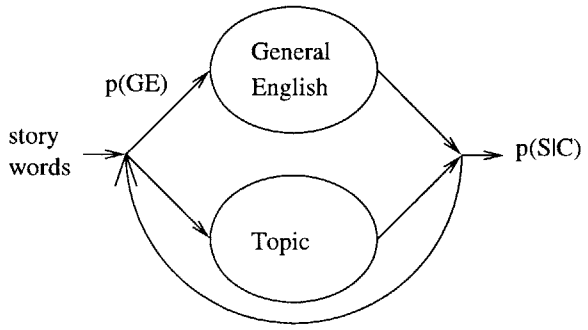


Figure 3-1: BBN topic spotting metric two-state model for a topic

BBN IR Metric

The BBN IR metric is used successfully for the Text REtrieval Conference (TREC) IR evaluation, where only one query is used at a time [12]. It looks at the problem in the opposite way. Given a query Q (consisting of the words in the cluster), we want to estimate the probability that any new story S is relevant to the query. But in this case, we assume that the query was generated by a model estimated from the story:

$$p(S \ \ is \ \ R|Q) = p(S \ \ is \ \ R) \cdot \frac{p(Q|S)}{p(Q)}. \tag{3.11}$$

Assuming independence of words in the query, we have:

$$p(S \ \ is \ \ R|Q) \approx p(S \ \ is \ \ R) \cdot \prod_n \frac{p(q_n|S)}{p(q_n)}. \tag{3.12}$$

Again, we use a two-state model (shown in figure 3-2), where one state is a unigram distribution estimated from the story S, and the other is the unigram distribution from the whole corpus.

This metric computes a product over the number of words in the cluster. Therefore, this metric is not appropriately normalized for comparing a number of different clusters to a single story, because the number of words in each cluster is different. In addition, the IR metric performs poorly when the clusters become large, because a single story cannot generate many stories in large cluster. Therefore, we did not explore using this metric to compare a single story to one cluster.

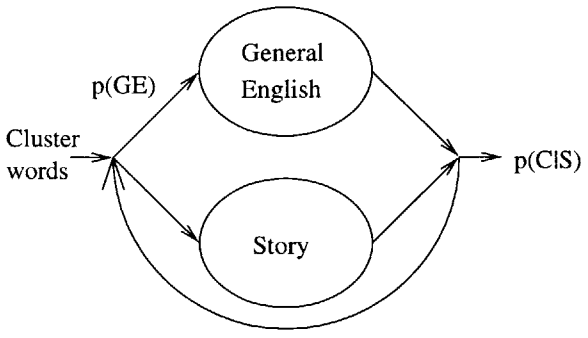


Figure 3-2: BBN IR metric two-state model for a story

3.3.4 Discussion of Probabilistic Metrics

One intuitively satisfying note about these probabilistic models is that they share a striking similarity to the cosine distance metric. Note that both equations 3.9 and 3.12 contain a product term over the words in either the story S or cluster C. Taking the logarithm of the likelihood in both cases yields a sum over the words in either S or C. However, the cosine distance is simply a sum over the words in common between S and C. Therefore, we see that we have developed probabilistic measures similar to the vector-space models; but they treat stories and clusters differently and can be extended formally.

3.4 Extension of Story-Story Measures

If we assume that clusters are made up of relationships between stories, a number of methods can utilize these relationships as the representation for clusters in the system.

Many methods for scoring a story against a cluster can be found in the clustering literature: average linkage, single linkage, and complete linkage [9]. Average linkage scores the story as the average score between a story of interest and each cluster story. In other words, if a cluster contains two stories A and B, the average linkage score between the cluster and story S is simply the average of the similarity between A and S and the similarity between B and S.

Single linkage and complete linkage are somewhat similar to each other. Single linkage takes the best similarity score between the story and cluster stories. Complete linkage takes the worst similarity score of those derived from comparing a story to the individual cluster stories.

Another method for combining the scores is to simply calculate one score between the story and the concatenation of all the stories in the cluster. For example, the cluster model for five stories on a particular topic would be one story with all the words from all five stories. The similarity score could be calculated between the story and cluster model.

Each of these techniques can be used for different interpretations of a cluster. If the clusters are viewed as random variations of the same basic topic, then either average linkage or story concatenation produce appropriate metrics for clustering. If the clusters are considered to be part of a rapidly-evolving topic, the single-linkage clustering algorithm should be chosen. Finally, if the clusters are limited in scope, the complete linkage method should be used for scoring the clusters.

Because of the presence of multiple-topic stories, the single-linkage technique is too unstable for our purposes. The complete-linkage is also inappropriate, because it constrains how far the stories can stray from the original topic. Therefore, we shall utilize the story concatenation method, because it is computationally simple and fits our model for the clusters.

3.5 Chapter Summary

This chapter discussed simple metrics for comparing stories to stories and stories to clusters. Preprocessing is an important step for generating a consistent representation for the stories in the system. These representations are used by both vector-space and probabilistic metrics. The cosine distance metric is among the class of vector-space metrics and is useful for comparing two stories with each other in a symmetric way. Probabilistic metrics provide a more formal way of comparing stories with clusters. Similarity metrics are combined using techniques described in Chapter 4 to be used for clustering.

Because we have not yet set criteria for evaluating different similarity metrics, it seems somewhat premature to attempt to measure the relative performance of different metrics. We therefore defer this discussion until the next chapter, where goals for clustering metrics are carefully outlined. The relative merits of the different metrics can then be evaluated and combined into metrics useful for clustering the data.

Chapter 4

Combining Similarity Metrics

Chapter 3 outlined various similarity measures between stories and clusters. These measures are useful under different circumstances because they work in different ways. We would like to combine the similarity measures to exploit their respective strengths and weaknesses. Therefore, in this chapter we describe methods for combining different similarity measures to produce good clustering metrics.

First, we outline the desirable characteristics for the two types of metrics useful for developing a clustering system: selection metrics and thresholding metrics. For a selection metric we suggest an intuitive probabilistic measure that performs well in a simple experiment. For a thresholding metric we suggest a number of techniques for combining similarity measures and evaluate these techniques based on their merit and performance.

4.1 Clustering Measures Background

Previous work in topic detection involves the use of one metric for making all clustering decisions [2]; however, by breaking the clustering problem down, we can develop different metrics for each clustering task. Recall that in Chapter 2 we describe incremental clustering and incremental k-means algorithms. Both of these require two basic types of metrics: a metric that finds the closest cluster to a story and a metric that indicates whether or not a story should be merged with a cluster. At this point,

it is worth specifying exactly what each type of metric should accomplish.

First, we use the notation $D(S, C)$ to signify a distance metric comparing a story S to a cluster C. Consider a story S and clusters C_1 and C_2, such that S is judged to be on the same topic as C_1 but not C_2. Then a decision metric should minimally obey the following rule:

$$D(S, C_1) < D(S, C_2). \tag{4.1}$$

In other words, the metric D is appropriate for selecting the most topical cluster to a given story.

