

ON-LINE NEW EVENT DETECTION, CLUSTERING, AND TRACKING

A Dissertation Presented

by

RON PAPKA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1999

Department of Computer Science

© Copyright by Ron Papka 1999

All Rights Reserved

ON-LINE NEW EVENT DETECTION, CLUSTERING, AND TRACKING

A Dissertation Presented

by

RON PAPKA

Approved as to style and content by:

James Allan, Co-chair

W. Bruce Croft, Co-chair

Andrew G. Barto, Member

Kourosh Danai, Member

James F. Kurose, Department Chair
Department of Computer Science

For Claudia, Sophia, and Ursula.

ACKNOWLEDGMENTS

Behind every Ph.D. candidate are forces that push one towards and pull one away from the completion of the degree. I am grateful to the sources of both types of force, especially my parents, Benjamin and Yonina Papka, my wife Claudia, my daughter Sophia, and my dog Ursula. I am also grateful to James Allan, Bruce Croft, and Jamie Callan, who have advised me throughout my studies at the University of Massachusetts. They have introduced me to the exciting field of Information Retrieval, and provided me with an environment extremely rich in resources and research opportunities. I would also like to thank Kate Moruzzi and Sharon Mallory for making my experience at the Department of Computer Science a pleasant one that was virtually free from the bureaucracy and paperwork inherent at large institutions.

In addition, I thank Charles Wayne, George Doddington, Yiming Yang, Jaime Carbonell, Jon Yamron, and Richard Schwartz, with whom I have worked on the Topic Detection and Tracking project. I am also grateful to several of the TDT participants including Victor Lavrenko, Hubert Jin, Fred Walls, Tom Pierce, Paul van Mulbregt, and Mike Schultz for helping maintain a collaborative research environment for TDT in which ideas and implementation issues were discussed freely, without interference from the competitive element of the project.

The TDT initiative is a DARPA-sponsored project that supported this work. This material is also based on work supported in part by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor.

ABSTRACT

ON-LINE NEW EVENT DETECTION, CLUSTERING, AND TRACKING

SEPTEMBER 1999

RON PAPKA

B.Sc., COLUMBIA UNIVERSITY

M.Sc., BROWN UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor James Allan and Professor W. Bruce Croft

In this work, we discuss and evaluate solutions to text classification problems associated with the events that are reported in on-line sources of news. We present solutions to three related classification problems: *new event detection*, *event clustering*, and *event tracking*.

The primary focus of this thesis is *new event detection*, where the goal is to identify news stories that have not previously been reported, in a stream of broadcast news comprising radio, television, and newswire. We present an algorithm for new event detection, and analyze the effects of incorporating domain properties into the classification algorithm. We explore a solution that models the temporal relationship between news stories, and investigate the use of proper noun phrase extraction to capture the *who*, *what*, *when*, and *where* contained in news. Our results for new

event detection suggest that previous approaches to document clustering provide a good basis for an approach to new event detection, and that further improvements to classification accuracy are obtained when the domain properties of broadcast news are modeled.

New event detection is related to the problem of *event clustering*, where the goal is to group stories that discuss the same event. We investigate on-line clustering as an approach to new event detection, and re-evaluate existing cluster comparison strategies previously used for document retrieval. Our results suggest that these strategies produce different groupings of events, and that the on-line single-link strategy extended with a model for domain properties is faster and more effective than other approaches.

In this dissertation, we explore several text representation issues in the context of *event tracking*, where a classifier for an event is formulated from one or more sample stories. The classifier is used to monitor the subsequent news stream for documents related to the event. We discuss different approaches to classifier formulation, and compare feature selection and weight-learning steps as extensions to a baseline process used for new event detection. In addition, we evaluate an unsupervised adaptive approach to event tracking that captures the property of *event evolution* in broadcast news.

The implementations of our approaches to on-line new event detection, clustering, and tracking have been evaluated in comparison to other systems, and we present cross-system comparisons for all three classification problems. In general, the results using our approaches compared favorably to other approaches for each problem.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGMENTS	v
ABSTRACT	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiv
 Chapter	
1. INTRODUCTION	1
1.1 What is an Event?	2
1.2 Applications for Event Detection	6
1.3 Research Contributions	8
1.3.1 New Event Detection	9
1.3.2 Clustering	9
1.3.3 Tracking	10
1.4 Dissertation Overview	11
2. RELATED WORK	13
2.1 Event-Based Text Classification	14
2.2 Representations for Supervised Document Classification	16
2.2.1 Categorization, Routing, Filtering, and Tracking	17
2.2.2 Ranked Retrieval Approaches to Document Classification	18
2.3 Clustering Algorithms	21
2.3.1 Agglomerative Hierarchical Clustering	24
2.3.2 Single-Pass Clustering	25
2.3.3 Cluster Comparison Strategies	26

2.4	Conclusion	29
3.	EXPERIMENTAL DATA AND EVALUATION METHODOLOGIES	31
3.1	TDT Corpora	31
3.2	Evaluation	33
3.2.1	Effectiveness Measures	34
3.2.2	Experiment Methodologies	36
3.2.3	Multiple-Pass and DET Curve Evaluation Methodology	40
4.	TRACKING	42
4.1	Implementation	45
4.1.1	Inverted Index File	46
4.1.2	Text Representation	47
4.1.2.1	Classifier Formulation	48
4.1.2.2	Document Representation	50
4.1.3	Comparing Classifiers to Documents	51
4.2	Optimization	52
4.2.1	Parameter Estimation for Tracking	52
4.2.2	Adaptive Query Formulation	53
4.2.3	Varying IDF Sources	54
4.2.4	Decision Score Normalization	56
4.3	Tracking Experiments	59
4.3.1	Static vs. Adaptive Tracking	60
4.3.2	Multiword Features and Weight-Learning	61
4.4	Cross-system Comparison of TDT2 Tracking Systems	64
4.5	Conclusion	67
5.	THRESHOLD ESTIMATION BIAS IN TEXT CLASSIFIERS . .	69
5.1	Threshold Estimation	71
5.2	Learning Threshold Estimator Bias	74
5.2.1	The Histogram Method	75
5.2.2	Linear Regression Method	75
5.2.3	Comparison of Methods	78

5.3	Bias in Relevance Assessments	79
5.4	Conclusion	81
6.	CLUSTERING	82
6.1	On-line Clustering Algorithms	83
6.2	Implementation	84
6.2.1	Time-based Threshold Model	84
6.2.2	Decision Scores	86
6.3	On-line Cluster Comparison Strategies	87
6.4	On-line Clustering Experiments	89
6.5	TDT Detection Experiments	93
6.6	Cross-System Comparison for TDT2 Detection Problem	96
6.7	Conclusion	100
7.	NEW EVENT DETECTION	102
7.1	New Event Detection Algorithm	104
7.2	Implementation	105
7.3	New Event Detection Experiments	106
7.3.1	Parameter Estimation Experiments	107
7.3.2	Clustering Approaches to New Event Detection	110
7.3.2.1	Retrospective Experiments	111
7.3.2.2	Predictive Experiments	114
7.3.3	Impact of ASR technology	117
7.3.4	Can Natural Language Phrases Increase Effectiveness?	120
7.4	TDT Pilot Study	125
7.4.1	CMU and Dragon System Approaches	125
7.4.2	Cross-System Comparisons	126
7.5	Discussion of Approach to New Event Detection	128
7.5.1	The Good News	128
7.5.2	The Bad News	129
8.	CONCLUSION	131
8.1	Summary of Research Contributions	131
8.2	Discussion of Experimental Results	132

9. FUTURE WORK	138
APPENDIX: APPENDIX A	141
1 Events in TDT1 Corpus	141
2 Events in TDT2-Train Corpus	142
3 Events in TDT2-Development Corpus	144
4 Events in TDT2-Evaluation Corpus	146
BIBLIOGRAPHY	148

LIST OF TABLES

Table	Page
3.1 TDT Corpora.	32
3.2 TDT Events.	33
3.3 Sign Test for Significant Improvements	40
4.1 Static tracking threshold parameter θ	53
4.2 Threshold parameter θ for adaptive tracking.	54
4.3 Story-weighted cost for static and adaptive tracking.	60
4.4 TDT2 cost for extensions to static tracking. (ASR+NWT, Nt=4). . .	63
4.5 NIST evaluation of TDT2 Tracking Systems (Nt=4).	66
5.1 Percent of cost increase from replacing estimator \hat{u} with estimator \hat{e} . . .	78
6.1 Detection results on TDT2-Evaluation corpus.	92
6.2 TDT Detection: TDT2-Evaluation corpus.	96
6.3 NIST evaluation of TDT2 detection task: TDT2-Evaluation corpus (ASR+NWT).	99
6.4 NIST evaluation of TDT2 detection task: TDT2-Evaluation corpus (CCAP+NWT).	99
7.1 New Event Detection: TDT1 corpus using 50-feature queries.	107
7.2 New Event Detection: TDT2-Train corpus using 50-feature queries. . .	108

7.3	New Event Detection: TDT2-Development corpus using 50-feature queries.	108
7.4	New Event Detection: TDT2-Evaluation corpus using 50-feature queries.	109
7.5	Cross-Corpora Comparison of New Event Detection	109
7.6	Comparison of clustering strategies for New Event Detection (Pooled averages, TDT2-Evaluation corpus, ASR+NWT).	116
7.7	New Event Detection using good parameters for Event Clustering (Pooled averages, TDT2-Evaluation corpus, ASR+NWT).	116
7.8	Percent improvement using New Event Detection threshold parameters.	117
7.9	Comparison of clustering methodologies for New Event Detection (Pooled averages, TDT2-Evaluation corpus with Closed Caption source).	119
7.10	Percent improvement using CCAP vs. ASR transcriptions.	119
7.11	Effectiveness measures for systems presented at the first TDT workshop.	126

LIST OF FIGURES

Figure	Page
1.1 Event Evolution: Oklahoma City Bombing	5
1.2 On-line Event Detection and Tracking System.	7
2.1 TREC Information Specification	19
2.2 Hierarchical Agglomerative Clustering Algorithm	24
2.3 Single-Pass Clustering Algorithm	25
2.4 On-line Cluster Comparison Strategies.	28
3.1 Holdout Evaluation Methodology.	36
3.2 k -fold Cross Validation Evaluation Methodology.	37
3.3 Bootstrapping Evaluation Methodology	38
3.4 New Event Detection: Expected error rates from bootstrapping. . . .	39
4.1 Inverted Index File Organization.	47
4.2 Classifier and Document Representation.	49
4.3 Tracking: Varying sources for IDF.	55
4.4 Decision Score Normalizations.	58
4.5 DET Graph: Static vs. Adaptive Tracking (ASR+NWT, $Nt = 4$). . .	62
5.1 Threshold Problem: Crash of US Air Flight 427	72

5.2	Histograms of optimal threshold parameter θ for varying Nt (50 features).	76
5.3	Regression of optimal threshold parameter θ for varying Nt (50 features).	77
6.1	Daily document counts for two TDT events.	85
6.2	Event Clustering: TDT1 corpus.	90
6.3	Event Clustering: TDT2-Training corpus.	90
6.4	Event Clustering: TDT2-Development corpus.	91
6.5	TDT Detection: TDT2-Evaluation corpus (Story-weighted cost for ASR+NWT source).	94
6.6	TDT Detection: TDT2-Evaluation corpus (Story-weighted cost for CCAP+NWT source).	95
7.1	Effects of varying threshold parameters θ and β .	111
7.2	New Event Detection: TDT1 corpus	112
7.3	New Event Detection: TDT2-Train corpus	113
7.4	New Event Detection: TDT2-Development corpus	113
7.5	Comparison of clustering strategies for New Event Detection (TDT2-Evaluation corpus, ASR+NWT).	115
7.6	Comparison of clustering methodologies for New Event Detection (Pooled averages, TDT2-Evaluation corpus, CCAP+NWT).	118
7.7	Proper Nouns Produced by Natural Language Parser	122
7.8	Impact of Proper Noun Extraction on Error Rates	124
7.9	DET graph for systems presented at the first TDT workshop.	127
7.10	General “US Air plane crash” classifier.	130

CHAPTER 1

INTRODUCTION

In this dissertation, we discuss and evaluate solutions to text-based document classification problems associated with events reported in on-line sources of news. We present solutions to three related classification problems which we refer to as *new event detection*, *event clustering*, and *event tracking*. These problems comprise the underlying tasks of an event detection system that classifies and organizes news stories for a user interested in monitoring world-wide events from several sources of on-line news. In addition to newswire sources, we apply our approaches to broadcast news, which includes data from television and radio transmissions.

Addressing the problem of *new event detection* is the main focus of this dissertation. The goal of the task is to identify stories that discuss new events which have not been previously reported. For example, a new event detection system should alert the user to the first story about a particular airplane crash, the inception of a political crisis, or the initial rumor of a corporate merger transaction. We view this problem as an instance of unsupervised binary classification where the system makes a yes/no decision about the novelty of each story without assistance from a user or access to labeled training instances. Should the user have a need to monitor a particular new event over time, an *event tracking* process can be initiated with the document discussing the new event. The goal of the tracking task is to retrieve subsequent news stories that pertain to the event of interest.

Our approach to the problem of new event detection is related to the problem of on-line *event clustering*, in which news documents are grouped together if they

discuss the same news event. The new event detection process finds the stories with which to seed a document clustering algorithm. We posit that single-pass clustering using existing cluster comparison strategies is a good baseline approach to new event detection. The work described here extends clustering approaches with methodologies for incorporating the properties of broadcast news into an existing framework. An analysis of our data suggests that our approaches improve classification accuracy for both new event detection and event clustering.

The primary motivation for this work arose from a project called Topic Detection and Tracking (TDT). This joint project, originally including DARPA, Carnegie Mellon University, Dragon Systems, and the University of Massachusetts, set out to explore issues related to detecting and tracking events in broadcast news. The results of the first year's efforts were reported at a workshop in October 1997 [84]. The second phase of TDT (TDT2) ended in March 1999 with an evaluation presented at the DARPA Broadcast News Workshop [59]. The research focus of TDT2 was event detection and tracking problems. New event detection has been re-introduced in the third phase of TDT, where it is defined as the problem of *first story detection*. New event detection appears to be of interest to a growing number of researchers, and results are expected from several new contributing groups. One of the purposes of this manuscript is to establish baseline evaluation measures with which to compare TDT classification tasks.

1.1 What is an Event?

Without a set of rules, different individuals will have different *a priori* notions of what constitutes an *event*. The definition of event used in the following work is based on the rules outlined by the TDT research initiative, and we use the term *event* to avoid overloading the notion of *topic* explored by related query-based news classification research using Text REtrieval Conference (TREC) data [34].

One of the first issues explored in the TDT pilot study was the meaning of *event*, which was defined as some unique thing that happens at some point in time [5]. The property of time is what distinguishes an event from the more general *topic*. For example, “computer virus detected at British Telecom, March 3, 1993,” is considered an event, whereas “computer virus outbreaks” is the general topic comprising occurrences of this type of event. The definition can be extended to include the spatial component of an event, namely location. For example, “The 1995 earthquake in Kobe, Japan” is a description of an event that uses this property.

A more recent taxonomy used in TDT2 has the following definitions [28]:

topic A seminal *event* or *activity* along with all directly related events and activities.

event Something that happens at some specific time and place. (Specific elections, accidents, crimes, and natural disasters are examples of events.)

activity A connected set of actions that have a common focus or purpose. (Specific campaigns, investigations, and disaster relief efforts are examples of activities.)

Other definitions for events, such as Popper’s [62], suggest that “an airplane crash” should be considered an event, while “the crash of US Air flight 427” should be considered an *occurrence*. Popper’s definitions are similar to those used by other scientists and philosophers who have reasoned that, in a metaphysical sense, events occur when there is a conflict between physical objects. Event-related philosophy concludes that two events are the same if they have the same spatio-temporal history, and that events are identical if they have the same causes and effects. Lombard [50] discusses why these properties are not sufficient conditions for *event identity*. He presents a model for events that includes the aspect of change, which he defines as “the addition or loss of properties.” His theory of change applies to events appearing in real-time news, because the content of documents pertaining to the same event changes over time as the event evolves.

The motivation for the approaches described here is to model event identity in broadcast news. A system that has the capability of modeling the properties of event identity and determines when two events are the same can be used to perform new event detection. For example, using an event identity process, the system could determine for the current document whether it contains an event identical to one appearing in any previously processed document. If so, the system would not detect a new event; otherwise it would. The event identity process could also be used to detect documents containing the same information, which is useful for eliminating duplicate news stories from different news sources.

From a journalist’s perspective, a news story about an event will typically specify 1) when the event occurred; 2) who was involved; 3) where it took place; 4) how it happened; and 5) the impact, significance, or consequence of the event for the intended audience [53]. This information is what the journalist’s readers or listeners are expecting. Some of these properties of news reporting are relatively simple to incorporate into a solution to new event detection. For example, the proper nouns that convey the *who*, *what*, *when*, and *where* of an event can be identified using natural language parsing and extraction techniques. Other properties, such as *cause*, *effect*, *significance*, and *impact*, are more difficult to identify and model.

As an event evolves, many of its properties are either not initially known, or are assumed to be known by the audience, and therefore they are not necessarily explicit in news text. For example, we found broadcast news coverage about the 1995 earthquake in Kobe, Japan, which referred to the event as “the worst disaster in Japan’s history,” with no explicit mention of Kobe or the fact that the story was about an earthquake.

In addition, the appearance of new lexical features for an event should be expected as the event evolves. Consider the temporal histogram of documents for the 1995 Oklahoma City bombing event depicted in Figure 1.1. After the blast, most of the

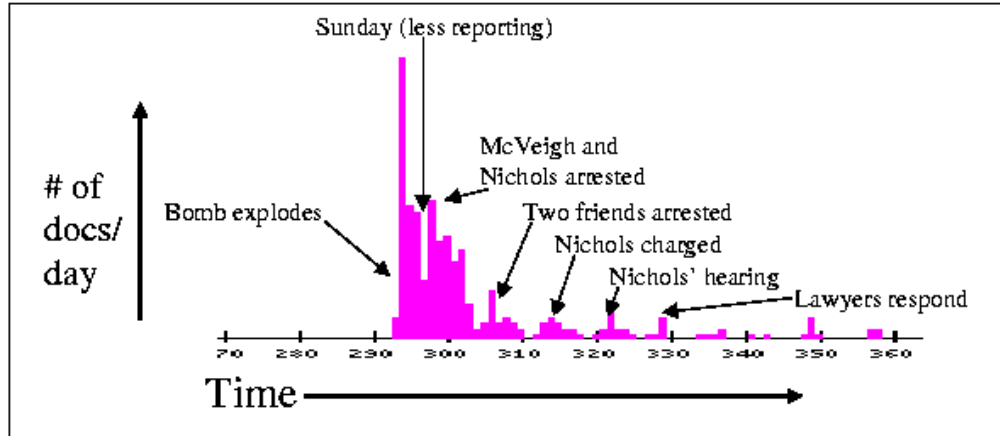


Figure 1.1. Event Evolution: Oklahoma City Bombing

coverage pertained to the rescue efforts at the disaster site. As time passed, the terrorists were apprehended, and the news coverage began to focus on their trials. The terrorists' names are distinguishing lexical features that are useful for classifying this event. However, they were not known when the event first occurred, and became available only as the event evolved.