We intentionally exclude metrics of the opposite sense that find the most topical story to a given cluster. These metrics are inappropriate for the detection problem, because the clustering must be done in a causal way. In other words, because the system can randomly read and merge clusters but not stories, a metric that finds a particularly relevant story to a given cluster is not very useful.

A different type of desirable metric is a quantitative assessment of the relevance of a document to a cluster. This type of metric obeys the following relationship:

$$D(S, C) > T \tag{4.2}$$

if and only if S is on the same topic as C, where T is some predefined threshold. In other words, the metric indicates whether or not a single given story is relevant to a single given cluster. The metric score can be compared to a threshold to produce a decision regarding whether or not the story belongs in the cluster.

We shall refer to metrics that best satisfy equation 4.1 as *selection metrics* because given a story they select the most topical cluster. Metrics that best satisfy equation 4.2 are referred to as *thresholding metrics*, because they are compared with a threshold to yield a topicality judgement. These equations should be seen as goals rather than absolutes, as the notion of stories and clusters being related by topic is a very abstract one. In addition, this is not an exhaustive list of potentially desirable metric characteristics; however, these goals are easily defined and measured, and they provide a practical basis for a topic detection clustering algorithm.

As an aside, we assume in this formulation that the two tasks of selection and thresholding are independent of one another; however, this is not necessarily the case. Given that the thresholding metric is always used with the cluster that is closest to the story, the thresholding metric could incorporate that information. In other words, the threshold T could be modified based on how many clusters are close to S or how much further the second-closest cluster is from S. However, our system does not attempt to exploit this information.

4.2 Selection Measures

Fortunately, one of the metrics discussed in Chapter 2 behaves similarly to the the selection equation 4.1. As is shown in the following, the BBN topic spotting metric is appropriate for the selection problem. Moreover, a simple experiment suggests that this metric is more appropriate than the vector-space metric described in Chapter 2.

4.2.1 BBN Topic Spotting as a Selection Metric

The BBN topic spotting metric satisfies some of the goals of a selection metric. The selection problem can be formulated probabilistically as an attempt to find the most probable cluster given a story. In other words, from a set of clusters $C_1, C_2, \ldots, C_n$, we attempt to find k such that:

$$k = \arg\max_i p(C_i|S). \tag{4.3}$$

Assuming the clusters are *a priori* equally likely and combining with equation 3.9, the equation simplifies to:

$$k = \arg\max_i \prod_m p(s_m|C_i). \tag{4.4}$$

where s_m are the story words. Therefore, the selection metric could be chosen such that:

	Data set		
	TDT-1	TDT-2 (Jan-Feb)	TDT-2 (Mar-Apr)
Cosine dist.	1.32%	3.95%	0.18%
Probabilistic	0.09%	1.66%	0.00%

Table 4.1: Comparison of selection metrics according to misclassification rates for reclustered stories

$$D(S, C) = \prod_m p(s_m|C), \tag{4.5}$$

where $p(s_m|C)$ is computed according to the two-state mixture model. $D(S, C)$ is therefore a justifiable metric for doing cluster selection.

4.2.2 Experimental Evidence

To test the effectiveness of the BBN topic spotting selection metric, we attempted a simple experiment. From each of the TDT-1, TDT-2 Jan-Feb, and TDT-2 Mar-Apr corpora (described in section 5.1), a data set of human-generated clusters was extracted. Each cluster contained stories on one topic. Each story was removed from the data set one at a time and reclassified among the clusters in the data set. The story was reclassified according to the highest-scoring cluster. If the highest-scoring cluster was not the cluster the story was drawn from, it was counted as an error. We report results using both the cosine distance and the BBN topic spotting (i.e., probabilistic) selection metrics.

The misclassification rates for each data set are given in table 4.1. The table indicates that the probabilistic selection metric reclassifies a larger percentage of stories correctly for all data sets. This suggests that the probabilistic metric is a more likely candidate for the selection problem than the cosine metric.

4.3 Thresholding Measures

Designing an effective thresholding metric is a bit more tricky. We develop a number of metrics for thresholding, leading up to a probabilistic binary classifier.

The thresholding metric is discussed in the context of binary classification — given one story and one cluster, the story is either on the same topic as the cluster or not. The merit and effectiveness of several possible thresholding metrics are then evaluated.

4.3.1 Thresholding Metric Combinations

The goal of a thresholding metric is to determine whether or not a story should be merged with a cluster. Such a metric is important for virtually any clustering algorithm one could conceive, because it reveals whether or not a story belongs in a cluster. Therefore, we develop the following methods for combining scores and features from the system into an indicator about whether a story should or should not be merged with a cluster.

Note that this is a binary classification problem. Given two sets of data ("on topic" and "off topic"), the metric determines which set a new data point belongs to. There are several pitfalls associated with this approach:

1. The data points change over time. One classifier might be useful when the system contains few clusters, but another might be better after all the clusters have been generated. Therefore, the classifier must generalize to different types of clusters.

2. There is no obvious method for determining the threshold T. This is a difficult problem, because T should be determined by optimizing a particular cluster evaluation metric. However, because of the complicated steps between thresholding and evaluating the final clustering, estimating T is a formidible task. Therefore, for the purposes of this dissertation we ignore this problem and simply estimate the optimal T experimentally.

4.3.2 Normalization and Single-Variable Classifiers

One type of thresholding metric is the so-called "normalized score," which is based on normalizing a single metric. To be effective, the normalization must minimize the effects of story and cluster size. The drawback of this approach is that the normalized score is only generated by one similarity metric.

The cosine distance is naturally normalized — a score of 1 indicates that the stories are identical, and a score of 0 indicates the stories shared no common words. Therefore, the cosine distance metric is one metric that can be used for thresholding.

The BBN topic spotting metric is unfortunately not inherently well-normalized. The score varies with the size of the story compared. Fortunately, there are a few methods that can be used to normalize this metric.

For one normalization, we observe that the log probability produced by the topic spotting metric is proportional to the number of words in the story. Therefore, one possible normalization is to simply divide the log probability by the story length. While this produces a reasonable score, it is an *ad hoc* normalization.

Another normalization is to assume (by the Central Limit Theorem) that the log probabilities of a particular story S_i for different clusters are roughly distributed normally. This assumption can be made if we view the individual word probabilities as independent random variables and assume that the story has a reasonably large number of words. Then, let μ_i be an estimate of the mean of story log probabilities for cluster C and σ_i be an estimate of the standard deviation. Then, the normalized score for story S_i is given by

$$D(S_i, C) = \frac{\log p(S_i|C) - \mu_i}{\sigma_i}. \tag{4.6}$$

This normalization depends very little on the length of S_i, because any factor multiplying $\log p(S_i|C)$ would cancel after the normalization. This normalized score is also a reasonable thresholding metric.

4.3.3 Modeling Feature Distributions Parametrically

In this section, we treat the similarity metric scores as components in a feature vector that can be used as input to a classifier. This approach allows a trained classifier to decide what features are most important in the thresholding problem. Methods for acquiring feature vectors are discussed, followed by a description of several different parametric classifiers.