In general, the words used to describe an event will differ among documents describing the same event. The inclusion and exclusion of lexical features suggests that simple word-matching approaches will need several algorithmic extensions to work well for event-based text classification. Our data indicate that some properties of

broadcast news, such as time, can be modeled inexpensively. Event evolution and proper nouns that include the *who*, *what*, *when*, and *where* features of news stories require relatively expensive processes, but we find these properties of news important to model. In the following chapters, we present experiments and results that discuss the effectiveness of modeling these properties for on-line news classification tasks.

1.2 Applications for Event Detection

News represents an information domain ideally suited for the study of new event detection, and we view new event detection as part of a larger process for organizing news. News is temporally ordered, discusses world-wide events in different languages, and contains video, audio, and text signals that could be combined into many useful applications. Our goal for new event detection in the on-line broadcast news domain is to determine when discussion of a new event first appears on a textual stream of news.

There are several practical applications that could arise from a good solution to new event detection. This task is currently performed manually by financial traders, media analysts, and on-line digital news editors who have the task of collecting, interpreting, and presenting news from multiple news sources. In addition, a system that could organize events automatically would be useful for financial and world news applications where decision-making processes are based on new events and the evolution of existing events.

The diagram in Figure 1.2 depicts the environment we envision for combining the event classification problems discussed in this work. Broadcast news signals from various sources are monitored by the system. When the source is radio or television, an Automatic Speech Recognition (ASR) process converts the audio signal to text [93], then a segmentation process finds document boundaries [61, 8], which delineate complete news stories. The detection system monitors the document stream and

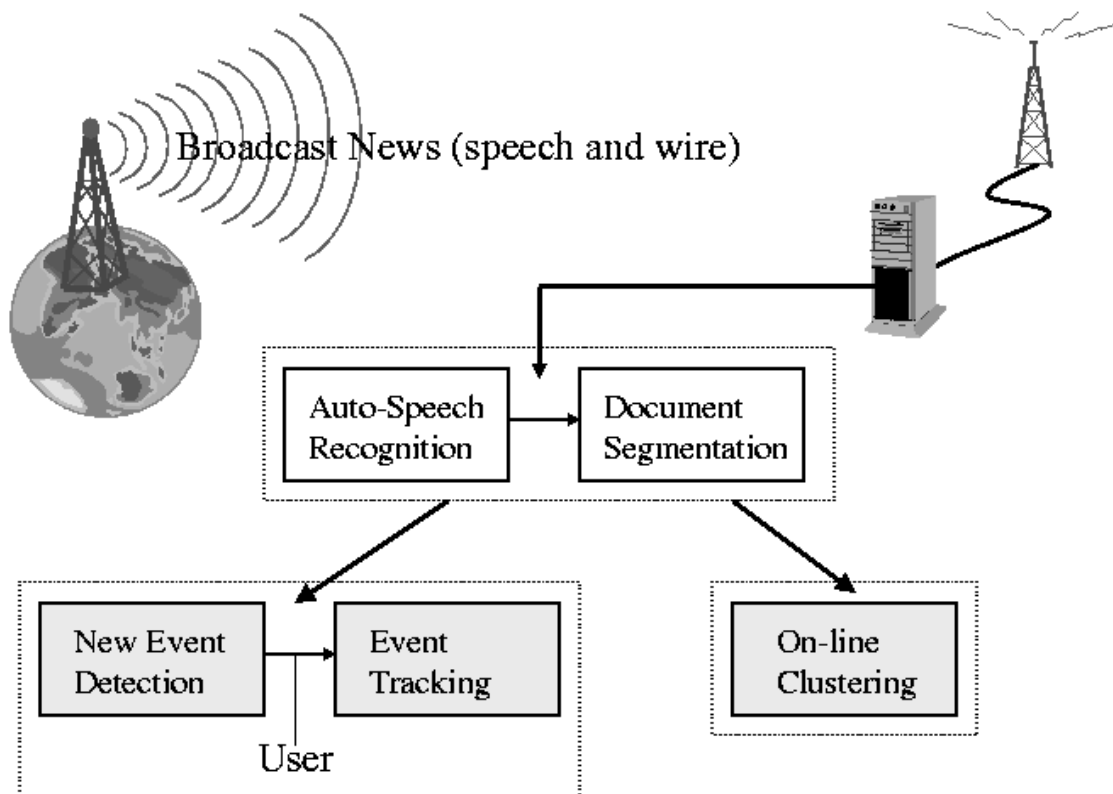


Figure 1.2. On-line Event Detection and Tracking System.

alerts the user to new events. If a document discussing a new event is of interest to the user, a tracking process is instantiated by creating a classifier from the contents of the document specified by the user. The classifier is then used to make on-line decisions about subsequent documents on the stream.

The new event detection system has two on-line modes of operation: *immediate* and *delayed*. In immediate mode, a strict real-time application is assumed, and the system indicates whether the current document contains or does not contain discussion of a new event before looking at the next document. In delayed mode, classification decisions are deferred for a prespecified time interval. For example, the

system could collect news throughout the day and provide the user with new events at the end of the day. In this thesis, we focus on solutions to immediate mode operation, and plan to study the effects of delay in future work.

Depending on the amount of user involvement, several extensions to the basic processes described here can be incorporated in a supervised process where the user provides relevance feedback on system classification decisions. In addition, some users may already know the event they want to track, and can provide some *natural language description* or *information specification* that can be used to instantiate the tracking process. Other users may want to create taxonomies or hierarchies of news by related events. Ideally, our system could do this automatically, and return an annotated organization of events using extensions of the clustering process. The combination of event classification problems in this context becomes an information organization task where the goal is to detect documents discussing new events and track documents related to the user’s needs.

In what follows, we implement and evaluate several of the core components of the system described above using the TDT task definitions, data, and evaluation methodologies, which are explained in subsequent chapters.

1.3 Research Contributions

This research describes effective techniques for new event detection, event clustering, and event tracking using extensions to the Inquiry retrieval system [12]. We study an Inquiry representation for text documents and on-line classifiers, and evaluate their behavior and effectiveness on these three related classification problems. In doing so, the research in this dissertation furthers the understanding of issues relating to document classification in general.

1.3.1 New Event Detection

The problem of new event detection has not been the focus of research prior to the TDT Pilot Study. The results contained in this manuscript are among the first to be reported for the problem, which is now referred to as “first story detection” in TDT.

Our main contributions to new event detection include:

- the first thorough investigation of the problem of new event detection;
- the introduction of a classification algorithm for new event detection;
- the introduction of extensions to previously studied text representations that explicitly model the properties of broadcast news;
- an analysis of on-line clustering approaches to new event detection that provides baseline effectiveness measures for previously used cluster comparison strategies.

We present evidence that suggests our single-pass algorithm is a good basis for new event detection. We find that modeling the properties of broadcast news as extensions to our approach results in improved effectiveness.

1.3.2 Clustering

In this dissertation, we test previous approaches to document clustering and evaluate their effectiveness for on-line event clustering. In general, clustering is a well studied problem in various fields. Most of the previous approaches for text are based on retrospective clustering, where all the data are available before clustering begins. Here, we apply retrospective approaches to an on-line environment. We re-evaluate single-link, average-link, and complete-link hierarchical agglomerative clustering strategies, but use them in a single-pass (incremental) clustering context, in which a cluster is determined for the current document before looking at the next document.

Our main contributions to the task of event clustering include:

- the introduction of a classifier-based approach to on-line clustering;
- the introduction of a threshold model that incorporates the temporal relationship between news stories; and
- a comprehensive analysis of single-pass clustering approaches for the domain of broadcast news.

Our results are based on evaluating event clustering as the TDT detection task, and they suggest that when using automatic transcriptions of broadcast news, an on-line single-pass single-link clustering strategy extended with a time component is more effective than other cluster comparison strategies.

1.3.3 Tracking

The representation we used for new event detection is derived from a classifier formulation process previously used for supervised text classification problems such as filtering and routing, where a user supplies a prespecified information request or query. We test previous approaches to classifier formulation through relevance feedback, but apply them to the TDT tracking problem. Much of the related work in this area is based on news story classification by topic. Here we evaluate previous approaches, and apply them to on-line event-based classification.

Our main contributions to tracking include:

- a further understanding of feature selection, feature weight assignment, and threshold estimation issues for on-line environments;
- the introduction of a theoretical framework for estimating classifier thresholds; and
- a comparative analysis of static and adaptive query formulation techniques.

A major focus of our tracking experiments is to evaluate the impact of relatively few training samples on classification accuracy. In addition, we compare supervised and unsupervised approaches to the TDT tracking problem. Our results suggest that it is possible to find stable parameters for our representation using statistical and learning techniques.

1.4 Dissertation Overview

The event classification problems are presented in the reverse order from which they appear in the title of this dissertation. Our work on event tracking is discussed first, followed by clustering, and then by new event detection. In the next chapter, we discuss the research that strongly influences our approaches to these three problems. The previous literature suggests that text representations applied to supervised and unsupervised text classification could be used for event-based classification as well. In Chapter 3 we describe the TDT corpora and evaluation methodologies we used to explore this hypothesis which is tested in subsequent chapters.

We investigate several text representation issues applicable to new event detection using the TDT tracking environment. We present our solutions and results for the tracking problem in Chapter 4, where the effectiveness of our approaches to classifier formulation is evaluated. In particular, we test the impact of multiword classifier features and supervised weight learning algorithms. We also test unsupervised adaptive approaches to the tracking problem, which capture the property of event evolution in broadcast news.

Classification accuracy is dependent on the effectiveness of parameter and threshold estimation techniques. In Chapter 5, we present a theoretical framework for threshold estimation. Two techniques that learn threshold estimator bias are discussed that improve tracking classification accuracy, and result in a further understanding of thresholding issues for other on-line text classification problems.

Our approach to new event detection is related to the problem of on-line clustering, and we explore this relationship in Chapter 6. We evaluate different cluster comparison strategies and their effectiveness at grouping stories that discuss the same event. In addition, we describe our method for including the temporal relationship between documents in the classification model, and evaluate our implementation in a cross-system comparison with other systems.

In Chapter 7, we present our algorithm for new event detection. The results for retrospective and predictive experiments are evaluated. One focus of these experiments is to evaluate the effectiveness of on-line clustering as the basis for a solution to new event detection. In addition, we describe a method for identifying proper noun phrases in news, and evaluate new event detection using this data. Our algorithm is also compared to two other systems evaluated as part of the TDT Pilot Study [84]. Our conclusions and plans for future work are discussed in Chapters 8 and 9.

CHAPTER 2

RELATED WORK

The main motivation of this dissertation is to test the hypothesis that representations for supervised and unsupervised text classification found in the field of Information Retrieval can be extended to a solution for new event detection. We explore methods for creating text classifiers in addition to approaches that extend the existing text representation with the domain properties of broadcast news.

The representation that we use for our solutions to event classification is borrowed from the document filtering and routing research, in which a text classifier is constructed for each topic of interest to the user. These problems assume a supervised classification environment, where classifiers are constructed from the lexical features of the sample documents provided by the user. Each classifier is formulated to retrieve documents that are similar in content to the relevant samples provided. We evaluate approaches to text classifier formulation and apply the same algorithm to the three event classification problems explored in this dissertation.

In this chapter, we discuss work related to the representation we use for new event detection, clustering, and tracking. Our approach to these problems is based on both supervised and unsupervised approaches to text classification. Event tracking is a supervised classification problem where the users of the system know *a priori* the events they are interested in tracking. Event clustering is an unsupervised problem where the users are interested in an automatic organization or grouping of documents by events, for all the events that may exist in the data. The salient distinction between these environments is that the supervised setting assumes that there exist

labeled training documents about each event of interest from which to formulate a classifier, while the unsupervised setting operates without training documents. In contrast, new event detection is an abstract classification problem in which the goal is to separate documents that discuss new events from documents discussing existing events, and the problem is unsupervised because no training documents are required for processing the data.

Our solution to new event detection is however related to the problem of on-line document clustering. In particular, one approach to new event detection is to cluster the document stream, and to return to the user the document in each cluster with the lowest timestamp. Assuming this approach is effective, the problem of new event detection becomes an issue of finding a clustering approach that works well in an on-line environment. In general, the previous approaches to clustering have been pursued using a retrospective environment where all the documents are available to the process before clustering begins. In this thesis we focus on an on-line solution to new event detection, and therefore we re-evaluate some of the prevalent approaches to retrospective clustering and analyze their effectiveness in an on-line environment.

The remainder of this chapter discusses in more detail the supervised and unsupervised classification problems that have been studied in Information Retrieval. In particular, we focus on the research that contributes to our solution to new event detection, which is described in Chapter 7.

2.1 Event-Based Text Classification

Much of the news classification research prior to Topic Detection and Tracking evaluates classification problems using *topics* and not *events*. For example, Hayes et al. [36] describe a news organization system in which a rule-based approach was used to categorize 500 news stories into 6 topics. The filtering problem analyzed at the Text Retrieval Conferences [38] is another example of topic-based news classifica-

tion. However, some event-based research has been reported prior to the first TDT workshop [84].

One representation applied to problems in news classification uses a data structure called frames [77]. The frames are constructed manually and coded for a semantic organization of text extracted by a natural language parser. Frames contain slots for structured text that can be organized at different semantic levels. For example, frames can be coded to understand entire stories [30], or for understanding the constituents of a person’s name [16].

A frame-based system that attempted to detect events on a newswire was constructed by DeJong in the late 1970s. He used pre-specified software objects called sketchy scripts [26]. Frames and scripts for general news events such as “Vehicular Accidents” and “Disasters” were constructed by hand. The goal of his system was to predict which frame needed to be populated. During processing, a frame was populated with words from the text, and a script was traversed to produce a short summary of the event. In one UPI newswire experiment, this system chose the correct script with a classification accuracy of 38% for documents relevant to one of fifty sketchy scripts. DeJong’s system was primarily a natural language parser, and as a side effect it detected when a document contained an event; however, it made no effort to detect new events.

More recent work by Carrick and Watters discussed an application that matched news stories to photo captions using a frame-based approach modeling proper nouns [16]. They concluded that using the extracted lexical features in a word matching approach was nearly as effective as using the same features in their frame-based approach. Current research related to frame-based representations on news data are discussed at Message Understanding Conferences (MUC) [55].

Frames are perhaps useful for representing different aspects of the natural language parse; however, the number of frames and the details of their contents would quickly

become difficult to maintain as new types of events emerge and existing events evolve in a news environment. We believe that a better approach to new event detection is to use unstructured word cooccurrences between documents, coupled with a model for properties of broadcast news.

2.2 Representations for Supervised Document Classification

Supervised document classification has been studied extensively in Information Retrieval in the context of categorization, routing, and filtering problems. The underlying problem is to formulate a classifier for a specific topic from a training set of labeled documents comprising relevant and non-relevant instances. The classifier is then used to retrieve related documents from a target corpus. What has emerged as the classifier of choice in this research is a vector of features comprising stemmed words and associated weights. The classifier’s words and weights are determined through statistical and learning techniques utilizing inter- and intra-document word occurrences in the training data.

The most common document representation is a vector of *tfidf* weights cooccurring with the words in the classifier, and our approach to new event detection is influenced by this representation. A *tfidf* feature weight comprises a term frequency component, *tf*, and an inverse document frequency component, *idf*. The *tf* component causes the weight to increase as a word’s occurrence in the document increases, and the *idf* component causes the weight to decrease as the number of documents in the collection in which the term occurs increases. The underlying model is that higher weight is given to features occurring frequently in the document and infrequently in the collection.

The *tfidf* weights are usually applied to stemmed words. For example, occurrences of the words “skiing” and “ski” may be conflated into one lexical feature by a stemming process, which has the tendency to decrease the space requirements for

a retrieval engine by decreasing the number of unique words in the lexicon. In practice, classification effectiveness tends to improve when words are stemmed and word senses are disambiguated. Porter [63] and Krovetz [44] have discussed methods for these processes. However, results from Harman [33] and Schultz [79] suggest that stemming does not necessarily improve effectiveness.

Many variations of *tf·idf* formulae have been presented in the literature, several of which were compiled and tested by Salton and Buckley [74] in the context of ranked-retrieval, which is described below. In addition, Zobel and Moffat [101] exhaustively tested several combinations of *tf·idf* variants. In general, no variant or combination of *tf* and *idf* has been shown to be significantly more effective than all others for document retrieval. It is therefore common for systems incorporating this representation to make use of heuristics and parameters that are determined empirically.

2.2.1 Categorization, Routing, Filtering, and Tracking

One of the earlier supervised classification problems studied in Information Retrieval was the task of document *categorization*. Lewis describes categorization as “the classification of documents with respect to a set of one or more pre-existing categories” [47]. In this environment, training documents or descriptions of pre-specified groups, i.e, categories, are provided. The goal of a categorization system is to determine to which of the existing categories each document in a corpus belongs.

Specializations within text categorization have emerged, including the problems of document routing, filtering, and tracking. Experimental results for several research and commercial systems that simulate routing and filtering are reported in the proceedings of in Text REtrieval Conferences (TREC) [34]. In TREC, the definitions of routing and filtering are tightly coupled with a specific evaluation methodology, but the underlying classification problem is the same for each task. Classifiers for each of several topics are constructed from a query specification, such as the one in Figure

2.1, and from a training set of labeled documents. TREC routing is evaluated as a ranked retrieval task, and thus it skirts automatic classification issues. Recently, the routing problem has been replaced by the *filtering* problem [38] which is evaluated using measures of effectiveness for hard classification decisions.

Over the years, variations of the routing and filtering problems have been tested that utilize different domains of data, and that vary the use of the query specification. Recently, TREC included the evaluation of *adaptive tracking*, which simulates an on-line classification environment. This task is like filtering, but includes an additional feedback step in which the system is provided the true label for documents that are classified with a positive on-topic decision. The feedback can be used to reformulate the classifier before making decisions about the next document in the testing corpus.

The research based on TREC data often uses thousands of relevant training samples for each topic. However, it is unlikely that a real user would provide this amount of relevance feedback. A more realistic environment assumes that the user will provide only a few documents. The effect of small amounts of user feedback is one of the focuses of the tracking problem for TDT [5, 4]. We discuss the tracking problem in more detail in Chapter 4.

2.2.2 Ranked Retrieval Approaches to Document Classification

Perhaps the most important artifact of Information Retrieval is text-based *ranked retrieval*. The problem is to assign a similarity score to each document relative to how well the document satisfies an information request. A list of documents, sorted by score, is returned to the user, and classification is performed manually. The ranking of documents is common practice for Internet search engines such as Infoseek, Lycos, Excite, and Yahoo, where the documents, in this case, are indexed web pages that the search engines maintain. Commercial and research IR systems such as Inquiry [12], Okapi [68], and SMART [72], use representations that are more complex than

Domain	Law and Government
Topic	Welfare Reform
Description	Document will report those proposed or enacted changes to U.S. federal, state, or local welfare laws and regulations which are propounded as reforms.
Narrative	A relevant document will reveal attempts by any U.S. jurisdiction to change the legal or regulatory context of existing welfare programs, or to add new programs, which are described or labeled as a reform. For the purposes of this topic, a welfare program must revolve around a transfer payment for which the recipient had not previously made contributions, other than through general tax payments. Examples include General Assistance, Aid for Families with Dependent Children (AFDC), public housing, Medicaid, income subsidies (rent and fuel subsidies, earned income tax credit, food stamps, etc.), child care programs (day care subsidies, nutrition programs, Head Start, etc.), and vocational training programs. NOT relevant are those programs which could be viewed as the payment of insurance benefits to those who had previously made contributions (either directly or indirectly through a family member or employer), such as Medicare, Social Security survivor's benefits, and unemployment insurance. Also NOT relevant are those documents which deal solely with the financing or administration of existing programs.
Concept(s)	welfare recipient, public assistance, family assistance, welfare benefit, relief, welfare reform, welfare dependency, reform movement, workfare, homeless, welfare culture, general assistance, aid for families with dependent children, AFDC, Medicaid, earned income tax credit, food stamps, Head Start, job training, public housing.
Factor(s)	Nationality: U.S.
Definition	Welfare program: although the above distinction between a "pure" transfer payment program and a putative insurance program is debatable, the distinction is made to focus the data search on those programs which are clearly in the welfare category, as opposed to the wide variety of social programs supported by U.S. government entities.