Feature Vectors

We need to consolidate the similarity metric comparisons into a feature vector that can be used as an input for the classification problem. The simplest way to do this is to combine scores from each comparison into a vector. We can then use the vectors as feature inputs to a classifier.

Although similarity metric results probably contain the most information about the story topic, other information could also affect the scores and/or the topicality of a story. Examples include story length, cluster length, story age, number of unique words, news source, etc. These quantities can also be used as components of the feature vectors.

Training the Classifier

An important requirement for building a classifier is sufficient annotated training data. Classifier training data consists of feature vectors and corresponding decision labels that indicate whether or not the story is on the cluster topic. Acquiring training data is a tricky subject for a few reasons. If we limit ourselves to using pure clusters (for instance, taken from human-labeled data) to acquire feature vectors, then the decision label for each can be diagnosed easily. Unfortunately, the implemented metric does not always compare stories to pure clusters, because the detection system sometimes makes clustering mistakes. If training data is created from impure clusters, then the corresponding decision labels are difficult to ascertain. The optimal classification for an impure cluster often even depends on the order in which the stories are processed.

We generate the training data using the latter of these approaches. Judgements about whether or not a story S should be merged with cluster C_i are based on optimizing the evaluation metric. The following criteria are used:

- If the story S is not annotated . . .
  - . . . and none of the stories in C_i are annotated, no judgement can be made.
  - . . . and some of the stories in C_i are annotated, the judgement is automatically "should not be merged".
- If the story S is annotated for reference topic R_j . . .
  - . . . and none of the stories in C_i are annotated for topic R_j, the judgement is automatically "should not be merged".
  - . . . and some of the stories in C_i are annotated for topic R_j, the judgement is based on choosing the maximum of the following:
    * Evaluation score given R_j is mapped to $C_i \rightarrow$ "should be merged"
    * Evaluation score given R_j is mapped to a singleton cluster containing $S \rightarrow$ "should not be merged"

Probabilistic Binary Classification

We preface this discussion by discussing generic probabilistic binary classifiers. We begin by describing two classes $\mathcal{C}_0$ and $\mathcal{C}_1$ corresponding to merging and not merging a story with a cluster. The techniques described below attempt to calculate $p(\vec{y}|\mathcal{C}_0)$ and $p(\vec{y}|\mathcal{C}_1)$ according to models, where $\vec{y}$ is the feature vector to be classified. We use these quantities to acquire a score S corresponding to:

$$S = \frac{p(\mathcal{C}_0|\vec{y})}{p(\mathcal{C}_1|\vec{y})} \gtrless T, \tag{4.7}$$

which can be written according to Bayes' Rule:

$$S = \frac{p(\mathcal{C}_0) \cdot p(\vec{y}|\mathcal{C}_0)}{p(\mathcal{C}_1) \cdot p(\vec{y}|\mathcal{C}_1)}. \tag{4.8}$$

This equation can be used with the class-conditional density estimates $p(\vec{y}|\mathcal{C}_i)$ such as those given by the Gaussian classifier, the Gaussian mixture model, and the k nearest neighbor algorithm.

Single Gaussian Distribution

One method for modelling numerical features is to assume they are distributed according to a Gaussian density [5]. Training points for each class i can be used to estimate the parameters (mean $\vec{\mu_i}$ and covariance matrix $\mathbf{\Sigma_i}$) for the two class-conditional distributions. A likelihood ratio test can be used to decide what class the feature vector belongs to. This simple technique can only provide quadratic discriminant surfaces, which do not accurately reflect the division between the classes in general. The class-conditional distributions are given by the Gaussian distribution:

$$p(\vec{y}|\mathcal{C}_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\mathbf{\Sigma_i}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{y}-\vec{\mu_i})^\top \mathbf{\Sigma_i^{-1}} (\vec{y}-\vec{\mu_i})}. \tag{4.9}$$

Gaussian Mixture Model

The features can also be modelled according to a mixture of Gaussian distributions using the Gaussian mixture model (GMM) approach [5]. A GMM involves estimating the distributions for the classes based on a sum or mixture of Gaussian distributions. Once the mixture distributions are estimated for each class, the class-conditional densities can be compared using a likelihood ratio test to obtain a thresholding metric.

The GMM has greater flexibility in determining the discriminant surface; however, it requires significant computation to train. It also requires an initial estimate of the mixture distributions. Once the initial distributions are chosen, an Expectation-Maximization (EM) algorithm can be used to refine the distributions. The EM algorithm can be run for several iterations. Techniques such as cross-validation or smoothing must be used to prevent the model from overfitting the data. Despite

these complexities, the GMM allows a greater variation of discriminant surfaces than a simple Gaussian model.

4.3.4 Modeling Feature Distributions Non-parametrically

The feature vectors can also be modelled non-parametrically. Examples of such approaches include the k nearest neighbor algorithm and decision trees.

k Nearest Neighbor Algorithm

The k nearest neigbor algorithm involves estimating the class-conditional density using the k closest training vectors to the test vector $\vec{y}$ [3]. The algorithm finds the volume V_i of the smallest sphere with center $\vec{y}$ that encloses k_i training vectors from class C_i. The quantity V_i is then used to estimate the class-conditional density according to:

$$p(\vec{y}|C_i) = \frac{k_i}{n_i} \cdot \frac{1}{V_i}, \tag{4.10}$$

where an appropriate value for k_i is approximately $\sqrt{n_i}$. The class-conditional densities can then be used with Bayes' Rule in the likelihood ratio test (equation 4.8) to generate a score that can be compared to a threshold. Because of the variation in the feature vector components, they should be normalized beforehand by subtracting the mean and dividing by the standard deviation estimated from all the training data.

Decision Trees

Decision tree algorithms involve partitioning the feature space to build separate models for different regions. More explicitly, the decision tree recursively splits the feature space into two separate regions according to a specific characteristic of the data. Each of the new regions can use an independent classifier to model the data.

Although decision trees require a substantial amount of data to train, they allow for a better model if one or more data features substantially affect the discriminant surface. In other words, a decision tree models the dependencies within the feature

vector better than other techniques.

4.3.5 Evaluating the Thresholding Metrics

The performance of a thresholding metric is difficult to assess. Experimentally, the thresholding metric can be evaluated by considering the errors the classifier makes at different values for the threshold. Because the threshold can vary, we must specify the threshold at which the classifier should be evaluated. Unfortunately, this depends substantially on the clustering.

Each of the thresholding algorithms given above seem theoretically reasonable. Therefore, the best thresholding algorithm is judged based on its performance when used as part of a clustering system. Moreover, we measure performance based on the official evaluation metric, because it is the quantity we are trying to optimize for. Because of the computational complexity of implementing the GMM, we eliminated it from our analysis.

Each thresholding metric was trained on the same data set. The system performance was measured at the optimal threshold, which was estimated by minimizing the official evaluation cost metric (because lower evaluation scores are better). The results of using each classifier with the incremental k-means clustering algorithm are shown in table 4.2. The first score is a cost function where mistakes are weighted by story; the second is weighted by topic and is used for the official evaluation. Section 5.2.3 offers more detail on the evaulation metric..