Figure 2.1. TREC Information Specification

those used for simple binary word matching searches. Turtle and Croft [87] discuss the limitations of binary word matching for ranked-retrieval, and they present results that suggest effectiveness improves from more extensive use of word-cooccurrence statistics such as those found in *tf-idf* feature weights described above.

Ranked retrieval naturally extends to automatic document classification. Given a ranked list of documents, the goal using this approach is to determine a threshold within the ranked list that maximizes some function of user utility or minimizes some function of cost. The threshold may be global or determined for each classifier. Supervised learning techniques for determining thresholds have been previously described in the literature [48, 56, 13, 100, 69]. When available, the documents that are known to be relevant or non-relevant to a topic can be used to build a better classifier through expansion, weight-assignment, and threshold selection techniques. Once obtained, a classifier can be used in an on-line setting to track related documents. It seems plausible that a better ranking system would lead to a better classification system; however, choosing an appropriate threshold in a ranked list of documents is in itself a difficult problem.

The hypothesis that learning and optimization can be used to improve ranked-retrieval effectiveness and classification accuracy has been suggested by many researchers. Bartell [7] adds significant evidence that suggests this hypothesis is true. Earlier contributions include a widely used query expansion technique introduced by Rocchio [70]. This method has been improved using machine learning approaches [48, 49, 56, 80, 78] and heuristic optimization techniques [10] on very large collections.

Most of the previous work for document classification has focused on supervised methods that use training documents with known relevance. Since relevance assessments are not always available, it is desirable to improve retrieval effectiveness without them. A paradigm has emerged in the literature, in which improvements in retrieval effectiveness are realized by *assuming* documents or subsets of documents are relevant

to a request. The assumed relevant documents are sampled from the initial results of a ranked retrieval process, and they are subsequently used to modify the classifier using supervised methods. Approaches have been reported that use the top n documents from an initial search to improve ad hoc ranked retrieval results [22, 68], where n is chosen using empirical techniques. Improvements over baseline methods have been reported by Xu and Croft using a query expansion technique called Local Context Analysis (LCA) [95]. Allan et al. [3] used an adaptive tracking technique similar to the one presented here, which reformulates classifiers with documents assumed to be related to the event being tracked.

2.3 Clustering Algorithms

Document clustering is an unsupervised process that groups documents with similar content. The problem is often referred to as *automatic document classification* because the clustering algorithm is not seeded with labeled training instances or a description of the groups it should form. Clustering has been studied extensively in the literature, and the common element among clustering methods is a model of *word cooccurrence* that is applicable to text classification problems in general. The clustering of documents using word cooccurrences between documents results in groups of documents that contain overlapping sets of words. A historical account of clustering research is given by van Rijsbergen [89], who also discusses cluster similarity coefficients applied to simple word matching techniques. Salton [75] discusses clustering approaches that use *tf-idf* representations for text. In addition, several probabilistically based clustering algorithms have been presented, including one by Croft [23], and more recently by TDT participants in the context of event clustering [5, 92].

Efforts to provide cluster-based organizations for large collections have become a focus of user-interface research. In this setting, the clusters of a collection are presented to the user, perhaps based on an initial query, for subsequent manual browsing

and honing of initial results [85]. These techniques have been incorporated into user interfaces for search engines on the web. For example, many internet search engines provide this feature as an alternative method for examining the ranked list of web pages that result from a search. Kohonen feature maps [43] have been adapted to display a colorized landscape of topic concentrations, which can be further explored by the user by drilling in and out of specific topographical locations. Other clustering algorithms, such as Scatter/Gather, have been incorporated in user-interface visualization techniques by Hearst and Pedersen [37]. In addition, visualization techniques based on Inquiry have been demonstrated by Swan and Allan [82], and Leuski and Allan [46].

One of the previous applications of document clustering is *cluster-based retrieval*, which is a method for improving document retrieval in terms of speed and effectiveness [73, 75]. When a user's request is posed to a collection that has been pre-clustered, the documents that fall into the clusters related to the request are returned to the user. In addition, if the collection is pre-clustered, searching cluster prototypes instead of all documents in a collection usually results in a faster search. Assuming that the contents of the documents in a cluster are related, returning the documents from clusters closest to the user's request should have the effect of improving the number of relevant documents returned. If this approach of pre-clustering works, it would suggest that the *cluster hypothesis* holds for the collection, that is, that documents relevant to the same request tend to cluster [89]. However, the results presented by Voorhees [90] as well as Croft [22] suggest that some collections and requests benefit from pre-clustering, while others do not.

There are several issues involved in the actual implementation of a clustering algorithm, including:

1. What document representation to use?
2. How to incorporate domain knowledge?

3. What comparison strategies result in useful groupings?
4. How many system parameters are required?
5. Can stable parameters be obtained automatically?
6. Can effectiveness be measured?

In this dissertation, we explore answers to these questions through a review of the previous approaches to document clustering, as well as by testing the effectiveness of on-line strategies for the problems of event clustering and new event detection.

Many of the previous approaches to clustering do not contribute to solutions to these problems. For example, many effective clustering algorithms, such as those presented by Cutting et al. [24] and Yang [98], assume that all the documents are available before clustering begins. In the on-line environment, documents are processed sequentially, and each document is either placed into an existing cluster or initiates a new cluster; therefore, many aspects of these retrospective approaches are not directly applicable to on-line document processing. In addition, several algorithms have been introduced that require specification of the number of clusters to generate before the clustering process begins. A review of such approaches is presented by Can and Ozkarahan [14]. However, these solutions are not applicable to event clustering since we do not know *a priori* the number of events that will be encountered during processing.

The previous work that most influences our approach to new event detection is based on agglomerative hierarchical clustering, single-pass clustering, and the cluster comparison strategies that are applicable to clustering methods. In the remainder of this section, we discuss this research in more detail.

2.3.1 Agglomerative Hierarchical Clustering

Several issues pertaining to document clustering have been explored using an *agglomerative hierarchical clustering* approach. Much of the previous research in this area is discussed by Willet [94]. Though probabilistic approaches to clustering have been presented [89, 22, 5, 92], in practice, much of the research in this area uses some form of the Vector Space Model, which is described in the context of hierarchical clustering by Salton [75]. In the Vector Space approach, each document in a collection has a vector representation where the elements of the vector correspond to the words in the lexicon of the corpus. It is common for *tfidf* weights to be used as vector feature weights for the words that appear in the document, and the similarity between two documents is often measured as the cosine of the angle between document vectors.

1. Compute all pairwise document-document similarities.
2. Place each of the N documents into its own cluster.
3. Form a new cluster by combining the most similar pair of current clusters i and j .
4. Update the similarity matrix by deleting the rows and columns corresponding to i and j .
5. Calculate the entries in the row and column corresponding to the new cluster resulting from the merge of i and j .
6. Repeat steps 3, 4, and 5 until one cluster remains.

Figure 2.2. Hierarchical Agglomerative Clustering Algorithm

The basics steps of the hierarchical clustering algorithm are described in Figure 2.2, which is an adaptation of the algorithm described by Salton [75]. The general approach is to populate a similarity matrix with an exhaustive pairwise comparison of

document vector similarity values. Initially, each document is placed its own cluster. After each pass of the data, the two clusters that are most similar are merged, and the matrix of similarity values is updated. In practice, the hierarchy can be represented by a tree structure that is updated after each merge. The process repeats until one cluster remains, which is represented as the root of the tree. The tree can be searched bottom-up or top-down by supplying a similarity threshold.

2.3.2 Single-Pass Clustering

Single-pass clustering or *incremental clustering* is an approach for creating clusters on-line. The algorithm is discussed by van Rijsbergen [89] and depicted in Figure 2.3.

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

Figure 2.3. Single-Pass Clustering Algorithm

The single-pass algorithm sequentially processes documents using a pre-specified order. The current document is compared to all existing clusters, and it is merged with the most similar cluster if the similarity exceeds a certain threshold. The single-pass

algorithm results in faster processing than the agglomerative hierarchical clustering algorithm, even though both approaches have an $O(n^2)$ asymptotic running time. The main disadvantage of the single-pass method is that the effectiveness of the algorithm is dependent on the order in which documents are processed. This is not a problem when the data are temporally ordered, because the order is fixed.

2.3.3 Cluster Comparison Strategies

The methodology for comparing a document to a cluster or the contents of two clusters has the greatest effect on the resulting grouping of documents. For example, in the agglomerative hierarchical clustering algorithm (Figure 2.2, page 24), the similarity matrix is updated by deleting the rows and columns corresponding to the two clusters that are merged. The new cluster results in a new row and column containing a combination of similarity values to be used in subsequent comparison steps. The three common strategies for combining similarity values are known as *single-link*, *complete-link*, and *group-average* (average-link) clustering. Voorhees [90] reviews the graph-theoretic tie between these strategies and the properties of the clusters each produces. In general, the three strategies lead to different clusterings.

In the single-link strategy, when clusters i and j are merged into a new cluster, the similarity value between the new cluster and any existing cluster k is the maximum of the similarity values between clusters i and k , and clusters j and k . The complete-link strategy uses the minimum of the similarity values between clusters i and k , and clusters j and k ; and the average-link strategy uses the average of these values. Thus, the single-link method is based on the most similar pair of values, the complete-link method uses the least similar pair, and the average-link method is a mixture of the other approaches.

The single-link, average-link, and complete-link clustering strategies are associated with retrospective clustering algorithms, and in particular with agglomerative

hierarchical clustering, where a similarity matrix between all documents is available before processing begins. In the experiments to follow, we test these strategies in the context of on-line clustering. Each strategy, when applied to the on-line environment, results in a different grouping of documents than when the same strategy is applied to a retrospective environment.

Our on-line approach to event clustering is based on the single-pass algorithm (Figure 2.3, page 25). The algorithm described above specifies that when a document is merged into a cluster, the cluster’s prototype is recomputed (step 5). However, our previous work suggests that this is not a necessary step for an effective single-pass clustering algorithm for event-based classification [4, 59]. Instead of formulating a single cluster prototype or classifier, we use separate classifiers for each document, and compare clusters using single-link, group-average, or complete-link clustering strategies. In what follows, we refer to these three approaches collectively as *cluster comparison strategies*.

In the on-line clustering environment, each document is analyzed sequentially and is either placed into an existing cluster or initiates a new cluster and thus becomes a cluster seed. Figure 2.4 illustrates the differences the three comparison strategies can have on the classification of a document. In the figure, the current document is C , and six documents have been processed and assigned to clusters R , B , and G .

The first step to the on-line versions of the single-, average-, and complete-link strategies is to calculate the similarity between document C and the documents in each cluster. The next step is to calculate a cluster comparison value between document C and each cluster. The comparison values for this example are listed in separate boxes at the bottom of Figure 2.4 for each of the three comparison strategies. With the on-line single-link strategy, the cluster comparison value is the maximum document similarity in each cluster. Using this strategy, C is placed in the cluster containing the document with maximum similarity to C , which is cluster B in Fig-

ure 2.4. The on-line average-link cluster comparison values are calculated by taking the average of document similarities in each cluster. In the figure, cluster R has the highest similarity to document C using this strategy. In the on-line complete-link strategy, comparison values are the minimum document similarities in each cluster, and thus cluster G has the highest similarity using the complete-link approach.

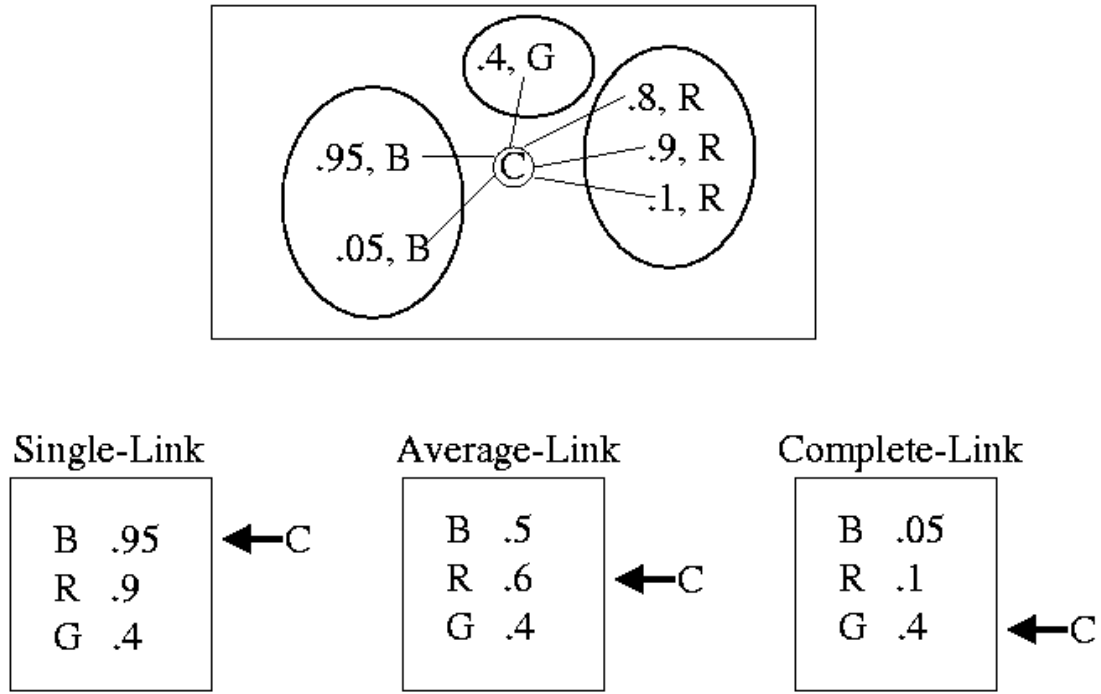


Figure 2.4. On-line Cluster Comparison Strategies.

In addition to the cluster comparison strategy, a thresholding methodology is needed that affects the decision for generating a new cluster. If we are to use clustering for new event detection, then choosing this threshold is important. From the example in Figure 2.4, choosing a threshold of 0.5 would result in document C initiating a new cluster using the on-line complete-link strategy. The other strategies result in cluster comparison values that are above this threshold, and thus C is assigned to an existing

cluster; however, a high enough threshold would result in C creating a new cluster for on-line average-link and on-link single-link strategies. We discuss threshold estimation issues in more detail in Chapters 6 and 7.

2.4 Conclusion

Our literature review finds that the problem of new event detection has not been studied prior to TDT research efforts. Event-based text classification has been reported in a frame-based context; however, the focus of this work was natural language processing and news-photo organization, not document classification. In addition, we found that the related document classification research using the domain of news is based on the notion of topics, not events.

The work that most influences our approach to new event detection can be found in the areas of ranked-retrieval and document clustering. The text representation for ranked-retrieval and document clustering that has emerged in the research uses an underlying model of word-cooccurrence. The lexical features extracted from documents are represented as *tf·idf* weights. The success of this approach suggests that it may be useful as a representation for new event detection. In addition, an approach to new event detection can be derived from the basic step of an on-line document clustering algorithm. In particular, a new event is detected whenever a new cluster is formed. Therefore, we explore document clustering using a *tf·idf* representation and test whether this approach will lead to a good solution for new event detection.

In addition to a representation for text, there are many aspects to clustering algorithms. Of primary concern to our work is the manner in which documents are processed, and the way a document is compared to existing clusters. The definition of on-line new event detection makes a single-pass clustering approach the logical starting point for a solution to new event detection. The work of Voorhees [90] and Willet [94] suggests that different cluster comparison strategies give rise to different

groupings of documents. Since we can not assume that previous comparison methods which worked well for retrospective clustering will work equally well for the on-line environment, it will be important to test different clustering comparison strategies in the context of new event detection.

In the experiments to follow, we use a single-pass algorithm as a basis for new event detection, and test the different cluster comparison strategies and their impact on classification accuracy. One approach to new event detection is to cluster news stories and return to the user the earliest story in each cluster, that is, the one with the lowest timestamp. Using this approach, the system would classify a document as containing discussion of a new event if the document becomes the seed of a new cluster. However, the data from our experiments suggest that new event detection is not necessarily the by-product of a document clustering algorithm. It will become evident that our solution to new event detection is a process for determining cluster seeds, but that the clustering of documents is not a fundamental function of our approach.

CHAPTER 3

EXPERIMENTAL DATA AND EVALUATION METHODOLOGIES

3.1 TDT Corpora

One of the main contributions of the Topic Detection and Tracking (TDT) research effort is the creation of event-based corpora comprising newswire and broadcast news sources. The Linguistic Data Consortium ¹ (LDC), a participant in the project, has collected, maintained, and annotated several sources of news. The data are the first corpora to be judged and annotated for events. In addition to event detection and tracking, other problems such as text segmentation have been evaluated using the TDT data.

The TDT1 corpus was used for the Pilot Study (TDT1). The corpus comprises 15863 documents evenly distributed between CNN transcribed broadcast news and Reuters newswire dating from July 1, 1994 to June 30, 1995. During the second phase of TDT, i.e., TDT2, the LDC collected and annotated 26 weeks of news from six sources, namely the New York Times News Service, Associated Press Worldstream News Service, CNN Headline News, ABC World News Tonight, PRI The World, and VOA English News Programs.

Relevance judgments were assessed for documents pertaining to *events* using the definitions in Section 1.1. The judgements in TDT1 were obtained by two independent groups of assessors and then reconciled through an adjudication process to form a set of final judgements. Documents were judged on a ternary scale to be non-relevant, to

¹<http://www ldc upenn edu>

have content relevant to the event, or to contain only a brief mention of the event in a generally non-relevant document. For the 25 events of the TDT1 Corpus, for example, 1132 documents were judged relevant, 250 documents were judged to contain brief mentions, and 10 documents overlapped between the set of relevant documents and the set of brief mentions. In what follows, we remove brief mentions from processing and measure classification using the relevant and non-relevant documents.

The broadcast news sources of the TDT2 corpora are available in parallel audio and text signal formats. The audio signals were converted to text using Dragon Systems’ automatic speech recognition (ASR) system [93]. The document boundaries for the audio sources of the TDT2 corpora were determined as part of the annotation effort. Manual text transcriptions, including Closed Caption (CCAP) and Federal Document Clearing House (FDCH) formats, were also available for the audio data. In the experiments that follow, we analyze the impact of the ASR technology by comparing results from corpora using the speech recognition process to those using the manual Closed Caption process.

The news collected for TDT1 and TDT2 is divided into four sets, depicted in Table 3.1. Each corpus contains relevance judgments for between 25 and 35 specific events. The participants in the Pilot Study chose the events in the TDT1 corpus, while the events for TDT2 were chosen by randomly sampling documents, and selecting the event in the sampled document if one was actually discussed. The corpora were exhaustively and independently judged for each selected event; however, all the events in the data were not identified.

Table 3.1. TDT Corpora.

Corpus	Dates	#-Sources	#-Documents	Words/doc
TDT1	07/01/1994 - 06/30/1995	2	15863	460
TDT2-Train	01/04/1998 - 02/28/1998	6	20404	341
TDT2-Dev	03/01/1998 - 04/30/1998	6	20462	314
TDT2-Eval	05/01/1998 - 06/30/1998	6	22443	333

Table 3.2. TDT Events.