First, we analyze the single-score metrics. The cosine distance metric performs well in generating small homogeneous clusters. It fails on larger topics where the focus changes substantially over time, as indicated by the poor story-weighted score. The two normalized topic spotting (Tspot) metrics work much better on large topics, probably because large topics provide more data to accurately estimate the cluster model. Unfortunately, due to data sparsity, the topic spotting metrics perform rather poorly on smaller clusters, although the length-normalized metric performs quite a bit worse than the mean- and variance-normalized metric.

In light of these observations, we can construct a feature vector classifier that

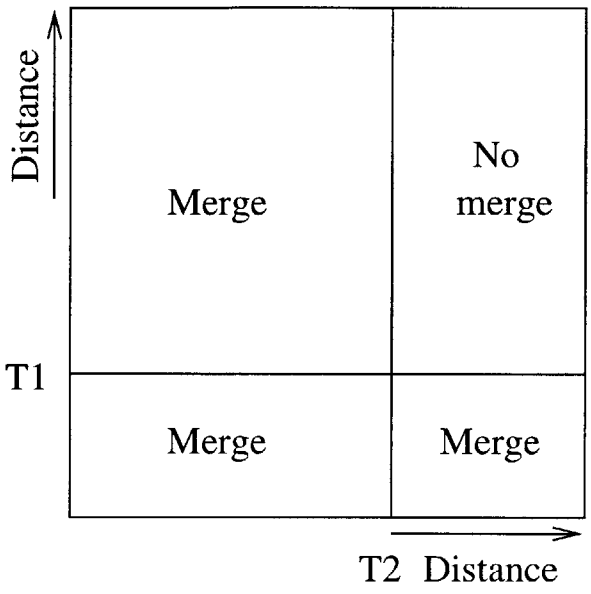


Figure 4-1: Simplified decision tree involving dividing the score feature space into four quadrants (T1 and T2 are thresholds for the scores)

performs well on both small and large clusters. One such classifier is the k-nearest neighbor algorithm. The k-nearest neighbor algorithm tends to produce very low rates of misclassification when cross-validated with data. Unfortunately, the performance gains on the evaluation metric are not as substantial.

Finally, decision trees can be used for the binary classification problem. We use a simple decision tree based on the score feature vector which finds single thresholds for each score and makes a hard decision for each quadrant (see figure 4-1). In other words, the final decision is a simple combination of the two single-dimension decisions. In the table 4.2, DTree 1 is a classifier based on the cosine distance and length-normalized topic-spotting metric; DTree 2 is based on the cosine distance and mean- and variance-normalized topic-spotting metric. The latter works very well, and this is what the final evaluation system used.

A more satisfying decision tree approach would be to automatically train a decision tree based on a larger feature vector. Unfortunately, due to time and complexity restrictions, we did not utilize this approach in our system.

System	Story-weighted C_D	**Topic-weighted C_D**
Cosine dist	0.0080	**0.0025**
Length-normed Tspot	0.0047	**0.0031**
Mean/sd-normed Tspot	0.0027	**0.0014**
KNN (Cosine+Tspot)	0.0047	**0.0026**
DTree 1	0.0027	**0.0022**
DTree 2	0.0025	**0.0013**

Table 4.2: Official evaluation results using different thresholding metrics on TDT-2 Mar-Apr CCAP+NWT data

4.4 News Sources and Score Biases

An important consideration when dealing with different sources is the proper normalization for each source. For example, ASR sources tend to make consistent errors especially on out-of-vocabulary (OOV) words. Therefore, the lower scores of comparing ASR sources to newswire stories should be considered when making decisions. Likewise, newswire sources tend to be very accurate but also contain more information than a newscast, affecting the scores.

A system can consider the source when making decisions about what the threshold should be in a particular setting. For example, we could add a bias to the threshold for closed-captioned (CCAP) data, because the error rate is higher than newswire data. The experimental results of clustering with added biases to the audio source thresholds are shown in table 4.3. Although the scores improve slightly with this technique, the biases do not always generalize to other data sets, and the performance improvement is relatively small.

4.5 Chapter Summary

There are many possible methods for combining similarity metrics to make appropriate selection and thresholding metrics for clustering. We advocate a simple selection metric based on the BBN topic spotting metric for finding a relevant custer to a given story. There are many methods that can be used for thresholding: single-

CCAP+NWT results		
System	Story-weighted C_D	**Topic-weighted C_D**
Unbiased	0.0027	**0.0013**
Biased	0.0024	**0.0012**
ASR+NWT results		
System	Story-weighted C_D	**Topic-weighted C_D**
Unbiased	0.0028	**0.0022**
Biased	0.0026	**0.0022**

Table 4.3: Official evaluation results generated by biasing thresholds for audio sources on TDT-2 Mar-Apr data

variable classifiers, parametric classifiers, and non-parametric classifiers. However, a simplified decision tree based on the cosine metric and mean- and variance-normalized topic spotting metric seems to perform well for the thresholding problem. Finally, the news source and other information can be considered when determining the metric threshold.

Chapter 5

Evaluation and Results

Thus far, we have described a system for the clustering of news stories into topics, but one important consideration in designing the system is evaluating how well it performs. This chapter describes methods for evaluating topic groupings and compares our experimental results to those from other research systems.

First, we describe the corpora that are used for experimentation in TDT. Then we discuss different evaluation metrics and their respective strengths and weaknesses. Finally, we give our results as evidence to the effectiveness of our topic detection system.

5.1 Corpora

The Linguistic Data Consortium (LDC) has released two corpora for the purpose of expanding research in TDT. The first corpus (TDT-1), originally used for a pilot evaluation conducted in 1996-1997, consists of 15,683 stories from newswire sources, collected over the course of one year [2].

The second corpus, referred to as the TDT-2 corpus, consists of about 60,000 stories collected over a six-month period from both newswire and audio sources [13]. The TDT-2 corpus is subdivided into three two-month sets: a training set (Jan-Feb), a development test set (Mar-Apr), and an evaluation set (May-Jun). Because a detection system is not trained, there is little functional difference between the

training set and development test set. Both sets can be used freely in the research and system design, but the evaluation set is withheld until the systems are evaluated.

The data is annotated at LDC by human annotators who listen to the audio data or view the text transcripts. The annotators are given a set of predefined topics to look for. For each story, an annotator determines which of the topics are relevant to the story. A judgement of "YES" indicates that over 10% of the story is relevant to the topic. A judgement of "BRIEF" indicates that less than 10% of the story is on topic. If the story is not on topic, it is judged "NO" [16]. The annotations are checked for consistency, and ambiguous judgements are arbitrated. After undergoing this procedure, most stories are not labeled, and some stories are labeled for multiple topics. Only about 3-20% of the stories are labeled into 30-40 topics per data set.

The data is divided into segments called files. Each file contains the equivalent of a half-hour newscast or about 50-100 newswire stories. The allowable look-ahead is expressed in terms of files: the system can look either 1, 10, or 100 files into the future, including the current file.