Corpus	#-Events	#-Rels	#-Briefs
TDT1	25	1132	250
TDT2-Train	35	4159	1103
TDT2-Dev	25	608	78
TDT2-Eval	34	1883	472

The events that were used in this dissertation are summarized in Table 3.2. A short description of each event is listed in the Appendix.

3.2 Evaluation

In this section we discuss some of the effectiveness measures that are used to evaluate Information Retrieval and text classification experiments. We describe the measures used for TDT, which are based on evaluating text classification as a *detection* task. Evaluation is a challenge for text classification, because there does not exist an agreed-upon single-valued metric that uniquely captures the accuracy of a system. It is often the case that two or more measures are needed, and efforts to define combination measures do not necessarily lead to an applicable measure of utility. In what follows, we assume utility is a function of user satisfaction with the classification effectiveness of a system.

In practice, utility is constantly changing, and effectiveness requirements for some information needs require higher utility than for others. For example, consider two detection systems: a smoke detector and a voice-activated password system. It may be acceptable for the smoke detector to sound occasionally when no fire exists, but if an alarm does not sound when there is an actual fire, the system is worse than useless, it is deadly. In other words, the owner of the smoke detector has a low tolerance for false-alarms and *no* tolerance for misses. A user of the voice activated password system has a different utility function. It may be tolerable for the system to miss a few times by failing to recognize a voice, but one false identification gives access

to a potentially harmful intruder. With these caveats in mind, we avoid inventing yet another measure, and we report several effectiveness measures which have been previously used for the analysis of text classification experiments.

3.2.1 Effectiveness Measures

Text classification effectiveness is often based on two measures. It is common for Information Retrieval experiments to be evaluated in terms of *recall* and *precision*, where recall is the percent of relevant instances classified correctly, and precision is the percent of relevant instances in the set of documents returned to the user.

In TDT, and the work described here, *system error rates* are used to evaluate text classification. These errors are system *misses* and *false alarms*, and the accuracy of a system improves when both types of errors decline. In new event detection, misses occur when the system does not detect a new event, and false alarms occur when the system indicates a document contains a new event when in truth it does not. In addition to system error rates, we report the traditional text-retrieval measures of recall and precision.

It is desirable to have one measure of effectiveness for cross-system comparisons. Unfortunately, no measure above uniquely determines the overall effectiveness characteristics of a classification system. Several definitions for single-valued measures have emerged, and are reviewed by van Rijsbergen [89]. One prevalent approach is to evaluate text classification using F1-Measure [48], which is a combination of recall and precision. In TDT2, and in this work, a *cost function* was used to analyze detection effectiveness. The general form of the TDT cost function is

$$Cost = cost_{fa} * P(fa) * (1 - P(event)) + cost_m * P(m) * P(event), \quad (3.1)$$

where $P(fa)$ is the probability that a system produces a false alarm, i.e., its false alarm rate, $P(m)$ is the probability that a system produces a miss, i.e., its miss rate, and $P(event)$ is the prior probability that a document is relevant to an event. In TDT2, cost was defined with $P(event) = 0.02$, and the constants $cost_{fa} = cost_m = 1.0$.

The methods for calculating the effectiveness measures for on-line new event detection, clustering, and tracking are the same, and are summarized below using the following table:

	Relevant	Non-Relevant
Retrieved	a	b
Not Retrieved	c	d

where the retrieved documents in the table are those that have been classified by the system as positive instances of an event, and the relevant documents are those that have been manually judged relevant to an event. The effectiveness measures used in this dissertation can be derived from the table as follows:

$$Recall = R = \frac{a}{a+c},$$

$$Precision = P = \frac{a}{a+b},$$

$$F1-Measure = \frac{2PR}{P+R},$$

$$Miss Rate = M = \frac{c}{a+c},$$

$$False Alarm Rate = F = \frac{b}{b+d}, \text{ and}$$

$$Cost = 0.98 * F + 0.02 * M.$$

We discuss these measures in the context of specific classification problems in later chapters.

3.2.2 Experiment Methodologies

It should be emphasized that we are interested in evaluating a text classification task, and not a text ranking task. Evaluation for ranking is well understood; however, there are several biases inherent in the evaluation of classification. One that is often studied is bias due to estimating effectiveness from data based on optimized parameters. We acknowledge that there will be bias in classification, and we attempt to learn estimation biases to improve classification effectiveness.

There are several available methodologies for classification experiments. Most of them are based on a variant of the *holdout method*, which is depicted in Figure 3.1. Our experimental methodology uses this method, with training done on the events from the TDT1, TDT2-Train, and TDT2-Development corpora, and testing is done on the events from the TDT2-Evaluation corpus. Since the number of judged events available is relatively small in comparison to the total number of events that appear in a corpus, the holdout method is a biased methodology for estimating the expected effectiveness of an approach in other corpora. This is because a single test using holdout is biased towards the measures resulting from one set of observations.

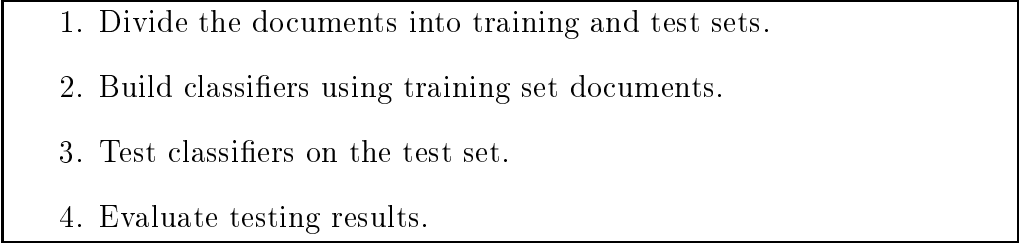
- 
1. Divide the documents into training and test sets.
 2. Build classifiers using training set documents.
 3. Test classifiers on the test set.
 4. Evaluate testing results.

Figure 3.1. Holdout Evaluation Methodology.

Kohavi [42] studies the bias inherent in expected (mean) effectiveness using holdout and randomization approaches to evaluation. These methods include *cross val-*

idation and *bootstrapping*. Cross validation, depicted in Figure 3.2, is a repeated application of the holdout method. We have found this method effective for estimating and selecting stable system parameters. For example, we used *leave-one-out cross-validation* in order to avoid overfitting our threshold parameters in previous experiments on the TDT1 corpus [3].

The general algorithm for k -fold cross-validation is to partition the events into k sets randomly, and to leave one set out while finding parameters that best fit the remaining $k - 1$ sets. The process repeats k iterations, leaving a different set out each time. A stable parameter would show a constant mean and low variance in effectiveness when measured over the k iterations.

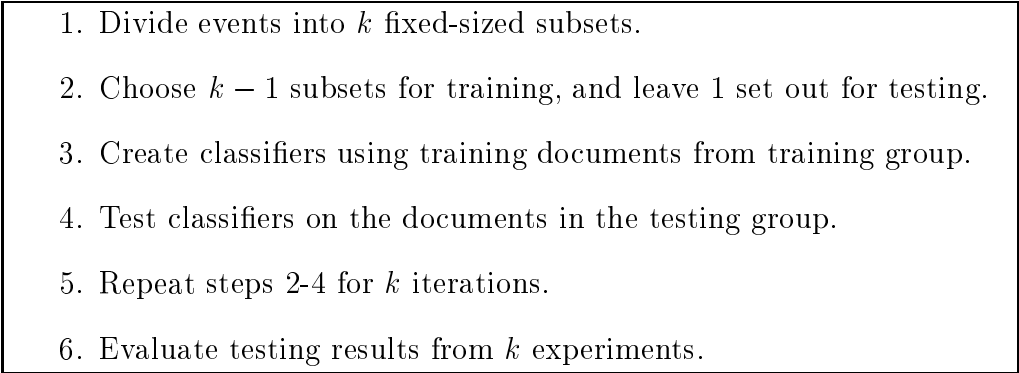
- 
1. Divide events into k fixed-sized subsets.
 2. Choose $k - 1$ subsets for training, and leave 1 set out for testing.
 3. Create classifiers using training documents from training group.
 4. Test classifiers on the documents in the testing group.
 5. Repeat steps 2-4 for k iterations.
 6. Evaluate testing results from k experiments.

Figure 3.2. k -fold Cross Validation Evaluation Methodology.

We also tested the *bootstrapping* methodology for reducing estimation bias [21, 42]. Bootstrapping is a random sampling approach that statistically increases the number of evaluation points in a testing sample. The assumption in bootstrapping is that the distribution of effectiveness obtained during testing is representative of the distribution that exists over the population as a whole. Figure 3.3 describes the procedure we used on the effectiveness measures resulting from new event detection

experiments. Bootstrap samples are randomly selected from events with replacement. Mean effectiveness measures for the sample are calculated. The process repeats for many iterations, giving rise to distributions of mean effectiveness measures.

1. Formulate classifiers on training set.
2. Evaluate classifiers on testing set.
3. Assuming that there exist k classifiers from steps 1 and 2, create a bootstrap sample of size k by randomly sampling *with replacement* the effectiveness measures from step 2.
4. Repeat step 3 for n iterations.
5. Evaluate based on effectiveness distribution from the n random bootstrap experiments.

Figure 3.3. Bootstrapping Evaluation Methodology

To evaluate the effectiveness of bootstrapping, we ran $k = 25$ events and $n = 10000$ iterations on error rates from a new event detection experiment on the TDT1 Corpus. This experiment produced the bootstrap distribution in Figure 3.4, which resulted in effectiveness measures with a mean miss rate of $\mu = 40.5\%$ with $\sigma = 7.6\%$, and a mean false alarm rate of $\mu = 7.8\%$ with $\sigma = 4.0\%$.

We evaluated the bootstrap process by comparing the distributions obtained for new event detection on the TDT1 corpus with the results obtained on the TDT2 corpora. In general, the new event detection results from the TDT2 corpora using the Closed Caption sources fall within two standard deviations of the mean we obtained from the TDT1 corpus. This suggests that bootstrapping was somewhat effective for estimating expected effectiveness; however, the number of events for the TDT1 corpus are most likely too few to be representative of the population of news events as

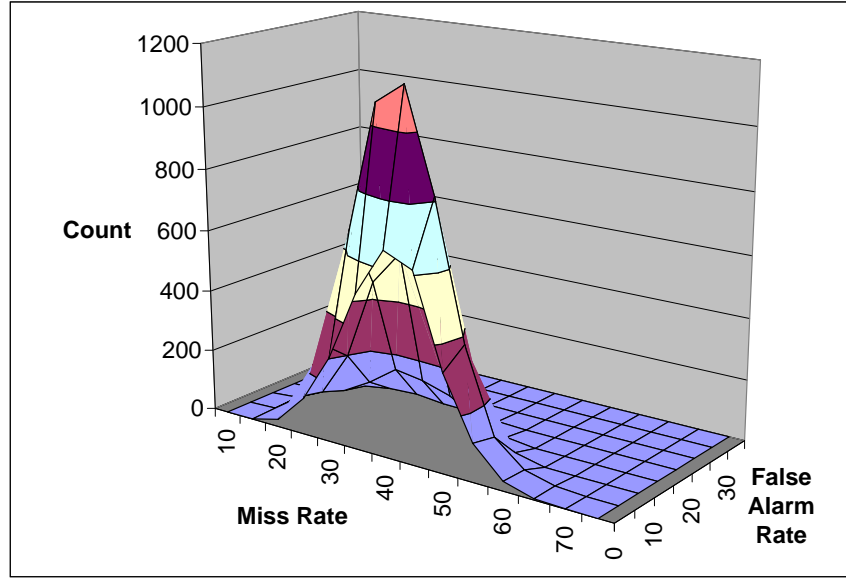


Figure 3.4. New Event Detection: Expected error rates from bootstrapping.

a whole. It is therefore not surprising that the effectiveness measures from the TDT2 corpora are not closer to the bootstrapped mean from the TDT1 corpus. Since the bootstrap experiments did not yield more predictive measures, we did not use this methodology to evaluate effectiveness in the experiments that follow.

A final important issue regarding evaluation is a significance test for effectiveness improvements. During system development, an approach is considered successful if it significantly improves effectiveness over another approach. In what follows we chose a two-tailed sign test with confidence $\alpha = 0.05$ to determine significance. The sign test is convenient because it can be performed manually. It assumes that two approaches to the same problem are independent and will each claim better effectiveness on 50% of the events being tested. Furthermore it is used to test the hypothesis that approach A is better than approach B. The values in Table 3.3 illustrate the number of events one approach must claim over another to be significantly better using this methodology.

Table 3.3. Sign Test for Significant Improvements

Number of Events	Number (A > B)
200	112
100	59
50	31
25	17
10	8

3.2.3 Multiple-Pass and DET Curve Evaluation Methodology

Since only 25-35 events in each corpus were judged, we use an evaluation methodology for new event detection that expands the number of experimental trials. The methodology uses 11 passes through the stream. Assuming that there are only 25 events available for a particular experiment, the goal of the first pass is to detect a new event in the 25 documents in which one of the 25 known events first occurs. The second pass excludes these documents, and the goal is to detect the second document for each of the 25 known events: the second document artificially becomes the first document in the stream. The process repeats to skip up to 10 documents for each event. If an event contains fewer documents than the number of documents to be skipped in the pass, the event is excluded from evaluation in that pass.

In our implementation, an appropriate change to a classifier’s threshold will trade-off recall for precision or misses for false alarms. In what follows we use *detection error tradeoff* (DET) graphs [52] to visualize the tradeoff between misses and false alarms when evaluating new event detection and tracking. These graphs are similar in nature to Receiver Operating Characteristic (ROC) graphs [83], which have been used to evaluate machine classification [65] and medical diagnostic experiments.

An example of a DET graph is illustrated in Figure 7.5 (page 115). The decision scores for each document over the 11 passes of the data described above are pooled and sorted by score. The points on a DET curve are determined by incrementally moving down the sorted list of documents and re-calculating miss and false alarm

rates at each document. A side-effect of this methodology is that events with more relevant documents are more heavily weighted in the DET curve. In addition, the graph is scaled based on a normal distribution of the error rates, and points closer to the origin indicate better overall effectiveness. The graph also contains the evaluation point corresponding to the pooled average effectiveness across events.

The DET graph is also used to evaluate tracking results; however, the methodology for tracking uses decisions from single passes through the data. When comparing two DET curves from different approaches to the problems of new event detection or tracking, we assume that approach A is more robust than approach B when *all* the points on DET curve A are below the points on DET curve B.

CHAPTER 4

TRACKING

In this chapter we discuss our approaches to the problem of *event tracking* in a stream of broadcast news. The task involves formulating a classifier from a few sample documents that contain discussion of the same event. The classifier is applied to the subsequent stream to retrieve documents that are related to the target event.

The tracking task was defined by the Topic Detection and Tracking (TDT) research initiative, a DARPA-sponsored effort comprising research groups from several commercial and academic sites. The task is a supervised process that is similar in nature to the problems of information filtering, routing, and categorization (Section 2.2.1). In the Information Retrieval literature, these problems have been discussed in the context of *relevance feedback* approaches, where it is assumed that a user provides judgements for some of the documents retrieved by the system from an initial information request. The user's judgements become part of a process that reformulates the classifier to improve the set of documents returned.

We assume that users of a tracking system will provide a small number of training samples for events they are interested in following over time. For example, consider an emerging markets securities trader who is interested in following stories pertaining to the political developments in the countries he trades. In one scenario, news about a coup in Brazil sends Brazilian financial instruments spiraling downwards. The trader supplies the system with a few sample documents describing the coup (perhaps only the first story identified by the new event detection process), and the tracking system builds a classifier to find subsequent related stories from on-line news feeds that he

already monitors manually. The tight coupling of the new event detection and event tracking processes into one system allows the trader to monitor more news than would be possible manually.

The text representation and classifier formulation process we use for tracking is strongly influenced by our previous work on filtering [49, 56, 57], which is based on testing feature selection, weight learning, and classifier expansion approaches using the Inquiry [12] retrieval system. In the following work, we test our hypothesis that these approaches significantly improve document classification effectiveness by evaluating them in the context of the TDT2 tracking problem. In particular, we adapt previous implementations and test the approaches that may be useful for classifying events on-line. We discuss the details of our tracking implementation in this chapter.

Much of the research related to tracking has been explored using TREC’s filtering methodology [34], which focuses on training classifiers for general topics with one to a few thousand relevant training documents from heterogeneous sources including news, web, email, and other types of documents. In TREC filtering, a classifier for a topic is constructed from a query specification (Figure 2.1, page 19) and a labeled training set of documents. The classifier is then evaluated on a corpus independent of the one containing the training documents.

The tracking task is similar in nature to the TREC filtering task; however, there are some differences. In tracking, the information requests specifically concern *events* and not *topics*, and classifiers are trained using only 1, 2, 4, 8, or 16 relevant training documents for each event. In general, tracking is based on a process where a user would be willing to provide only a few documents with which to formulate classifiers. Each tracking experiment is run over a set of events using a prespecified number of relevant training documents (Nt). If the system is being evaluated for four relevant training stories, that is, $Nt = 4$, then all stories in the stream up to and including the fourth relevant training story are considered the training corpus, and the testing

corpus is the remainder of the stream. In contrast, the experimental methodology for filtering uses separate training and testing corpora with varying numbers of relevant training documents.

The purpose of exploring tracking in this dissertation is to find a representation for text that can be applied to the problem of new event detection. Our solution to new event detection uses a text classification approach to the problem, and the tracking task provides a controlled testing environment in which to evaluate approaches to classifier formulation and threshold estimation.

One of the goals of the tracking experiments here is to show how few documents are needed to track events effectively. In particular, we focus on analyzing predictive tracking experiments that were done in the context of the second phase of TDT, the results for which were produced by the National Institute of Standards and Technology (NIST). The NIST evaluations are based on tracking with 1, 2, and 4 relevant training instances for each event. Another goal of our tracking experiments is to determine the impact that automatic speech recognition (ASR) technology has on our process. In addition, we compare the results from our approaches to those of other TDT participants.

Though much progress has been made in improving the representation for text classification, several questions remain only partially answered, including:

- What representations are applicable to on-line classification ?
- Are there properties particular to events that can be exploited to improve effectiveness?
- How many and what types of features are useful ?
- Which and how many non-relevant documents are useful ?
- Which weight learning approaches help ?

- Is global parameter estimation stable ?
- Does on-line classifier reformulation improve effectiveness ?
- Does *tf* or *idf* help ?
- Does ASR technology affect classification ?

In this chapter, we attempt to answer these questions through experiments based on the implementation described in the next section. We analyze different approaches to classifier formulation, and compare feature selection and weight-learning steps as extensions of a baseline classifier formulation process. We also discuss an unsupervised adaptive approach to event tracking that captures the property of *event evolution* in broadcast news.

4.1 Implementation

This section describes the details of the classification model we used to track events. The same text representation was also applied to the separate problems of new event detection and event clustering.

The general representation for document classification that has emerged in our research uses the Inquiry retrieval system [12], which is based on the Inference Network model for document retrieval [86]. Inquiry was developed in the Information Retrieval Lab at the University of Massachusetts, and its parameters have been honed through many empirical studies. Over the past decade the system has been tested as a component technology for solving problems in areas such as Stemming [45], Query Formulation [96], Segmentation [61], On-line Classification [59, 13], Statistical Inference [32], and Cross-Language Processing [6]. The version that we used has been tested annually at Text REtrieval Conferences (TREC) [34].

A classifier is formulated automatically from the lexical features of the training set of documents using the operators from Inquiry’s query language. Thus, we represent

an *event* discussed in a set of relevant training documents with a classifier comprising a query syntax and threshold. A separate threshold is estimated for each classifier, and documents on the stream that have similarity exceeding the threshold are classified as positive instances of an event, that is, the contents of the document are assumed to discuss the same event as the relevant document(s) with which the classifier was formulated. In our tracking system, classifiers are formulated using one or more relevant training documents.

4.1.1 Inverted Index File

The *inverted index file* is a data structure of primary importance to our implementation¹. A schematic of an inverted index file is depicted in Figure 4.1. The underlying structure is a dictionary of the words that appear on the news stream. When processing a stream, the documents containing each word and the number of occurrences of the word in each document are updated on-line. If the position information for each word is stored in the index, then phrases and proximity occurrence statistics can be determined.

The inverted index file provides two main statistics that have been studied extensively in text retrieval: *term frequency*, or the number of times a word appears in a document, and *document frequency*, or the number of documents within which a word appears. Document frequency is usually expressed as *inverse document frequency*, or *idf*. These statistics are combined and normalized in different ways by the different text retrieval systems discussed in Chapter 2. The classifier and document features used in the text representation that follows can be derived from this structure.

¹This organization of text is sometimes referred to as a *concordance*.

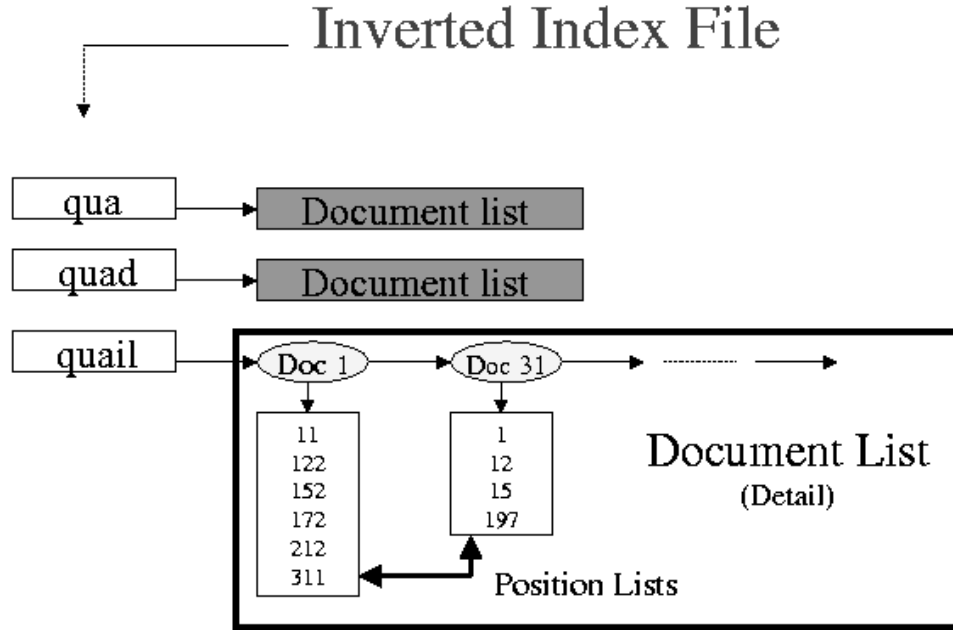


Figure 4.1. Inverted Index File Organization.

4.1.2 Text Representation

The implementation we evaluate for tracking is based on linear classification. Text is represented as vectors of features that are real-valued weights corresponding to words appearing in text. During processing, feature weights are potentially re-computed at the arrival of each new document on the stream. We have tested this representation with an implementation that makes use of both single and multiword features. In previous topic-based experiments, we found that features resulting from phrases and proximity information can improve classification effectiveness over features derived from single words [57]. One of the goals of our experiments is to evaluate

the effectiveness of different multiword feature representations and their impact on event-based classification.

In our implementation, we use a *tf* representation for classifier feature weights, which is a variant of a function originally introduced by Robertson et al.[68]. When comparing a classifier to a document, the document is represented with *tf.idf* feature weights where

$$tf = t / (t + 0.5 + 1.5 * \frac{dl}{avg_dl}), \text{ and} \quad (4.1)$$

$$idf = \frac{\log(\frac{C+.5}{df})}{\log(C + 1)}. \quad (4.2)$$

Equation 4.1 is used for both classifier and document weights. In these equations t is the number of times the lexical feature appears in the document, and dl is the document's length in words. The remaining values for Equations 4.1 and 4.2 are derived from an auxiliary corpus. C is the number of documents in the auxiliary corpus, avg_dl is the average number of terms in a document, and df is the number of documents in which the term appears. If the term does not appear in the auxiliary corpus, a default value of 1 is used for df .

4.1.2.1 Classifier Formulation

We use an automatic process that works on-line to create a classifier from single or multiple documents. The classifier formulation process has three main steps: feature selection, weight assignment, and threshold estimation. The selected features and weights are used to construct a classifier using Inquiry's query syntax. This process, using one relevant training document, is depicted in Figure 4.2. The threshold is estimated for the query using an optimization process which is described below.

The methodology for selecting features begins with collecting statistics from the words appearing in the training documents. We assume the training documents

Text Representation

Document Text

Dan Quayle is in the hospital today, and the doctors say he'll stay there for about one week. The 47-year-old former vice president was admitted to Indiana University Medical Center in Indianapolis yesterday evening. He had complained of a progressive shortness of breath. Doctors say they discovered a blood clot in his lung. The hospital spokeswoman says the clot constituted a ...

Text Classifier

Query syntax

```
#q7178 = #WSUM(1
2.408956 clot
2.156014 quayle
1.88575 dan
1.88575 doctor
1.595476 blood
1.595476 lung
1.282037 condition
1.282037 former
1.282037 hospital
1.282037 vice);
```

Threshold Estimation

Document Representation

clot	quayle	dan	doctor	blood	lung	condition	former	hospital	vice
0.987	0.843	0.783	0.034	0.127	0.234	0.002	0.021	0.131	0.387

Figure 4.2. Classifier and Document Representation.

have assessments for both relevant and non-relevant instances; however, the same methodology can be used without non-relevant training instances. We take the non-stopwords from the training documents and sort them by the following measure

$$\frac{r}{R} - \frac{nr}{NR} > 0 \quad (4.3)$$

where R is the number of relevant documents and NR is the number of non-relevant documents in the training sample. The values r and nr are the number of documents in the corresponding relevant and non-relevant training sample containing the word. This measure is greater than 0 if the word appears more in relevant than non-relevant

documents. The number of words in a classifier, i.e., the classifier’s *dimensionality*, is a parameter of the system. If the classifier is formulated with n words, then the top n words sorted by Equation 4.3 are used.

The classifier is ultimately represented as a vector of weights. In the experiments that follow, we use an extension of a weight assignment methodology introduced by Rocchio [70]. This method has been refined in previous work with the Inquiry retrieval engine [1, 57]. Each classifier feature weight maps to a word. Classifier weight $q_{i,k}$ is

$$q_{i,k} = c_1 * tf_{rel} - c_2 * tf_{nonrel}, \quad (4.4)$$

where tf_{rel} is the average tf score (Equation 4.1) for the word in relevant documents, and tf_{nonrel} is the average tf score for the word in non-relevant documents. The constants c_1 and c_2 are determined empirically, and we have found that setting $c_1 = c_2 = 0.5$ works well in comparison to other settings.

4.1.2.2 Document Representation

During processing, the representation of any document arriving at time j , when compared to a classifier formulated at an earlier time i is

$$d_{j,k} = 0.4 + 0.6 * tf_k * idf_k \quad (4.5)$$

where k is the index of the word cooccurring in the classifier and document, and idf is calculated using Equation 4.2.

Since future word occurrence statistics are unknown in real-time applications, the number of documents within which a word will appear is unknown. Our approach is to estimate df for Equation 4.2 from a larger sample of natural language by using an auxiliary corpus from a similar domain. In the tracking experiments that follow, we evaluate different sources for df and their resulting impact on classification accuracy.

4.1.3 Comparing Classifiers to Documents

A similarity value is calculated when comparing a classifier to a document. In the experiments that follow, the similarity between a classifier and document is calculated using the #WSUM operator of Inquiry, which is defined as

$$sim(q_i, d_j) = \frac{\sum_{k=1}^N q_{i,k} \cdot d_{j,k}}{\sum_{k=1}^N q_{i,k}}. \quad (4.6)$$

A document is assumed to discuss the event represented by the classifier if its similarity to the classifier exceeds a certain threshold. When each classifier is formulated, a threshold is determined to control hard classification decisions. Our threshold model for tracking is based on an optimization process over the training data used to formulate the classifier. More specifically, if a classifier is created at time i , that is, when the last relevant training document arrives, then the resulting classifier's threshold for a document arriving at a later time j is :

$$threshold(q_i, d_j) = 0.4 + \theta * (s_{optimized} - 0.4), \quad (4.7)$$

where 0.4 is an Inquiry constant, and $s_{optimized}$ is the similarity value resulting from the classifier that, when applied to the event's labeled training documents, optimizes the target utility function. In the following experiments we optimize for the TDT2 cost function defined by Equation 3.1.

The parameter θ controls the threshold model. When $\theta = 1.0$, then the threshold is $s_{optimized}$. During development, we explored variations of the processes that calculate $s_{optimized}$, and empirically observed that optimization using the same measure used for evaluation leads to the best results. Methodologies for determining appropriate values for θ are discussed below and in Chapter 5.

In order to generate Detection Error Tradeoff (DET) curves (Section 3.2.3), a score reflecting the confidence of each document decision is needed. Our confidence in the

decision that classifier q_i discusses the same event as document d_j is the following score:

$$decision(q_i, d_j) = sim(q_i, d_j) - threshold(q_i, d_j), \quad (4.8)$$

Decision scores greater than 0 imply that classifier q_i classified d_j as a positive instance. We can control classification decisions during processing by adjusting parameter θ in Equation 4.7 above.

4.2 Optimization

In this section, we discuss the results of retrospective tracking experiments that produced the implementation decisions and parameters we used for predictive tracking experiments. Our experimental methodology for tracking is similar to the one used for new event detection and event clustering in that we develop an implementation using the TDT1, TDT2-Train, and TDT2-Development corpora, and evaluate different approaches using the TDT2-Evaluation corpus (Section 3.1).

4.2.1 Parameter Estimation for Tracking

The most important parameter in our implementation is the value for θ to be used in the threshold model described by Equation 4.7 (page 51). We used exhaustive parameter searches over the TDT corpora to determine stable values for θ across different classifier dimensionality as well as different numbers of relevant training documents (Nt). We found that formulating classifiers with 50 features resulted in effective classification. The threshold parameters we found that optimized pooled (story-weighted) cost on the TDT1, TDT2-Train, and TDT2-Development corpora are listed in Table 4.1. We discuss the resulting effectiveness of these parameters in subsequent sections, and in Chapter 5 we describe in more detail the methods we used to obtain our threshold parameters.

Table 4.1. Static tracking threshold parameter θ .

Nt	θ
1	0.2
2	0.3
4	0.4
8	0.6
16	0.8

4.2.2 Adaptive Query Formulation

In addition to our static approach, we tested adaptive tracking approaches that extend some techniques we previously used on the TDT1 corpus [3]. Our adaptive approach is designed to catch the appearance of new lexical features for an event as it evolves in the news coverage. In adaptive tracking, a classifier is initially formulated using the static approach, and then reformulated on-line with features from new documents in the stream. During on-line processing, two lists are maintained that contain the documents assumed to be relevant and non-relevant to each event. The documents from these lists are then used to reformulate the classifier using the same formulation process.

The threshold model for adaptive tracking uses Equation 4.7 (page 51) for two thresholds. One threshold is estimated for detection decisions and another threshold is estimated for reformulation decisions. If a document exceeds the second threshold, it is used to reformulate the classifier and determine a new threshold. If a document's similarity to the classifier is above the decision threshold but below the reformulation threshold, then it is excluded from the reformulation process.

After reformulating a classifier, we calculate the TDT2 cost of the new classifier with respect to its training data. If cost increases, we discard the reformulated classifier and revert to the previous one. This additional step is necessary to prevent the classifier from over-generalizing. We found that for adaptive tracking, classifiers using 10 features worked better than with 20 and 50 features.

In the adaptive tracking experiments that follow, we determined values for the parameter θ for different values of Nt using the optimization process described in the previous section over the TDT1, TDT2-Train, and TDT2-Development corpora. We found that a reformulation threshold of $\theta = 0.85$ worked best. The decision thresholds that were found from the optimization process appear in Table 4.2.

Table 4.2. Threshold parameter θ for adaptive tracking.

Nt	θ
1	0.45
2	0.55
4	0.65

4.2.3 Varying IDF Sources

Our static classifier formulation process described above uses *idf*, that is, inverse document frequency (Equation 4.2, page 48). The document frequency of a word, df is the number of documents in a collection in which the word occurs. However, in the context of an on-line tracking process, this statistic is not readily available; that is, the number of documents in which a word will appear is not yet known. Several solutions to this problem have been tested by TDT participants. They include estimating df from an auxiliary corpus [59], using the df from the current stream [79], or some combination thereof.

We evaluated different sources for *idf* over varying dimensionality for overall minimum cost on the TDT1, TDT2-Train, and TDT2-Development corpora. Our empirical observation is that *idf* calculated using document frequencies from TREC volumes 1,2, and 3, combined with the TREC-4 Routing volume (TREC 123R4), produced lower cost and DET curves than *idf* calculated using the news-only documents from TREC 123R4 as well as an independent CNN broadcast news corpus. Also, TREC 123R4 was a more effective source for document frequency than using

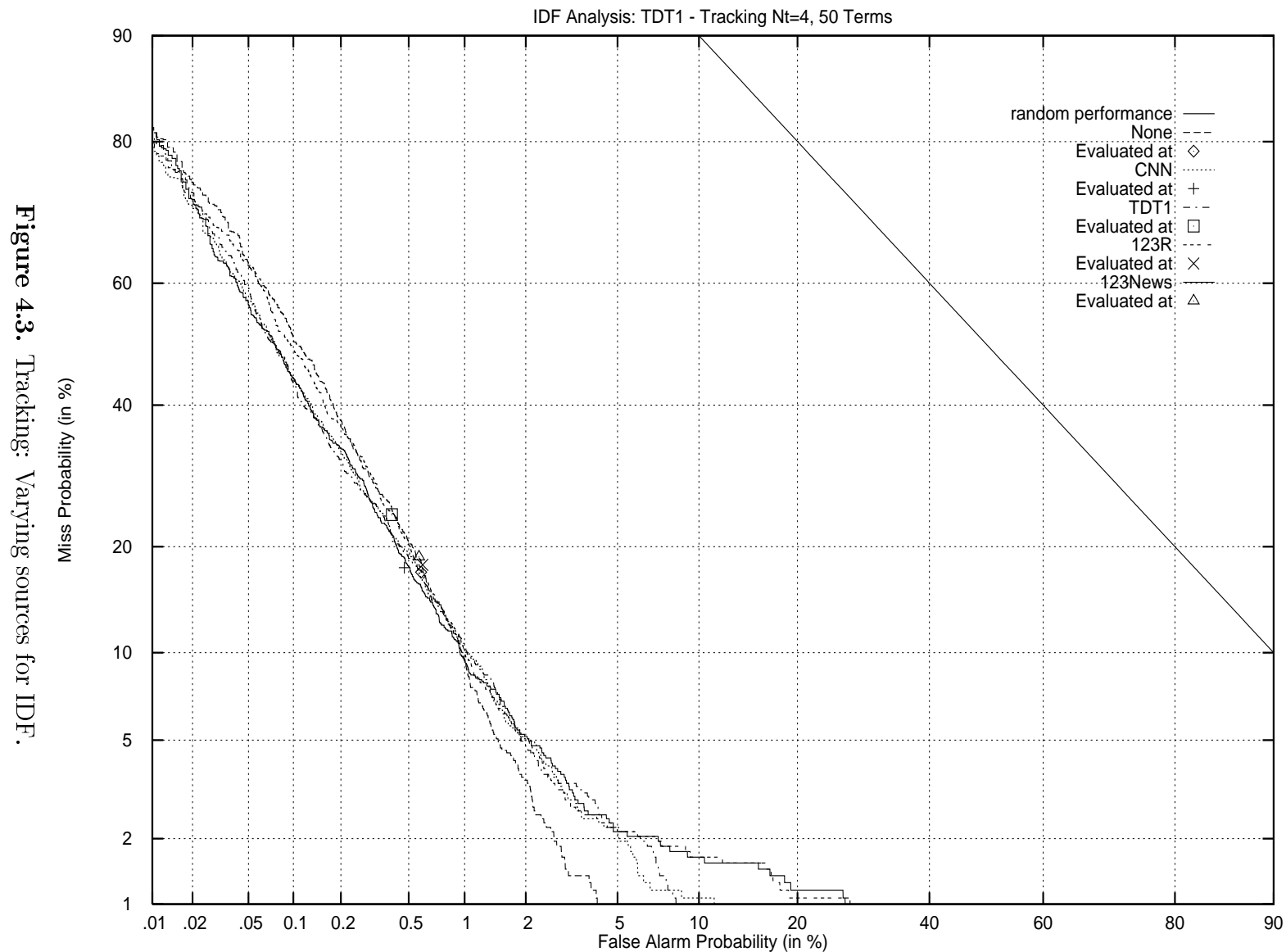


Figure 4.3. Tracking: Varying sources for IDF.

the entire corpus being processed. In general, the overall cost changes between using different sources for *idf* and not using *idf* at all were small at best. Our experience using incremental *idf* for tracking in the TDT Pilot Study [4], however, suggests that document frequency is a useful statistic.

The DET graph in Figure 4.3 contains the curves resulting from using the sources for *df* described above on the TDT1 corpus. The curves show that the different sources led to similar tradeoffs for false alarm and miss rates for the TDT1 corpus. The same analysis on the TDT2 corpora suggested that using the TREC 123R4 source provides a slight advantage over other sources, and we therefore use TREC 123R4 for *idf* in the experiments that follow.

4.2.4 Decision Score Normalization

Recall that the DET curve is based on ranking classification decisions by *decision score*. We found that our original function described by Equation 4.8 (page 52) did not yield good DET curves for tracking, which was due to the tendency of our system to produce different ranges of similarity values for the different events being tracked. We needed to normalize the similarity values from Equation 4.6 (page 51) to produce good decision scores for tracking. In addition, the normalization needed to meet the criterion for TDT, which requires that scores for positive decisions have higher value than scores for negative decisions within and across events.

We looked at different approaches to decision score normalization and their ability to improve DET curves by lowering them on the DET graph. We found that the most effective way to lower curves was to normalize similarity scores using a standard normal transformation. The general form of the transformation is

$$decision_score(q_i, d_j) = \frac{sim(q_i, d_j) - \mu}{\sigma}, \quad (4.9)$$

where $\text{sim}(q_i, d_j)$ is the similarity between classifier q_i and document d_j . μ is the mean, and σ is the standard deviation of a distribution of similarity values using Equation 4.6. This transformation has been previously used for feature representation normalization, and has been discussed in the context of text classification by Lewis [47]. Also, Jin et al.[40] discuss similar approaches for decision score normalization for tracking.

We observed that classifiers for most events produced distributions of similarity values that were relatively normal for relevant documents; but the distributions did not appear normal for non-relevant documents (see Figure 5.1, page 72).