5.1.1 Data Sources

The TDT-2 corpus data comes from a number of sources [13]. First, the newswire (NWT) stories come from the New York Times Service and the Associated Press Worldstream Service. Only stories from the New York Times newspaper are taken from the New York Times Service, and only English stories are used from the Associated Press. The newswires contain some stories that are summaries or previews of a wide variety of stories — these are labeled as "miscellaneous text" and are processed but not scored. Also, newswire stories sometimes repeat exactly, or with a very slight change in content.

A number of audio sources are used in the TDT-2 corpus [13]:

- **Cable News Network (CNN) Headline News:** About four television broadcasts are recorded every day. The closed-captioning is reasonably accurate, though deletion-type errors are common (where words, phrases, sentences,

and even stories are not transcribed) and misspellings occur several times in a broadcast. Stories that are not transcribed are labeled "untranscribed text", though are still considered in the scoring.

- **ABC World News Tonight:** This television program is broadcast once per day. The closed-captioning is significantly better than the CNN transcripts, though deletion errors are still common. An auxilliary set of higher-quality transcripts is also created by the Federal Documents Clearing House (FDCH).

- **Public Radio International "The World":** Broadcast once each day five days a week, this one-hour radio program features both American and non-American English speakers. Therefore, the ASR text is more prone to errors. Closed-captioning transcripts are acquired through an LDC contractor and are between standard closed-captioning and FDCH transcripts in quality.

- **Voice of America:** The Voice of America produces two radio programs used in the TDT-2 corpus: *VOA Today*, which is broadcast for an hour every day, and *World Report*, which is broadcast for an hour five days a week. Manual transcripts are provided by the Voice of America.

5.1.2 Discussion about the Data and Annotations

The human annotators missed about 6% of stories that were really on topic in the TDT-2 Mar-Apr set. Although the annotations are fairly accurate after they are rechecked, it should be noted that definitions of topics can become very specific over time. For example, a topic about the national tobacco settlement has very specific rules associated with it regarding what court cases are included and excluded. Therefore, we note that specific rules about what should and should not be included are impossible for an automatic system to learn.

We now return to the assumption made in Chapter 1 that multiple-topic stories belong to only one cluster. A judgement of "YES" indicates that as little as 10% of a story need be on the topic. This means that 90% of a story could potentially be on

some other topic. For example, a story talking about President Clinton's 1999 State of the Union adress might also refer to his impeachment trial. It is unreasonable to expect a system to be able to correctly cluster a story based on the 10% of a story that happened to be on the same topic as the one chosen by the annotators. The system is not required to correctly cluster stories marked "BRIEF" for a particular topic.

Furthermore, we note the presence of unannotated topics affects performance significantly. The evaluation acknowledges this problem by eliminating stories that are marked "YES" for more than one annotated topic. But it does not acknowledge the thousands of unmarked topics for which a story could also be considered a "YES". For example, a group of stories in the Jan-Feb data talked about a set of Thai auto worker riots. The system, although it clustered these together, did not cluster them with the labeled "Asian economic crsis" topic for which about half the riot stories were labeled. Either:

1. The annotations are correct, but the worker riot topic is an unlabeled topic that is confounding the system (figure 5-1), or

2. The annotations are wrong, and the Thai worker riots are really part of the Asian economic crisis (figure 5-2), a topic that happens to have an extremely large scope.

Because of these problems, it is unreasonable to expect a system to perform perfectly. The negative effect of unlabeled topics is shown by the experiment described in section 5.3.6.

5.2 Evaluation Metrics

This section describes a number of evaluation metrics that have been or could be used for the topic detecion task of TDT. First, goals are outlined for the evaluation metric. A few basic rules are described for preventing unreasonable demands on the system. Finally, the current and proposed evaluation metrics are described and discussed.

5.2.1 Goals

An evaluation metric is designed to measure the performance of a detection system by examining the clusters it generates. There are a number of important characteristics that an evaluation metric should possess.

First, as the metric needs to be accepted by the scientific community, the metric should represent a meaningful quantity. For example, a metric could reflect the cost for using the clusters in a particular application. If a metric minimally satisfies this criterion, then a well-performing system must be useful for some application.

Secondly, the metric needs to be computable regardless of how many stories in the corpus are labeled. Since most stories in the TDT corpus are not annotated, traditional clustering metrics such as cluster purity are not useful for measuring the system performance.

Finally, the metric should be comparable across different corpora, so that comparisons can be made not only between systems, but also across data sets. Note that though this is a desirable metric characteristic, it is less important for the purposes of comparing systems in TDT.

5.2.2 Basic Evaluation Rules

The evaluation metrics share a number of characteristics that are important to prevent several situations where it is not possible for the system to group a story correctly. First, non-stories (e.g., commercials, previews, etc.) are treated as stories by the detection system, but are ignored by the evaluation metric. Stories labeled as "BRIEF"

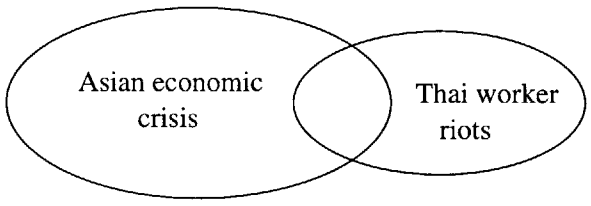


Figure 5-1: Venn diagram of a labeled topic (Asian economic crisis) and unlabeled topic (Thai worker riots) that overlap

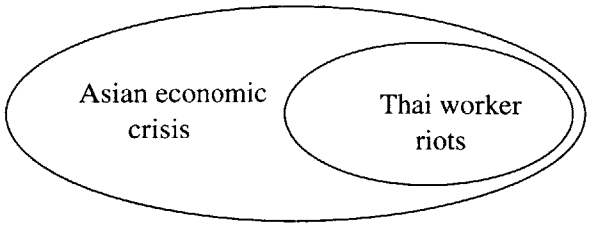


Figure 5-2: Venn diagram of a labeled topic (Asian economic crisis) and unlabeled topic (Thai worker riots) that have different scope

for a particular topic are not scored for that topic. Stories labeled as "YES" for two annotated topics are also ignored.

5.2.3 Currently Used Metrics

The evaluation metrics that have been used for topic detection measure one of two quantities:

1. *First story detection*, which is a system's ability to correctly find the first story on a new topic. This was used in the TDT Pilot Study.

2. *Cost function based on precision and recall*, which is an *ad hoc* quantity based on a system's performance that assumes the reference clusters can be mapped to the optimal corresponding system clusters. This is the basis for the TDT-2 evaluation metric.

First Story Detection

First story detection is a useful quantity if the main application of such a system is the prompt detection of new topics. In this case, a metric measuring a system's ability to detect the first story on a topic is an appropriate measure of performance. We can meaningfully define quantities such as the probability of miss P_M and the probability of false accept P_{FA}. Performance can be compared using some tradeoff between these two quantities.

Cost Function Based on Precision and Recall

The current official evaluation metric is a weighted cost function. Let R be the set of human-annotated topics and S be the set of system-generated clusters. Then, we map each cluster in R to a corresponding cluster in S by minimizing the quantity

$$C_D = P_M \cdot C_M \cdot P_T + P_{FA} \cdot C_{FA} \cdot (1 - P_T), \tag{5.1}$$

where P_M and C_M are the probability and cost of a miss, P_{FA} and C_{FA} are the probability and cost of a false accept, and P_T is the *a priori* probability of a topic. The quantities C_M and C_{FA} are fixed by the evaluation such that $C_M = C_{FA} = 1$. The probability of miss P_M is given by the number of stories in the reference cluster that are not present in the system cluster divided by the size of the reference cluster. The probability of false accept P_{FA} is given by the number of stories in the system cluster that are not present in the reference cluster divided by total number of stories that are not present in the reference cluster. More explicitly, if R_i is the set of stories in the reference topic that is mapped to the set S_j corresponding to a system cluster, then

$$P_M = \frac{|R_i - S_j|}{|R_i|}, P_{FA} = \frac{|S_j - R_i|}{|\bar{R}_i|}, \tag{5.2}$$

where $|\bullet|$ is the size of a set and $\bar{R}_i$ is the complement (i.e., all stories not present in R_i) of R_i. [16]

To get the final C_D, we average the detection cost for each cluster either over the topics (*topic-weighted* score) or the stories (*story-weighted* score). The topic-weighted score counts each topic's contribution to the total cost equally. Unfortunately, if a single story is missed in a relatively small topic, the final cost can be affected dramatically. The story-weighted score counts each story's contribution to the total cost equally. Although one story on a small topic is inconsequential in this case, large topics tend to dominate the score. The official evaluation is based on the topic-weighted score. [16]

Discussion

The first story detection metric seems reasonable, because it represents a legitimate application of the detection technology. It defines the quantities P_M and P_{FA} in a rigorous way and can compare the performance of systems for this application meaningfully. Unfortunately, there are only a few (30-40) topic starts that are annotated in the corpus. Therefore, there is not much data to evaluate the usefulness of the first-story detector. In the TDT Pilot Study evaluation, the data sparsity was overcome by creating several new corpora by incrementally eliminating the first story in each topic.

Although the cost function based on precision and recall is the official evaluation metric, it suffers from a number of important drawbacks. First, because of the mapping of reference clusters to system clusters, the evaluation of the cost function is analagous to the case where interesting topics are known in advance. It does not measure the system's ability to discover what are the important features of the data. In addition, there is no corresponding useful application that fits this metric.

Secondly, the cost function is based on mapping reference clusters to a small percentage of the system clusters (roughly 1-5%). This makes the measurement of the cost function very sensitive to variations in the labeled data. Because of this sensitivity, small differences between systems can be attributed to random variation.

5.2.4 Another Possibile Evaluation Metric

Because of the drawbacks of the currently available evaluation metrics, most notably the official measure, we consider an alternative that corrects some of the flaws.

BBN/YDZ Metric

One such metric is known as the BBN/YDZ (Yanguas, Doddington, and Zissman) metric, based on the speaker clustering evaluation metric [22]. It proposes a strategy for the application of the clustering system results based on the goal of finding all interesting stories in the corpus. Namely, the metric assigns a value to finding an

interesting story and a cost for looking at one story. The strategy is to look at a random element of a cluster. If it is interesting, the strategy dictates that the entire cluster should be examined; otherwise, the entire cluster should be discarded. The metric calculates the expected value of using this strategy to analyze the data. The metric is given by:

$$C_{BBN} = \sum_t \sum_c p_{t,c} \cdot (n_{t,c} V_H + n_c C_L) - (1 - p_{t,c}) \cdot C_L \tag{5.3}$$

where t is summed over all labeled topics, c is summed over all system clusters, $p_{t,c}$ and $n_{t,c}$ are the fraction and number of stories labeled t in cluster c, n_c is the number of stories in cluster c, C_L is the cost of a look, and V_H is value of a hit.

Discussion

While the BBN/YDZ metric measures the cost of using a particular strategy, it is not the only strategy for finding interesting stories that could be used. For example, the analyst could look at 10% of a cluster before deciding whether it is of interest. However, without a defined application, it is difficult to evaluate a system on this basis.

The BBN/YDZ metric has a number of desirable characteristics. It is derived from an application, so the result represents a real quantity, although the merits of any application can be debated. It can be computed for an arbitrary number of labeled clusters. It takes into account the system clusters that do not map to reference clusters. Unfortunately, the output of the metric is not comparable across different data sets.

5.3 Results Summary

We present results of our experiments to demonstrate the system performance. First, we provide a brief summary of the system used to generate these results. Then, we compare the performance of our system to that of other systems designed within a

similar constraint. Finally, we provide results showing the effects of changing the evaluation conditions.

5.3.1 System Description

We summarize the detection system used to generate the following results. First, the system preprocesses the stories by eliminating stop words, stemming, and creating term vectors. These vectors are used as inputs to both the topic spotting and cosine distance metrics. The topic spotting score is normalized by the mean and standard deviation of scores, which are estimated robustly by comparing each story to background clusters. These background clusters are generated by automatically clustering 4000 outside stories.

We utilize the topic spotting metric for cluster selection. The simplified decision tree based on the IDF-weighted cosine distance and the mean- and variance-normalized topic spotting score is used for thresholding. We cluster using the incremental k-means approach, though we use a look-ahead of only one file. The optimal decision tree thresholds are found by experimentation.

5.3.2 Performance Results

According to the December 1998 evaluation of topic detection systems, our system outperformed many of the others. The results of the default conditions of the evaluation are given in table 5.1. Henceforth, we use abbreviations for automatic speech-recognized data (ASR), closed-captioned data (CCAP), and newswire text data (NWT).

One concern with experiments conducted using the Jan-Feb and Mar-Apr data is the dependence of the decision tree thresholds on the corpus and human-chosen topics. We show in table 5.2 the dramatic difference between the thresholds chosen for the Jan-Feb data versus the Mar-Apr data. By slightly tuning the metric thresholds, we can improve the May-Jun set topic-weighted C_D by 0.0003 points. Because this improvement is relatively small, the decision tree thresholds were estimated fairly well

Story-weighted results			
Site	P_M	P_{FA}	C_D
BBN	**0.0884**	**0.0022**	**0.0039**
CIDR	0.3780	0.0018	0.0093
CMU	0.3575	0.0004	0.0076
Dragon	0.1563	0.0013	0.0044
IBM	0.11822	0.0008	0.0045
UIowa	0.5396	0.0009	0.0117
UMass	0.0831	0.0023	0.0039
UPenn	0.2919	0.0011	0.0069

Topic-weighted results			
Site	P_M	P_{FA}	C_D
BBN	**0.1220**	**0.0022**	**0.0047**
CIDR	0.3257	0.0018	0.0084
CMU	0.2586	0.0004	0.0057
Dragon	0.1736	0.0013	0.0048
IBM	0.1629	0.0008	0.0042
UIowa	0.4214	0.0009	0.0095
UMass	0.2088	0.0023	0.0064
UPenn	0.2627	0.0011	0.0063

Table 5.1: Official TDT-2 evaluation results (based on May-Jun NWT+ASR data with a 10 file look-ahead period). The official metric is the topic-weighted C_D (lower is better).