In Figure 4.4, we illustrate DET curves for the TDT2-Evaluation corpus using several resulting normalizations based on the means and standard deviations from the distributions of training instances. The curves in the graph appear very different; however, all the curves in the graph are from the same tracking test run, and thus have the same decision point, which is delineated by the small diamond on the lowest curve in the graph. Experiments setting μ and σ to the values estimated from the relevant documents in the training sample produced the curve furthest from the origin in Figure 4.4. We found that dividing the similarity value by the mean of the relevant documents in the training sample improved the DET curve, resulting in the curve second furthest from the origin. When we set μ and σ to estimates based on the non-relevant training documents, the resulting curve improved significantly over the approaches using the statistics from the relevant training samples. This curve is the second closest to the origin (without the hard decision point). However, in all of these cases, the transformation does not satisfy the decision score requirement that relevant decisions have a higher score than non-relevant decisions across events.

We found that setting $\mu = \text{threshold}(q_i, d_j)$ (Equation 4.7), and σ to the standard deviation of similarity values from both the relevant and non-relevant training data resulted in the curve with points closest to the origin in Figure 4.4. This nor-

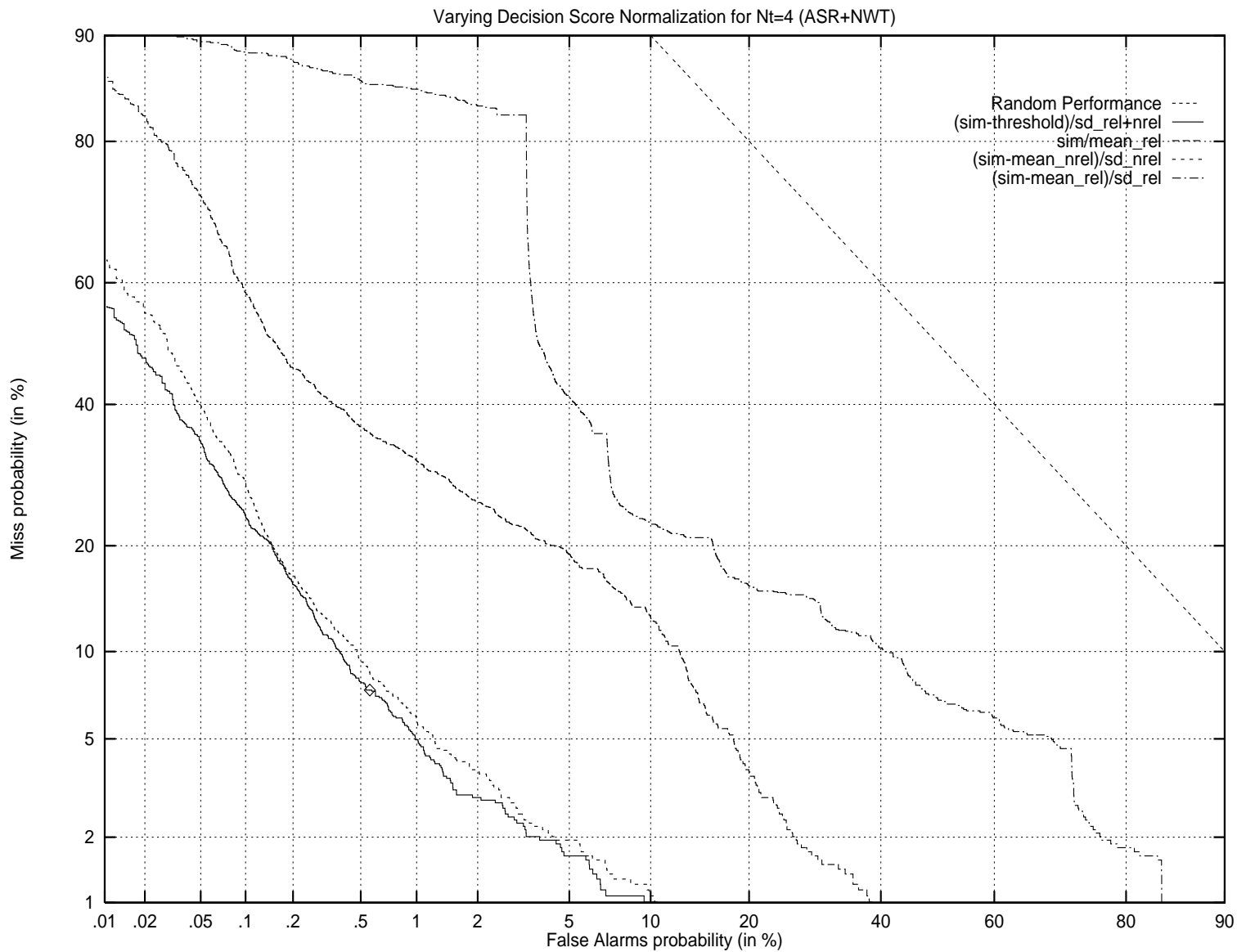


Figure 4.4. Decision Score Normalizations.

malization appeared to work slightly better than using the distributions from the non-relevant training documents, with the additional property of producing decision scores normalized around a value of 0. We used this normalization process for the DET curves produced below.

4.3 Tracking Experiments

In the following section, we evaluate our text representation and classifier formulation process for tracking using the approaches described above. Training and parameter optimization for the following experiments were done using the 85 events from the TDT1, TDT2-Train, and TDT2-Development corpora. The evaluation, which was conducted by NIST, used the 34 events in the TDT2-Evaluation corpus. The results are based on tracking with $Nt = 1, 2$, and 4 relevant training documents, and the non-relevant documents that were provided by NIST for each event.

We compare approaches using the TDT evaluation methodology for tracking, which uses TDT2 cost as the target effectiveness measure (Section 3.2.1). Recall that the tracking methodology is based on novel use of the data, in which the same corpus is used for both training and testing. In what follows, if the system is being evaluated for four relevant training stories, that is, $Nt = 4$, then all stories in the stream up to and including the fourth relevant training story are considered the training corpus, and the testing corpus is the remainder of the stream. This methodology implies that different events effectively have different training and test corpora.

Static classifier formulation serves as our baseline system for evaluation. Classifiers are formulated for each event using the n most frequent nonstopwords from the relevant documents in the training data, where n is the pre-specified dimensionality of the classifier (Equation 4.3, page 49). The words are given weights using an assignment based on tf (Equation 4.1, page 48). The process is described in more detail in Section 4.1.2.1. We also experimented with feature selection and weight assignment

variants of the baseline process. We tested static classifiers expanded with multi-word features (MWF) [57]. We also tested two weight-learning algorithms: Dynamic Feedback Optimization (DFO) [10] and Exponentiated Gradient Descent (EG) [41].

In addition to comparing extensions to static classifier formulation, one of the goals of our tracking experiments is to determine the impact that automatic speech recognition (ASR) technology has on our process. We compare results using ASR data to those obtained from manual transcriptions of the same broadcast news sources, which were available in closed caption (CCAP) format.

4.3.1 Static vs. Adaptive Tracking

A comparison of our static and adaptive tracking approaches is listed in Table 4.3. The associated DET curves for tracking with four relevant training instances ($Nt = 4$) are depicted in Figure 4.5. The data indicate that adaptive classifiers were more effective than static classifiers in terms of lower story-weighted average cost at $Nt = 4$. The adaptive approach had lower cost for 9 topics and higher cost for 7 topics. However, the results in Table 4.3 suggest that the adaptive approach is less effective at lower values of Nt . With the exception of adaptive tracking at $Nt = 2$, both approaches showed similar cost on the closed caption (CCAP+NWT) source. This suggests that the impact of ASR on tracking was minimal.

Table 4.3. Story-weighted cost for static and adaptive tracking.

Type	Nt	ASR+NWT	CCAP+NWT
Static	4	0.0070	0.0066
Static	2	0.0069	0.0071
Static	1	0.0073	0.0081
Adaptive	4	0.0059	0.0064
Adaptive	2	0.0107	0.0075
Adaptive	1	0.0103	0.0122

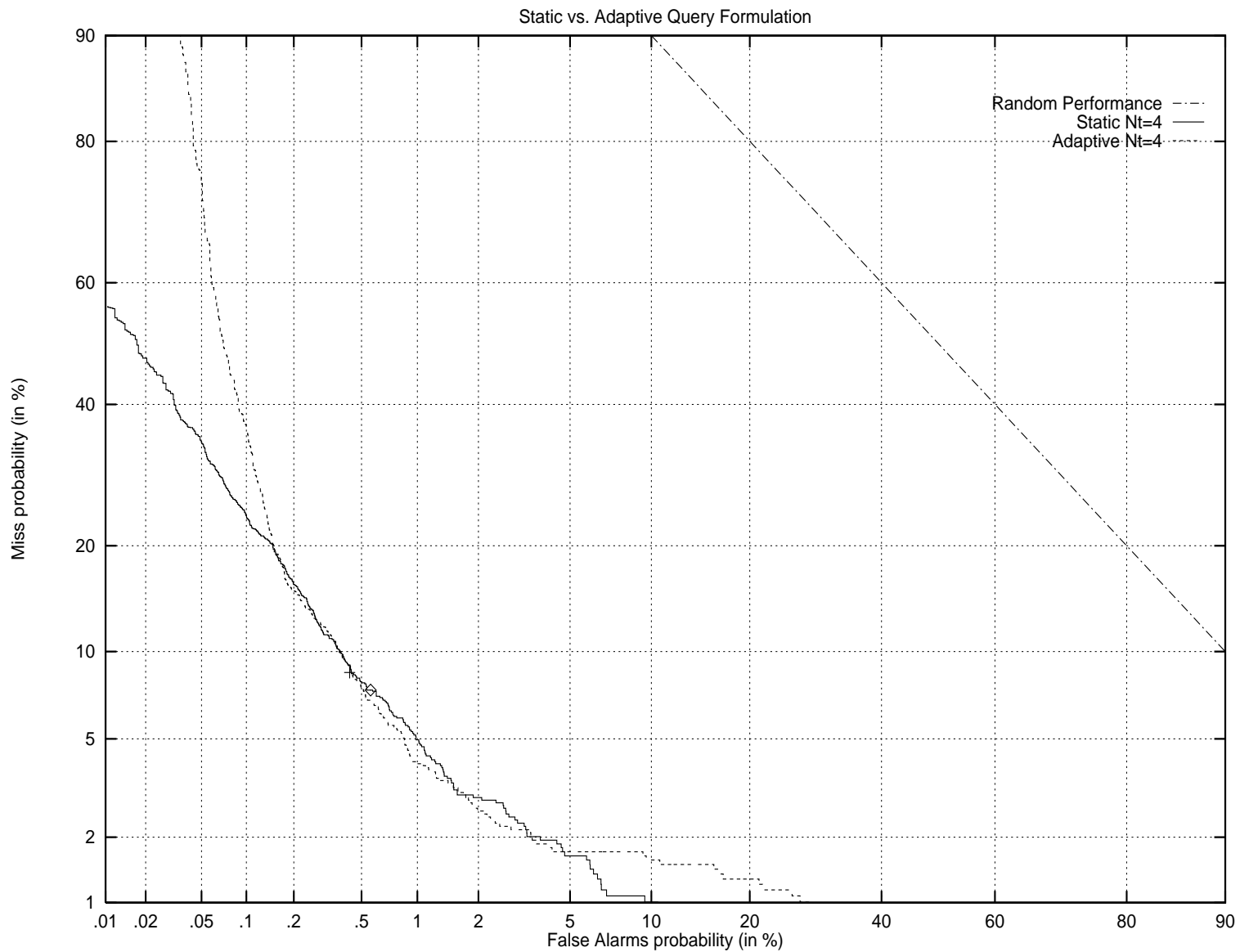
Our intention for the adaptive approach is to model the property of event evolution in broadcast news, the thrust being to include useful features in the classifier as they become available over time. The results presented here corroborate those we obtained on the TDT1 corpus [4], where the adaptive approach was significantly less effective at $Nt = 1$ than at $Nt \geq 4$. In addition, the data indicate that with only one relevant document, tracking is relatively effective using our baseline static classifier formulation process. This suggests that static and adaptive approaches should be combined in one system where the static approach is used for low values of Nt , and the adaptive approach is used when more training samples are available.

The DET graph in Figure 4.5 illustrates that at $Nt = 4$ the adaptive classifier curve is closer to the origin than the static classifier curve for false alarm rates between 0.2% and 5.0%; however, the static classifier curve has a more desirable error detection trade-off in general. The DET graph suggests that the adaptive approach is not more robust than the static approach using our representation, but that the adaptive approach leads to better hard decision classification effectiveness. Though not shown here, the static classifiers resulted in much lower (better) DET curves than their adaptive counterparts for both $Nt = 2$ and $Nt = 1$. This would be expected given the differences reported for the approaches in Table 4.3.

4.3.2 Multiword Features and Weight-Learning

We tested various extensions of static classifier formulation which have been hypothesized to significantly improve retrieval effectiveness over the baseline process described in Section 4.1.2.1. Multiword features (MWF) were found to improve tracking effectiveness in subsets of the training corpora. We used a process described by Papka and Allan [57] that expanded classifiers with Inquiry proximity operators of varying size windows of words. In the following experiments, we expanded 50-word

Figure 4.5. DET Graph: Static vs. Adaptive Tracking (ASR+NWT, $N_t = 4$).



queries with the top 10 pairs of words that appear in windows of size 5, 20, 50, and 100 words.

We also tested Dynamic Feedback Optimization (DFO) [10] and Exponentiated Gradient Descent (EG) [41]. These weight-learning algorithms adjust the classifier’s weights using supervised training techniques. The DFO algorithm was the same as that used by Schapire et al. [78]. This algorithm tweaks each classifier weight in turn, and recomputes average precision on the training data. If a weight change does not improve precision, the change is undone. The EG algorithm was a modification of the algorithm previously used for TREC filtering experiments [49, 56]. This learning technique is similar to other least-squared error reduction approaches used in Perceptron learning, but the weights are modified using an exponential function instead of a linear one. Classifier weights are adjusted based on pre-specified target values for relevant and non-relevant documents. We found the mean relevant and mean non-relevant similarity values from the distributions in the training sample to work well for target values.

In the following experiments, we explore these variations using $Nt = 4$ relevant training documents. The non-relevant documents were those supplied by NIST for the TDT2 evaluation. Before tracking begins, the static classifier is expanded using multiword features, or weights are modified using EG or DFO. We used the same threshold parameters as those in Table 4.1. The results are listed in Table 4.4.

Table 4.4. TDT2 cost for extensions to static tracking. (ASR+NWT, $Nt=4$).

Type	Story-Weighted	Percent Change	Topic-Weighted	Percent Change
Static	0.0070		0.0066	
Adaptive	0.0059	-15.7%	0.0074	12.1%
Static+DFO	0.0061	-12.9%	0.0067	1.5%
Static+EG	0.0070	0.0%	0.0080	21.2%
Static+MWF	0.0072	2.9%	0.0064	-3.0%

As with the adaptive approach, we found that expansion and weight-learning approaches did not improve overall effectiveness significantly. For the 21 events evaluated, improvements were minimal. The classifiers expanded with multiword features had lower cost for 10 events, but increased cost for 6. Further analysis of the weight-learning approaches revealed that the DFO algorithm decreased cost for 3 events, while increasing cost for 2 events. EG decreased cost for 1 event, and increased cost for 1 event.

The weight-learning approaches are of little use at $Nt \leq 4$ using our representation. We find that most classifiers and their thresholds already separate the training data, so not much improvement can be expected from weight-learning². But we observed that for higher values of Nt the occurrence of training data separation decreases. An analysis of the TDT2 train and development corpora, for example, revealed that 95% of the classifiers and thresholds formulated with 4 relevant training documents separated the training data. At $Nt = 16$ only 10% of the classifiers completely separated their training data. This suggests that weight-learning algorithms are more likely to be effective for higher values of Nt , and training data separation should be tested at lower values.

4.4 Cross-system Comparison of TDT2 Tracking Systems

The evaluation of the TDT2 tracking systems described in this section was the culmination of a set of experiment that took over 16 months to conduct. Participants trained and developed their tracking systems on the 85 events from the TDT1, TDT2-Train, and TDT2-Development corpora. The evaluation conducted by NIST was performed on 21 out of the 34 events of the TDT2-Evaluation corpus. (Events

²When the training data are separated by a classifier TDT2 cost on the training data is 0. Furthermore, no further improvements to Average Precision (used by DFO) can be obtained because the training data is perfectly sorted.

that contained fewer than four relevant training instances were excluded from the evaluation.) The goal of the experiment was to develop a system that minimized classification error in terms of pooled average (story-weighted) TDT2 cost (Equation 3.1, page 34). Classifiers were evaluated on their ability to track on the ASR source condition using four relevant training instances. Participants did not have the assessments for the events being evaluated; however, for each event, NIST provided an index listing the four relevant training samples and the non-relevant training instances up to the last relevant training instance. Systems were required to track events independently; that is, the assessments from different events were not permissible knowledge when tracking a particular event.

Several of the TDT participants provided runs for the tracking evaluation. We summarize the salient features of the BBN, CMU, Dragon, and UPENN tracking systems as follows:

BBN The BBN tracking system is based on formulating a mixture of classifiers from three models: Topic Spotting (TS), Information Retrieval (IR), and Relevance Feedback (RF). The first two approaches are based on a probabilistic approach to word occurrence distributions. The TS model assumes that words in the test story are generated by the model from the training stories; the IR model assumes that the training stories are generated by the model from the test story; and the RF approach used frequently occurring terms in the training stories. In their report, they also show improvements using an adaptive query formulation approach. [40]

CMU The group from Carnegie Mellon University tested Decision Trees (DT) and a K-Nearest-Neighbor (KNN) approach to tracking. In their DT approach, they used features in addition to word cooccurrence statistics including the location of a word relative to the beginning of the story, whether the root of the word appeared in the story, and an adaptive time window approach. The KNN

approach used a *tf-idf* document representation. Their analysis suggests that the KNN approach appeared slightly more effective than the DT approach. [15]

DRAGON Dragon Systems uses statistical approaches based on a beta-binomial model and a unigram language model. Their data suggest that a mixture of their approaches leads to improved tracking effectiveness. They also apply background models that are constructed from an auxiliary corpus. A document is considered relevant to an event if it is more similar to the model resulting from the training documents than to one resulting from a background model. [97]

UPENN The system from the University of Pennsylvania is based on a similar representation to the one we used for tracking. They used a *tf-idf* representation for classifiers and documents, where incremental-*idf* was seeded with the document frequencies from the TDT1 corpus. A cosine similarity function was used to compare classifier and document vectors. [79]

A summary of the TDT2 tracking evaluation for the systems described above is listed in Table 4.5. The process we used for our official TDT tracking submission was adaptive classifier formulation, which captures the property of event evolution by including new lexical features that become available over time in the news coverage of an event.

Table 4.5. NIST evaluation of TDT2 Tracking Systems (Nt=4).

	ASR+NWT		CCAP+NWT	
	SW	TW	SW	TW
UPENN	0.0058	0.0066	0.0056	0.0063
UMASS	0.0059	0.0074	0.0064	0.0065
BBN	0.0063	0.0056	0.0064	0.0059
DRAGON	0.0070	0.0069	-	-
CMU	0.0077	0.0076	0.0073	0.0072

Our adaptive and static classifier formulation approaches compared well to the systems of other participants based on low average cost. In general, the systems

had mixed improvements on the CCAP data, which suggests the impact of ASR for tracking effectiveness is minimal. A cross-system comparison using a sign test ($\alpha = 0.05$) suggests that the systems are not significantly different.