Topic-weighted results			
Data set	Cos thresh	TSpot thresh	C_D
Jan-Feb CCAP+NWT	-.95	-9.5	.0056
Mar-Apr CCAP+NWT	-1.0	-8.0	.0013
Mar-Apr ASR+NWT	-.85	-7.0	.0020
May-Jun ASR+NWT	-.95	-7.5	.0042

Table 5.2: Optimal clustering thresholds for different data sets

for the evaluation.

We can also substantiate the performance of our system by examining its performance on the other metrics mentioned above. Unfortunately, because no other sites report results for the other metrics, it is difficult to compare the performance across systems. We suggest that although the current metric has drawbacks, it generally reflects the relative performance of systems. The next TDT evaluation is slated to incorporate the YDZ/BBN metric in some form; that will reveal the performance of different systems when tuned to new metric.

5.3.3 Effect of Using Different Data Sets

Unfortunately, we find substantial differences between the different data sets that have been produced for TDT-2. Curiously, the Jan-Feb data has a few topics that are very broad and a few that are very focused. This inconsistency is reflected in the system's performance. The Mar-Apr data contains roughly 1/8 the number of labeled stories than the Jan-Feb data. Therefore, the Mar-Apr set contains smaller topics that are generally more consistent. Finally, the May-Jun set contains roughly 3 times the number of labeled stories as Mar-Apr. The May-Jun data set again has more variation, with several smaller topics and many larger topics. The scores are shown in table 5.3.

These results seem to suggest a correlation between the number of annotated stories and the cost function. The more stories that are labeled, the worse the system performs on the official evaluation metric. This effect is shown in table 5.4. The degra-

Story-weighted results			
Data set	P_M	P_{FA}	C_D
CCAP+NWT Jan-Feb	0.3498	0.0021	0.0090
ASR+NWT Mar-Apr	0.1083	0.0004	0.0026
CCAP+NWT Mar-Apr	0.1128	0.0004	0.0027
ASR+NWT May-Jun	0.0930	0.0022	0.0040
CCAP+NWT May-Jun	0.0582	0.0023	0.0035
Topic-weighted results			
Data set	P_M	P_{FA}	C_D
CCAP+NWT Jan-Feb	0.1763	0.0021	0.0056
ASR+NWT Mar-Apr	0.0813	0.0004	0.0020
CCAP+NWT Mar-Apr	0.0435	0.0004	0.0013
ASR+NWT May-Jun	0.1292	0.0022	0.0047
CCAP+NWT May-Jun	0.1044	0.0023	0.0044

Table 5.3: Official evaluation metric using the same algorithm on different data sets (1 file look-ahead)

Data set	No. of labeled S	Ave topic size	Story-wtd C_D	Topic-wtd C_D
Jan-Feb	3613	103.2	.0090	.0056
Mar-Apr	576	23.0	.0027	.0013
May-Jun	1312	38.6	.0035	.0044

Table 5.4: Results showing the correlation of C_D with average topic size (using CCAP+NWT data)

dation in performance could be attributed to the lack of consistency in determining the human-annotated topics. The topics are determined separately for each data set by randomly sampling stories and heuristically determining the topic to which the sampled story belongs. Because the topics were determined months apart for each data set, the criteria used could end up being fundamentally different for each data set.

5.3.4 Effect of Manual Vs. Automatic Transcripts

The transcription method can have a significant effect on performance as well. ASR transcripts tend to have a very high error rate of about 23%, but the errors are rel-

Story-weighted results			
Data set	P_M	P_{FA}	C_D
ASR+NWT May-Jun	0.0930	0.0022	0.0040
CCAP+NWT May-Jun	0.0582	0.0023	0.0035
ASR+NWT Mar-Apr	0.1083	0.0004	0.0026
CCAP+NWT Mar-Apr	0.1128	0.0004	0.0027
Topic-weighted results			
Data set	P_M	P_{FA}	C_D
ASR+NWT May-Jun	0.1292	0.0022	0.0047
CCAP+NWT May-Jun	0.1044	0.0023	0.0044
ASR+NWT Mar-Apr	0.0813	0.0004	0.0020
CCAP+NWT Mar-Apr	0.0435	0.0004	0.0013

Table 5.5: Official evaluation metric comparison of ASR+NWT and CCAP+NWT data (1 file look-ahead)

atively consistent. CCAP transcripts have a smaller error rate, but the errors are usually typographical errors and are often inconsistent. Even so, the combination of the newswire stories (NWT) with the CCAP data produces significantly better clusters than using newswire stories and ASR transcripts. These variations are illustrated in table 5.5.

Interestingly, in the tracking task, there is very little degradation from using the ASR text versus CCAP text. This can be attributed to the training data that tracking systems are allowed combined with the consistency of the ASR errors. For example, a story that talks about "Iraq" might contain many consistent references to "a rock", because the two words are essentially homonyms. A detection system might split such a cluster into stories about Iraq and stories about rocks.

5.3.5 Look-Ahead Periods

The effect of increasing the look-ahead period using the incremental k-means clustering algorithm is not significant. Table 5.6 shows the improvement made by increasing the look-ahead period from 1 file to 10 files. We did not run experiments using a 100-file look-ahead period because this gain was insignificant, and the computation

Story-weighted results			
Look-ahead	P_M	P_{FA}	C_D
1 file	0.1007	0.0006	0.0026
10 files	0.1181	0.0002	0.0026

Story-weighted results			
Look-ahead	P_M	P_{FA}	C_D
1 file	0.0421	0.0006	0.0015
10 files	0.0598	0.0002	0.0014

Table 5.6: Official evaluation metric comparison of using different look-ahead periods on the Mar-Apr CCAP+NWT data

Topic-weighted results			
Data	P_M	P_{FA}	C_D
Full set	0.0435	0.0004	0.0013
Subset	0.0026	0.0003	0.0003

Table 5.7: Official evaluation metric results of using only the subset of human-annotated data (Mar-Apr CCAP+NWT data set)

required for looking ahead 100 files was too substantial.

5.3.6 Subset Experiment

To show the effect of multi-topic stories that contain non-annotated topics, we constructed a simple experiment. We created a data subset that contained only the stories in the Mar-Apr CCAP+NWT data set that were annotated "YES" for exactly one topic. We ran the same clustering algorithm described above on the subset data. The results, given in table 5.7, show that the subset performance is much better. Part of this gain can be attributed to the eliminated multiple-topic stories that confuse the system.