From the descriptions above and from our own comparison of tracking approaches, it becomes evident that *simplicity* led to lower story-weighted cost in the NIST evaluation. We believe that simplicity leads to parameter estimation stability. In other words, the more complex a system, the more parameters that need to be estimated, the less likely these parameters will be stable. For example, we found that the additional threshold parameter in our adaptive tracking approach made it more difficult to determine stable threshold parameters empirically, than when estimating the single parameter needed for our static tracking approach. For this reason we determined to explore the issues related to threshold parameter estimation, discussed in the next chapter.

4.5 Conclusion

The results in this chapter further our understanding of representations and techniques that work for on-line text classification. In real-world settings, a user will most likely provide very few relevant documents with which to formulate classifiers, and the experiments in this chapter suggest that many of the techniques that work well in the TREC environment are not effective in the tracking environment when few relevant documents are used.

In this chapter, we evaluated the representation we use for event classification in the context of the TDT tracking problem. In addition, we evaluated the effectiveness of several classifier formulation approaches that have been previously applied to TREC filtering. In particular, we tested extensions to our baseline static classifier formulation process with weight-learning and expansion steps that incorporate multiword features.

Our results suggest that our static classification formulation process is effective using one or more relevant training samples. Our adaptive tracking approach was effective for hard classification decisions using four or more relevant instances; however, the static approach appeared to be more stable.

We found insignificant effectiveness improvements using extensions of the static approach. Our data suggests that EG and DFO weight-learning approaches are of little use for tracking with few relevant training samples due to training data separation issues. The adaptive technique appeared to work well on the target evaluation condition ($Nt = 4$), but proved less robust than the static approach for $Nt < 4$. We recently completed experiments using $Nt = 8$ and $Nt = 16$ relevant training instances applied to the weight learning and adaptive approaches. As with the lower values for Nt , these approaches provided no significant improvements in tracking effectiveness over the static classifier formulation process.

We also compared our approaches to other systems participating in the TDT2 tracking evaluation, and our approaches were effective in comparison to the best systems. Since the classifiers we formulate with one relevant document are effective, it suggests that our text representation for tracking may be effective in a solution to clustering and new event detection. We explore this hypothesis in later chapters.

CHAPTER 5

THRESHOLD ESTIMATION BIAS IN TEXT CLASSIFIERS

In this chapter, we discuss processes that find effective thresholds for text classifiers using the representation we presented for event tracking in Chapter 4. Our approach to selecting thresholds for classifiers is based on estimating a threshold from the relevant and non-relevant documents provided as training data. We estimate a classifier’s threshold by finding a value that separates relevant from non-relevant documents while optimizing the effectiveness measure used for evaluation. The optimization process using few relevant instances has a tendency to overestimate thresholds with respect to the optimal value to use when tracking. In particular, we find the estimate from the optimization process to be more biased towards the training set when fewer relevant instances are available.

There are several biases inherent in classification systems. Mitchell identifies two types of bias, which are known as *inductive bias* and *estimation bias* [54]. Inductive bias refers to the bias inherent in the classifier resulting from a particular algorithm or representation used to solve a classification problem. For example, certain decision tree algorithms may have a bias towards producing trees that are relatively short and wide. Other decision tree approaches may consistently result in trees that are tall and narrow.

Estimation bias refers to the error resulting from an estimate of the value of a random variable. For example, when we test our estimations for parameters derived from training data on the same data, the expected effectiveness is optimistic and is

biased towards the training data. The cross-validation and bootstrap methodologies described in Section 3.2.2 attempt to reduce this type of bias.

Another bias that has been observed is *sampling bias*, which is caused by the way samples are selected from a population. For example, Rosenkrantz [71] retells the following experiment:

In a class exercise, a population of rocks of varying sizes was spread out on a large table so that all the rocks could be seen. After inspecting the rocks and lifting any that they wished, students were asked to select samples of five rocks such that their average weight would estimate as accurately as possible the average weight of the whole population of rocks. The average weights of the students' samples consistently overestimated the population mean.

In this case, the overestimation was possibly caused by the fact that “[a] larger and heavier rock is more easily noticed than a smaller one” [71]. Hence, the students had a tendency to select a larger rock first, which led to an upward bias in the estimation. Similarly, a certain amount of sampling bias exists in the random selection of events for the TDT corpora, since the final events selected were more likely to be events discussed in many documents.

In the following sections, we are primarily interested in the estimation bias associated with estimating a threshold for classifiers used by our tracking system. Bias in this context is the difference between the estimated threshold and that which would give rise to optimal effectiveness on the testing data. In what follows, we use the text representation described in Chapter 4 to illustrate bias that results from using an optimization process for estimating the threshold of a classifier used for tracking. We demonstrate that bias for a threshold estimator that uses this approach is affected by the number of relevant documents used in formulating the classifier. We assume that

the estimation of a threshold is biased due to factors in the training and optimization processes, and we attempt to identify the amount of bias inherent in an initial threshold and adjust for this amount in the final estimate of the classifier’s threshold.

5.1 Threshold Estimation

The *threshold estimation problem* is to determine a threshold for a classifier from training data that works well on the testing data. The problem, in the context of our system, is depicted in Figure 5.1. The data in the figure resulted from a classifier formulated for the “Crash of US Air Flight 427” event using eight relevant training instances. The distribution of similarity values between the classifier and the relevant and non-relevant training instances is shown in the top graph. The goal of the threshold estimation process is to find the similarity value that will separate documents relevant to the crash from non-relevant ones. In our case, the threshold should be one that minimizes TDT2 cost for the documents on the subsequent stream. The distribution of similarity values for the classifier, when applied to the documents for the testing portion of the stream, results in a similarity value distribution represented in the bottom graph of Figure 5.1.

One thresholding methodology relatively common in text classification involves an optimization process combined with a ranked-retrieval process. This process finds the score for documents in a sorted list of training instances in which the relevant documents are separated from the non-relevant documents by maximizing a target utility function [56, 3, 100]. However, our results for on-line processing using few relevant training instances suggest that it is more effective to lower an estimate based on an optimized threshold.

Recently, the TREC routing task has evolved into a filtering task where evaluation is based on binary classification. Several approaches to determining thresholds have emerged and are described by Hull [38]. In his analysis of TREC-6 filtering systems,

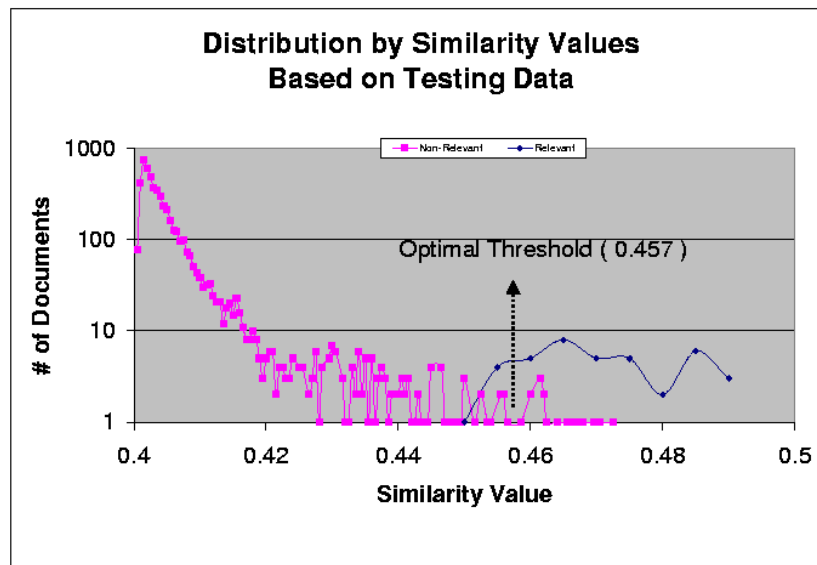
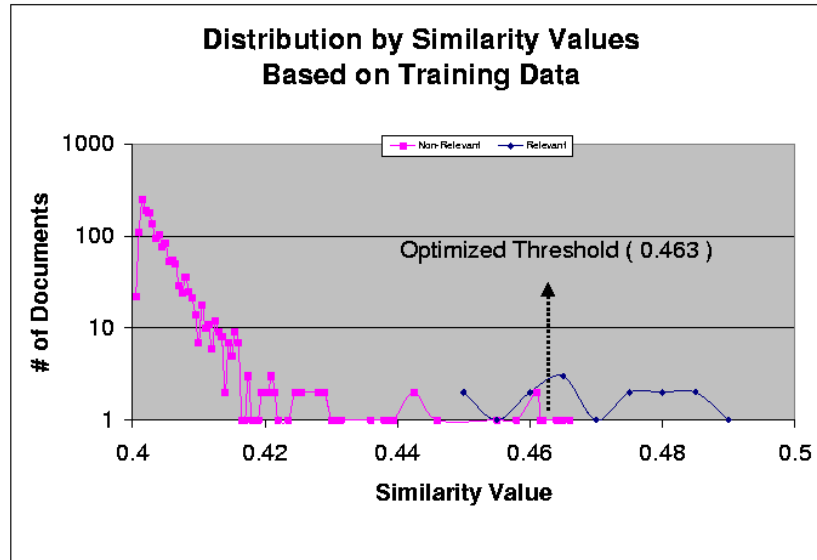


Figure 5.1. Threshold Problem: Crash of US Air Flight 427

Hull demonstrates that the bias in a threshold estimation algorithm is dependent on the utility measure. In this analysis he computed the average number of TREC topics

for which each system’s estimation for a threshold was the same, lower, or higher than the threshold that would have resulted in optimal effectiveness on the testing data. In general, the optimal threshold for the testing data is normally distributed across topics for effectiveness measures that are a function of miss and false alarm rates. However, a different skew emerges for thresholds resulting in optimal average-set-precision, which is the product of recall and precision.

Optimization of the utility function as part of the threshold estimation process is important regardless of the approach used for text classification. For example, Lewis [47] (reproducing earlier categorization experiments of Maron) found a classifier’s threshold using the minimum estimated probability of class membership. His data indicate that different probability thresholds give rise to different levels of recall and precision, where one is traded for the other as the minimum value used for the threshold is decreased. This implies that different thresholds are required for different utility measures that are a functions of recall and precision. For a real application, therefore, the user’s utility preferences would need to be incorporated in order for document classification to be effective.

Statistical inference techniques reported by Lewis and others build upon a Bayesian model for text classification. This approach is used in a maximum likelihood estimator model described by Mitchell [54], and also by Robertson [69] for his TREC-7 filtering system. Probabilities based on word-cooccurrence statistics are measured and binary classification in favor of relevance is assumed when the estimated probability that a document d is relevant to a topic is greater than the probability that it is not relevant i.e., when $P(G_1|d) > P(G_2|d)$, where G_1 is the group of relevant documents and G_2 is the group of non-relevant documents for the topic.

One of the problems of using this approach is that an arbitrary utility function is not necessarily maximized or minimized when $P(G_1|d) > P(G_2|d)$. For example, if user utility implies a strong aversion to false alarms, then it would be better to make

decisions based on a function such as $P(G_1|d) > P(G_2|d) + c$, where c is a positive constant. An appropriate value for c would insure that $P(G_1|d)$ is large enough before deciding the document is relevant. Other techniques for optimizing a particular utility function in a Bayesian framework have been described by Casella and Berger [19]. In their approach, a matrix of weights that represents the spectrum of user preference is part of the estimation process.

5.2 Learning Threshold Estimator Bias

In this section, we describe two automatic threshold estimation algorithms for the document tracking task. The methods we discuss produce bias-reduced estimators resulting in improved classification accuracy for tracking. We define and illustrate threshold estimator bias using our representation described in Chapter 4.

The threshold methodology that we use involves an optimization process combined with a ranked-retrieval process. The optimization step is to find the classifier/document similarity score s , that maximizes utility in the sorted list of training instances returned by the ranked-retrieval engine. In the experiments below, we estimate a threshold u for each query with estimator $\hat{u} = 0.4 + \theta * (s_{optimized} - 0.4)$, where 0.4 is an Inquiry constant, θ is a global system parameter, and $s_{optimized}$ is the similarity value resulting from the classifier that, when applied to the event’s labeled training documents, optimizes the target TDT2 cost function defined by Equation 3.1. Threshold estimator \hat{u} was used for the tracking experiments in the previous chapter.

From several experiments using the 85 events that were available from the TDT1, TDT2-Train, and TDT2-Development corpora, it was determined that when fewer relevant training documents were used, $s_{optimized}$ (our estimator \hat{u} when $\theta = 1.0$) was consistently above the parameter u it was trying to estimate, that is, the optimal threshold for the unprocessed stream of data for a particular event, or simply $s_{optimal}$,

was overestimated using $s_{\text{optimized}}$. In what follows, we assume that the quantity of estimator bias $b(\hat{u}) = E[s_{\text{optimal}}] - s_{\text{optimized}}$, which is similar to the definition of bias used to analyze classification effectiveness [54]. We attempt to learn threshold estimator bias over varying numbers of relevant training documents (Nt) and varying numbers of classifier features.

5.2.1 The Histogram Method

The histogram method, which utilizes threshold estimator \hat{u} , is illustrated with 50-feature classifiers in Figure 5.2. Using the TDT1, TDT2-Train, and TDT2-Development corpora as training data, histograms of optimal values for θ were collected for each value of Nt and for classifiers formulated with the top 10, 20, 50, 100, 200, 600, and 10000 words in the relevant documents (Section 4.1.2.1). Using this method on the training data, for example, determined that for one relevant document ($Nt = 1$) and 50-feature classifiers, 48 out of 85 classifiers had optimal cost when $\theta = 0.2$. For each pair of Nt and number of features, we calculate $E[\theta]$ from the corresponding histogram, and use the resulting value for θ when estimating thresholds on the TDT2-Evaluation corpus.

The data in Figure 5.2 illustrates estimator bias when $Nt \leq 16$ relevant training documents are used. The data suggest that as Nt increases $E[\theta]$ increases. Also, as $E[\theta]$ increases, on average, $s_{\text{optimized}}$ is closer to s_{optimal} ; thus less total estimator bias results when more relevant training examples are used. We also observed similar but less significant increases in bias when more features were used.

5.2.2 Linear Regression Method

The observation that increasing training instances reduces the bias of an estimator, in general, is not surprising. James, for example, shows that estimates move toward the true population values when training instances are increased for data assumed to have multivariate-normal distributions [39]. An explanation of the phenomenon

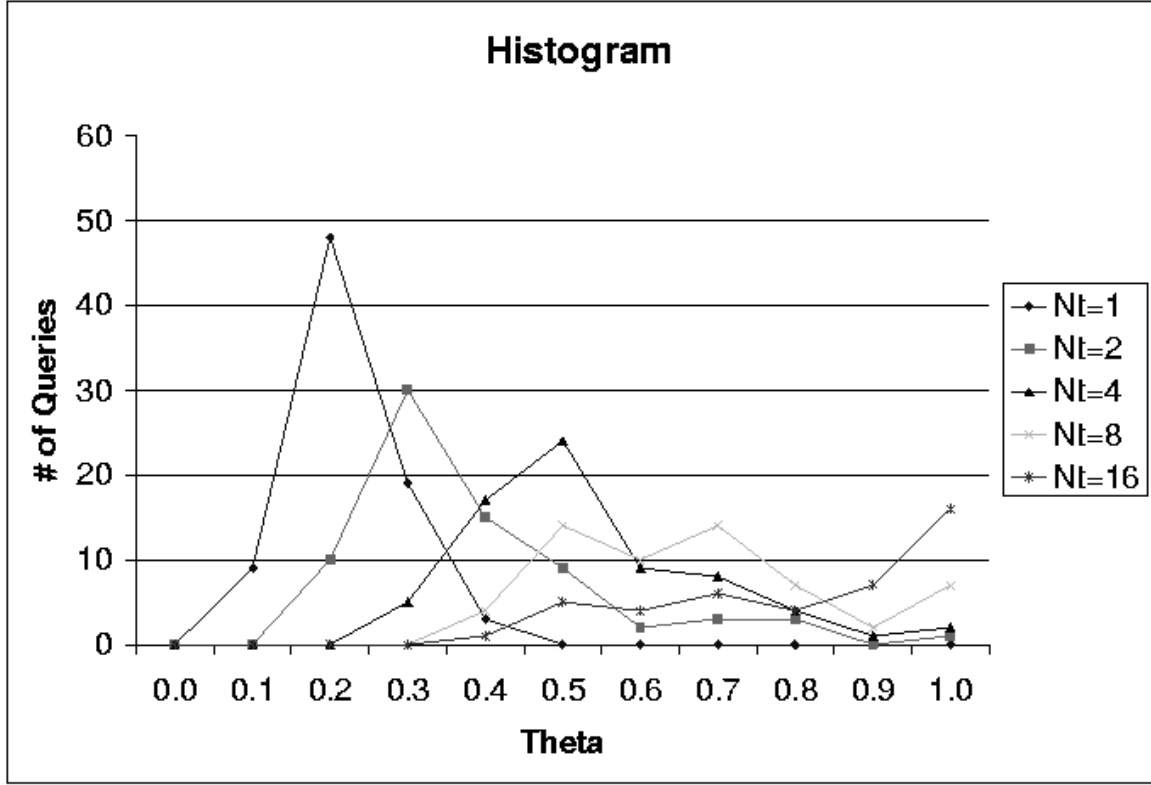


Figure 5.2. Histograms of optimal threshold parameter θ for varying Nt (50 features).

follows from the law of large numbers. However, James also proposes that once the bias is found, it may be possible to reduce it using a linear transformation.

We test James's theory in the following experiments. We define a threshold estimator \hat{o} , such that $\hat{o} = s_{optimized}$. We then define a new bias-reduced threshold estimator \hat{e} , such that $\hat{e} = m\hat{o} + b$. We then compare the effectiveness of the query thresholds produced by our original estimator \hat{u} , which is bias-reduced using near optimal values for θ , to those produced by \hat{e} , with parameters m and b learned through

linear regression. In what follows, we use linear regression to learn query bias for estimator \hat{o} .