5.3.7 Named Entity Extraction

We used the BBN IdentiFinder, a program used to find names in broadcast news, to add names into the term vectors [4]. The experiment consisted of simply adding the

Story-weighted results			
Data set	P_M	P_{FA}	C_D
Baseline	0.1168	0.0004	0.0028
Including names	0.1320	0.0003	0.0029
Topic-weighted results			
Data set	P_M	P_{FA}	C_D
Baseline	0.0814	0.0004	0.0021
Including names	0.0873	0.0003	0.0021

Table 5.8: Effect of using names as part of the story term vectors (on Mar-Apr ASR+NWT data)

names as separate terms to the story term vectors. The results of this experiment are show in table 5.8. These results seem to indicate that the use of names does not substantially improve the performance of our topic detection system.

5.4 Chapter Summary

This chapter contains the results of experiments run using the detection system. There are a number of alternatives to the current evaluation metric, but they are not currently useful because no other sites report results based on them. Finally, our results show that our system performed quite well compared to many others.

Chapter 6

Conclusions

In the preceding work, methods are developed for clustering news stories by topic. Several different causal clustering algorithms are considered for this purpose. Some similarity methods are discusussed, and a few are combined to form effective clustering metrics. The clustering metrics are divided into the cluster selection and thresholding problems, and different metrics are used for each. The results for a number of comparative experiments are given.

6.1 Accomplishments

Topic detection is a unique and valuable area, and it is increasingly important as the amount of textual information grows. The system developed herein performs very well despite the difficult and poorly-defined problem. Furthermore, this system contains several novel features.

While other systems use one single metric for clustering, our approach divides the problem into selection and thresholding. By using separate metrics for each of these problems, the system is able to exploit metrics that fit the problems posed by each. Because of this, performance can be improved substantially.

The use of the probabilistic BBN topic spotting metric is a new and effective technique for attacking the cluster selection problem. Because it uses a two-state model for the cluster and calculates the likelihood of the cluster formally, it performs

much better than the traditional vector-space models.

For cluster thresholding, the problem can be formulated similarly to the basic IR problem. Using a binary classifier to determine the topicality of a story to a cluster is a new approach to the thresholding problem. It appears promising because the binary classifiers discussed can be formulated in a probabilistic way.

The incremental clustering algorithm lends itself well to clustering the stories in a causal way. It is computationally simple, and it produces very good results. We show more complicated clustering algorithms that can exploit the look-ahead granted by the TDT evaluation as well, although the contribution of this work to clustering is small.

New evaluation metrics seem to provide a more reasonable way of measuring the performance of a topic detection system. Although the BBN/YDZ metric has been used before for speaker clustering, its application to TDT is a new area.

Finally, our results show that the system performance is comparable or superior to many other similar systems.

6.2 Future Work

Although our topic detection system performs quite well, a number of research avenues are left unexplored:

- Preprocessing: We do not explore in depth the use of features that model higher-order dependencies. An interesting study might be to examine further the effects of using bigram or n-gram feature vectors. We also do not thoroughly investigate the use of finding proper names to augment the feature vectors, though we utilize some basic methods for incorporating that information. We also did not explore the use of a thesarus for grouping related terms together.

- Similarity metrics: We explore many types of similarity metrics, though there are many more that we do not attempt to use. Alternative similarity metrics might be more effective, especially those that exploit cluster characteristics.

- Metric combinations: Other selection metrics could be formulated, though the selection metric proposed is the most effective of those described here. More interestingly, one could propose a wide range of classifiers for thresholding, especially those that use different information. For example, the feature vectors for the binary classification problem could incorporate story age, news source, story and cluster size, etc.

- Clustering: We use a simple clustering algorithm to cluster the stories. However, one could use much more elaborate techniques to take advantage of the look-ahead. One interesting possiblity might be to treat the clustering and metric combinations jointly (i.e., use a "fuzzy" clustering algorithm). The individual stories could then be assigned probabilistically to clusters.

- Evaluation metrics: Although the current evaluation metric is one method for evaluating a system, different metrics may produce different results. Therefore, one could examine the use of different evaluation metrics to find an application-independent clustering algorithm. Moreover, once an application is designed, optimizing the detection algorithm becomes more straightforward.

6.3 Final Thoughts

Topic detection is an important area of research with many intriguing applications. Although our system is optimized for only one particular metric, it could be easily and successfully modified for many different evaluation metrics. The probabilistic methods provide a nice framework for developing and extending the models we use. Finally, as this is a new and developing area, more research is needed to expand and develop this technology.

Bibliography

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proc. of the 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval*, Melbourne, Austrailia, August 1998.

[2] J. Allen, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic Detection & Tracking Pilot Study final report. In *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, February 1998.

[3] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*, 46(3):175–185, August 1992.

[4] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proc. of Fifth Conference on Applied Natural Language Processing*, pages 194–201, Washington, DC, March 1997.

[5] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1996.

[6] Douglass R. Cutting, David R. Karger, and Jan O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proc. of the 16th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, June 1993.

[7] B. Everitt. *Cluster Analysis.* Halsted Press, New York, second edition, 1980.

[8] L. Gillick, Y. Ito, L. Manganaro, M. Newman, F. Scattone, S. Wegmann, J. Yamron, and P. Zhan. Dragon Systems' automatic transcription of the new TDT cor-

pus. In *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, February 1998.

[9] Alan Griffiths, H. Claire Luckhurst, and Peter Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37(1):3–11, 1986.

[10] Hubert Jin. Topic tracking and detection for radio, TV, broadcast, and newswire. Submitted to the ACM-SIGIR conference, 1999.

[11] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An introduction to cluster analysis.* John Wiley & Sons, New York, 1990.

[12] T. Leek, D. Miller, and R. Schwartz. Labrador: A hidden Markov model information retrieval system. Submitted to the ACM-SIGIR conference, 1999.

[13] Linguistic Data Consortium. *Linguistic Data Consortium (LDC) TDT web site.* http://www.ldc.upenn.edu/TDT/.

[14] Kevin J. MacLeod and W. Robertson. A neural algorithm for document clustering. *Information Processing & Management*, 27(4):337–346, 1991.

[15] B. Mirkin. *Mathematical Classification and Clustering.* Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[16] National Institute of Standards and Technology (NIST). *The Topic Detection and Tracking Phase 2 (TDT2) Evaluation Plan Version 3.7*, September 1998.

[17] Kenney Ng. *Survey of Approaches to Information Retrieval of Speech Messages (DRAFT).* MIT Labratory for Computer Science, February 1996.

[18] Kenney Ng and Victor Zue. Phonetic recognition for spoken document retrieval. In *Proc. of ICASSP*, Seattle, WA, May 1998.

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

[20] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[21] R. Schwartz, T. Imai, L. Nguyen, and J. Makhoul. A maximum likelihood model for topic classification of broadcast news. In *Eurospeech Proc.*, Rhodes, Greece, September 1997.

[22] A. Solomonoff, A. Mielke, M. Schmidt, and H. Gish. Clustering speakers by their voices. In *Proc. of ICASSP*, Seattle, WA, 1998.

[23] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[24] F. Walls, H. Jin, S. Sista, and R. Schwartz. Probabilistic models for topic detection and tracking. In *Proc. of ICASSP*, Phoenix, AZ, March 1999.