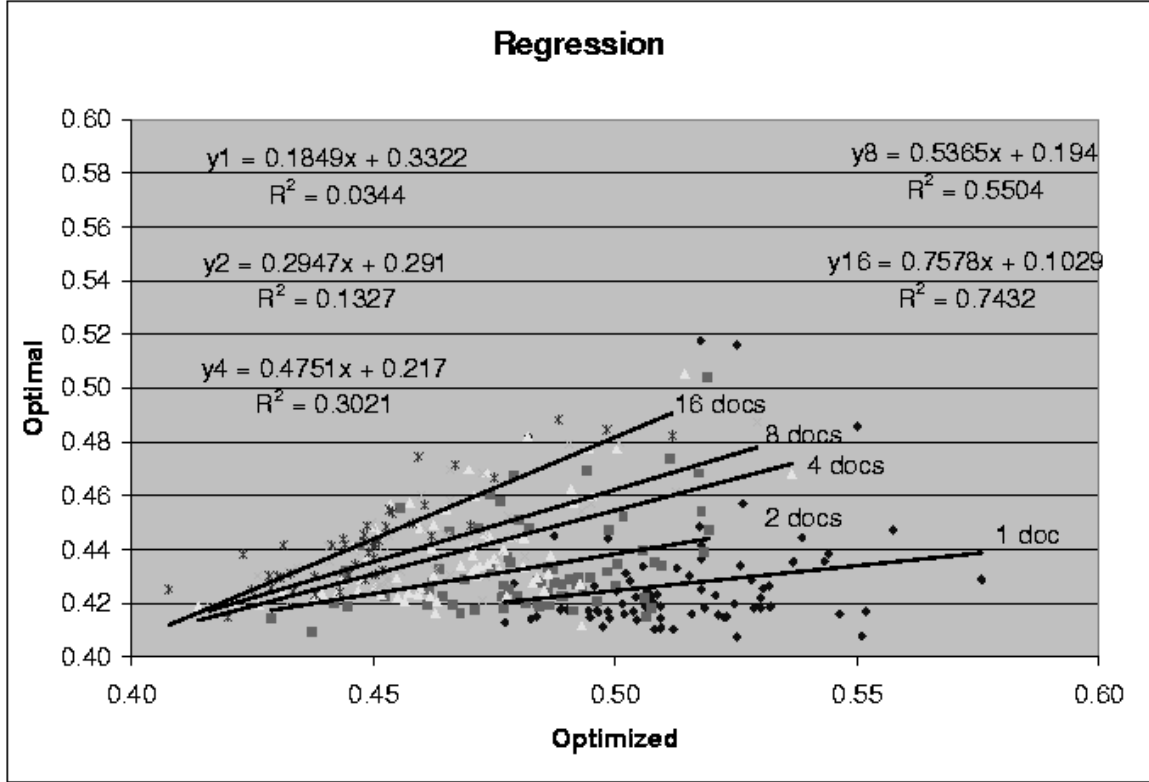


Figure 5.3. Regression of optimal threshold parameter θ for varying Nt (50 features).

The linear regression method for estimator \hat{e} is illustrated in Figure 5.3. Instead of collecting histograms, points represents by $s_{optimized}$ and $s_{optimal}$ are fitted using a line for each value of Nt and number of features. The slopes and intercepts of lines produced by the various regressions are subsequently used as parameters m and b for estimator \hat{e} during evaluation. This method learns the same tendencies in threshold estimator bias as the histogram approach for estimator \hat{u} . As the number of relevant training documents increases, the slope of the resulting regression line approaches 1.0.

As the slope approaches 1.0, $s_{optimized}$ approaches $s_{optimal}$, which implies that higher values of Nt give rise to less estimator bias in the training data.

5.2.3 Comparison of Methods

From the tracking experiments discussed in Chapter 4, we knew the histogram approach for estimator \hat{u} generalized and provided relatively low cost on the TDT2-Evaluation corpus. In the following experiment, we compare the effectiveness of threshold estimator \hat{e} to that of \hat{u} using the same static classifier formulation and tracking processes on the evaluation corpus. The results, which are listed in Table 5.1, illustrate the percent increase in cost resulting from tracking with estimator \hat{e} instead of estimator \hat{u} .

Table 5.1. Percent of cost increase from replacing estimator \hat{u} with estimator \hat{e} .

# of Features	Number of Relevant Training Documents (Nt)				
	1	2	4	8	16
10	8.2%	-20.4%	-13.2%	-21.9%	-2.9%
20	27.5%	-17.0%	-32.1%	-23.4%	-1.6%
50	5.0%	53.2%	0.0%	-4.0%	-8.6%
100	1.1%	12.7%	4.5%	-6.0%	-20.3%
200	3.3%	-7.5%	-8.2%	2.0%	-16.4%
600	27.8%	-6.0%	-7.3%	-8.0%	-1.8%
10000	27.8%	-6.0%	-7.3%	-8.0%	0.0%
Average	14.4%	1.3%	-9.1%	-9.9%	-7.4%

In Table 5.1, an increase in cost implies a decrease in classification effectiveness, hence the data suggest that, on average, the histogram method works better for $Nt = 1$ and marginally so for $Nt = 2$. However, for $Nt > 2$ the regression method appears to reduce cost consistently for most of the query sizes tested. The correlation coefficients (R^2) that are calculated for the regressions depicted in Figure 5.3 indicate that using fewer documents led to lower correlation than using more documents. This trend was evident across the varying number of classifier dimensionalities that were tested. This suggests that the resulting regression was not a good fit for $Nt \leq 2$.

However, the improvements realized by the regression method suggest a mixture of approaches where estimator \hat{u} is used for $Nt \leq 2$ and estimator \hat{e} otherwise.

5.3 Bias in Relevance Assessments

Our discussion of classifier bias would not be complete without addressing the biases that are inherent in the relevance assessments that were used for the experiments described in this thesis. This type of bias is a function of the assessor’s background knowledge for a particular event. Throughout the development phases of TDT, some participants reported that they found documents that were incorrectly judged for a particular event. Few of the problems reported were accepted as errors, and most, in fact, were not errors, but simply reflected a participant’s subjective view of the event.

One example of this was uncovered by our new event detection system and the event related to Warren Buffet’s recent manipulation of the silver market. Mr. Buffet decided to buy 5% of the world’s silver supply through the metal’s underlying futures instrument, which sent the price of silver skyrocketing. Based on the available judgements for this event, our system missed the one document judged to contain discussion of the new event. From the failure analysis we found that the system decided this event was discussed in an earlier document about an FTC probe into the recent volatility in the metals market. Our familiarity with the financial markets suggested that the document about the FTC probe is very much related to the actions of Buffet, and thus we felt our system correctly detected the new event. The evaluation software told us otherwise. One question that arises is how much this type of bias can affect our system’s relative effectiveness.

The cause of the differences that were observed between assessors of the TDT data could be explained by the following factors [31]:

1. Assessors A and B have conflicting *a priori* knowledge and simply view document content differently.

2. An assessor made a keystroke error while using the document-assessment system.

The real error made by the human is of the second type, while the first type implies that the degree to which an assessor feels that a particular document is relevant to an event is dependent on the type of event and the interpretation the assessor makes based on a description of the event.

Voorhees observed that the probability was very small for two systems to change rank in a cross-system comparison of ad hoc retrieval, due to bias in relevance assessments [91]. In the evaluation of TDT2 detection and tracking systems, the set of assessments went through three iterations of cleanup that resulted in no change in the relative rank of any system. We therefore conclude that the bias inherent in the relevance assessment process has a minimal impact on the overall relative effectiveness of document classification systems.

However, the bias in the judgements must impact classification effectiveness to some extent. We hypothesize that an upper bound on system classification effectiveness is the overlap of the agreement between assessors. In addition, we posit that this agreement should be greater for events than more general topics, because the information request for the former can be more specifically defined. The analysis of the data from the assessment process and the results that are reported for TDT and TREC text classification suggest that these hypotheses are true. In TDT, approximately 90% of the documents were judged identically by multiple assessors [20]. This would suggest an upper bound for F1-measure to be 0.9, and the TDT tracking systems appear to produce average F1-measures in 0.7 to 0.85 range. In Voorhees' analysis of TREC data [91], only 50% of the assessments for the TREC topics analyzed had the same assessment from multiple assessors. It is very likely to see F1-measures reported for the TREC filtering problem that are in the 0.35-0.45 range for these data. If these hypotheses are in fact true, that would suggest tracking and filtering

systems are already very close to upper bound effectiveness, and that constructing text classifiers from sample documents has many very effective solutions.

5.4 Conclusion

We described two approaches for automatic threshold parameter estimation for our static classifier formulation process applied to the problem of event tracking. We view the threshold as a statistic of the incoming news stream that is estimated using an optimization process applied to the training data. We defined the notion of classifier bias in terms of threshold estimators, and illustrated that the amount of bias increases when fewer relevant training samples are used on the TDT data. Our results suggest that our automatic thresholding approaches can learn the bias and result in effective estimates of threshold parameters for classifiers. We suggest using the histogram approach when estimating thresholds for $Nt \leq 2$ relevant instances, and the linear regression approach otherwise.

The histogram approach, in turn, leads us to the threshold estimation methodology that we use for new event detection and event clustering. In this approach, the parameter θ that controls the threshold model is set to the value that, on average, gives rise to the best overall tracking effectiveness for the events tested during development. Our approach to new event detection and event clustering is based on formulating text classifiers for each document appearing on the stream. In the context of tracking, our approach to these problems is to formulate tracking classifiers for the entire stream using $Nt = 1$ relevant training documents. In the next chapter, we discuss how we adapt the histogram method for event clustering.

CHAPTER 6

CLUSTERING

In this chapter, we extend our text and classifier representation for tracking to the more general problem of event clustering, that is, the unsupervised process of grouping stories discussing the same event. We focus on solutions to the on-line clustering task, where a story is assigned to a cluster before processing subsequent stories on the stream. We extend our classification model to exploit the temporal relationship between stories. The motivation for this approach is that news stories appearing on the stream closer in time are more likely to contain discussion of the same event than stories appearing further apart.

In the next section, we review our event clustering approach, which is based on the implementation of our tracking system described in Chapter 4. Our implementation for clustering is described in Section 6.2, and a review of the clustering strategies we tested appears in Section 6.3. In Section 6.4, we discuss retrospective experiments and our optimization effort used for predictive experiments. We evaluate our approaches using the data and evaluation methodology developed for the Topic Detection and Tracking (TDT) research initiative. In TDT, the problem of clustering is referred to as the *detection task*, which is described in more detail in a Section 6.5. We also present clustering approaches from other systems that participated in the recent TDT2 detection task evaluation.

6.1 On-line Clustering Algorithms

In what follows, we use a single-pass (incremental) approach to event clustering (Section 2.3.2). When a new document appears on the stream, it is either put into one of the existing clusters, or becomes the seed of a new cluster.

There are three main issues to consider when implementing an on-line clustering algorithm:

1. partitioning
2. prototyping, and
3. comparison strategies.

The partitioning issue for clustering involves simply whether a document can appear in more than one cluster. A process creates a partition of the stream if each document appears in only one cluster.

When cluster prototyping is used, the documents in each cluster are represented by one classifier per cluster. A prototype for a cluster can be created by formulating a single classifier from the documents in the cluster, or averaging the feature weights of multiple classifiers. Prototyping potentially reduce the number of document comparisons, and may lead to more effective groupings than using a separate classifier for each individual document. However, we found prototyping did not improve effectiveness in several previous experiments using a clustering approach to new event detection [3]; therefore, in what follows, we focus on non-prototyping methods to clustering. In addition, several of the TDT participants have pursued uses of cluster prototyping and centroids, and the comparison of approaches suggests that prototyping does not lead to improved effectiveness over our approach.

The comparison strategy is what places a document in a particular cluster or not. The strategy also determines if the document is a seed for a new cluster. In the experiments to follow, we re-evaluate cluster comparison strategies that have been

used for agglomerative hierarchical clustering (Section 2.3.1). We test three commonly studied comparison strategies, namely single-link, group-average, and complete-link strategies, which have been previously applied to retrospective clustering in which all the documents are assumed to be available before clustering begins [89, 22, 90, 94, 75]. The names for these strategies are associated with hierarchical clustering, and we use them here in the context of on-line single-pass versions. The distinction is that single-pass clusters are dependent on the order in which documents are processed, while the hierarchical clustering approach produces the same set of clusters regardless of document order.

6.2 Implementation

Our basic algorithm for event clustering is the following: For each document we formulate a fixed-length classifier from the n most frequent words in the document excluding stopwords. This process is the same as our static classifier formulation process for tracking (Section 4.1.2.1) using $Nt = 1$ relevant training documents. The classifier’s initial threshold is its similarity value when compared to the document from which it was created. We assume no subsequent document will exceed this threshold, and so we use it as an initial estimate for the threshold. We re-estimate the threshold using our threshold model, which is discussed in the next section. As new documents arrive on the stream, they are compared to previously formulated classifiers, and clusters are formed based on a particular comparison strategy. In Section 6.3, we describe the strategies we tested for event clustering.

6.2.1 Time-based Threshold Model

A side-effect of the temporality of broadcast news is that documents closer together on the stream are more likely to discuss related events than documents farther apart on the stream. When a significant new event occurs, there are usually several

documents per day pertaining to it; over time, coverage of old events is displaced by more recent events. Figure 6.1, for example, depicts the number of documents per day that arrived on CNN broadcast news and AP newswire pertaining to two events: an earthquake in Japan and the downing of a US Airforce pilot over Bosnia. Most of the documents pertaining to these events occurred within 10 days, and this appears to be the pattern for many of the events in the TDT corpora. Therefore, one hypothesis about the domain of broadcast news is that exploiting the time between documents will lead to improved classification accuracy.

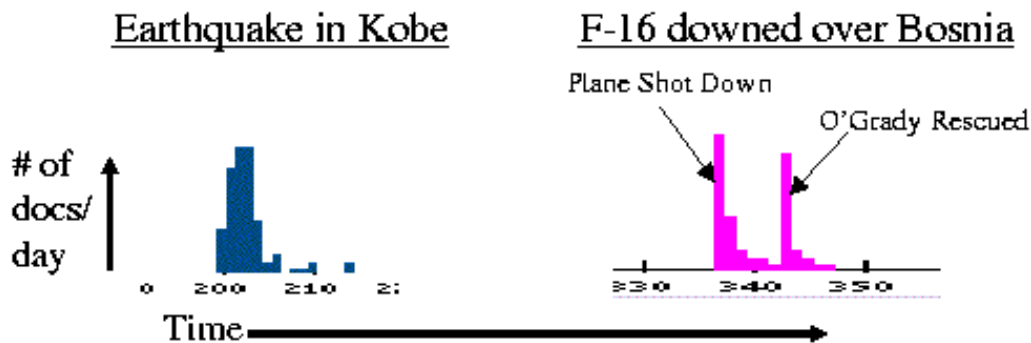


Figure 6.1. Daily document counts for two TDT events.

In the implementations that follow, time is incorporated into our threshold model. The thresholding technique we use for clustering is an extension of the methods we used for tracking in Chapters 4 and 5. The threshold model defined by Equation 4.7 (page 51) is extended with a linear function that controls similarity based on the number of days between the formulation of the classifier and the arrival of the document. During processing, a classifier’s actual threshold is potentially recomputed at each time step, that is, each time a new document arrives on the stream. For any classifier formulated at time i , its threshold for a document arriving at a later time j is

$$threshold(q_i, d_j) = 0.4 + \theta * (sim(q_i, d_i) - 0.4) + \beta * (date_j - date_i), \quad (6.1)$$

where $sim(q_i, d_i)$ is the similarity value between the classifier and the document from which it was formulated (Equation 4.6, page 51), and the constant 0.4 is an Inquiry parameter. The value of $(date_j - date_i)$ is the number of days between the arrival of document d_j and the formulation of classifier q_i , and we use the global system parameter β to control the effects of this value. The values for θ and β control classification decisions, and our method for finding appropriate settings are discussed later in this chapter.

6.2.2 Decision Scores

Our threshold model described above is what decides for each classifier whether a document is a positive instance. Our confidence in the decision that document d_j is a positive instance of classifier q_i is the extent to which the document exceeds the classifier's threshold. In what follows, the score we used between a document and a classifier is

$$decision(q_i, d_j) = sim(q_i, d_j) - threshold(q_i, d_j). \quad (6.2)$$

This was our original decision score for tracking described in Section 4.1.3. A decision scores greater than 0 implies that d_j is a positive instance of q_i , and it also implies that documents d_i and d_j are similar. In many clustering approaches, the similarity between d_i and d_j is symmetric. Since the documents and classifiers have different representations using our approach to clustering, the similarity between documents is not necessarily symmetric.

6.3 On-line Cluster Comparison Strategies

The decision scores resulting from Equation 6.2 are used to calculate *comparison values* between documents and clusters. There are several ways to combine decision scores resulting from a document compared to the classifiers in each cluster. In this section, we discuss the way decision scores are combined using the cluster comparison strategies we tested.

When on-line clustering begins, a classifier for the first document on the stream is formulated and becomes the first cluster. The classifier formulation process (Section 4.1.2.1) is repeated for the second document, but immediately afterwards, a *comparison value* is determined between the second document and the cluster resulting from the first. If the second document exceeds the classifier's threshold then the classifiers from the first and second document are merged into one cluster; otherwise, the classifier from the second document initiates a new cluster. Once a cluster contains more than one classifier, a policy must be invoked which determines the comparison value between the document and the cluster.

In this dissertation, we re-evaluate document clustering based on on-line single-pass variants of single-link, group-average, and complete-link cluster comparison strategies. These strategies are described in more detail in 2.3.3, and we summarize our approaches as follows:

- In the *on-line single-link* strategy, the comparison value is the maximum positive decision score for the classifiers contained in a cluster. The classifier from the current document initiates a new cluster if it does not result in a positive comparison value for any existing clusters; otherwise it is placed in the cluster that has the highest decision score.
- In the *on-line average-link* strategy, the comparison value for each cluster is the averaged decision score for the classifiers contained in a cluster. If the decision score average is negative, then the document is not assumed to have similar

content to the documents associated with the cluster. If the average decision score is negative for all existing clusters, then the document under inspection initiates a new cluster, otherwise it is placed in the cluster that has the highest average decision score.

- We also implemented an *on-line complete-link* strategy, where the comparison value for each cluster is the minimum decision score of the classifiers in a cluster. If the minimum decision score is negative for all existing clusters, then the document under inspection initiates a new cluster, otherwise it is placed in the cluster that has the highest minimum score.

Based on our implementation, we did not find our on-line complete-link strategy to be effective in comparison to other strategies, and therefore we focused on comparing on-line average-link to on-line single-link strategies. Our implementation of the on-line average-link strategy required more processor time than the on-line single-link implementation. For example, clustering with on-line single-link ran in a few minutes, while average-link strategy required a few hours for each experiment. In general, all three approaches have the same asymptotic running time of $O(n^2)$, that is, the current document is compared to the classifiers resulting from all the previously processed documents. However, in practice, it is sufficient to evaluate only those classifiers that have lexical features that cooccur in the currently processed document; therefore, all three approaches require much fewer than n^2 comparisons. In our implementation, the on-line average-link strategy was slower because it required the additional step of determining the comparison values for *all* clusters, whereas the on-line single-link strategy required determining a comparison value for one cluster, i.e., the one containing the classifier resulting in maximum decision score.

6.4 On-line Clustering Experiments

In this section, we test our text classifier approach to event clustering and evaluate the on-line single-link strategy and the on-line average-link strategy. In addition, we test the effectiveness of using the time component in the threshold model described by Equation 6.1. When the time component is used, then $\beta > 0$ in the equation, and its effects on running time performance is minimal. From several preliminary experiments, we found that the on-line average-link strategy did not benefit from values of $\beta > 0$, and thus the time component is only reported for the on-line single-link strategy. When the time component is used with the on-line single-link strategy, we refer to the strategy as *on-line single-link+time*.

In what follows, we evaluate effectiveness using a methodology developed by TDT. Effectiveness is measured based on the best cluster that resulted for each of the known events, where *best* is the cluster that contains the most stories for each event. In what follows, we measured recall, precision, and F1-Measure (Section 3.2.1) and found parameters that led to optimal F1-measure for classifiers of length 25, 50, 100, and 200. We use F1-measure for these experiments, and report results optimizing for TDT cost function in the next section.

The optimization process involves finding appropriate values for θ and β for our threshold model described by Equation 6.1. Our approach is to increment and test a range of values for each parameter and dimensionality setting to find optimal effectiveness over the events with known judgements. In the experiments that follow, we used the automatic speech recognition (ASR) versions of the TDT1, TDT2-Train, and TDT2-Development corpora to find parameters, and we used the ASR version of the TDT evaluation corpus and its events for predictive experiments. Results comparing the on-line single-link, single-link+time, and average-link strategies are listed in Figures 6.2 to 6.4 below.

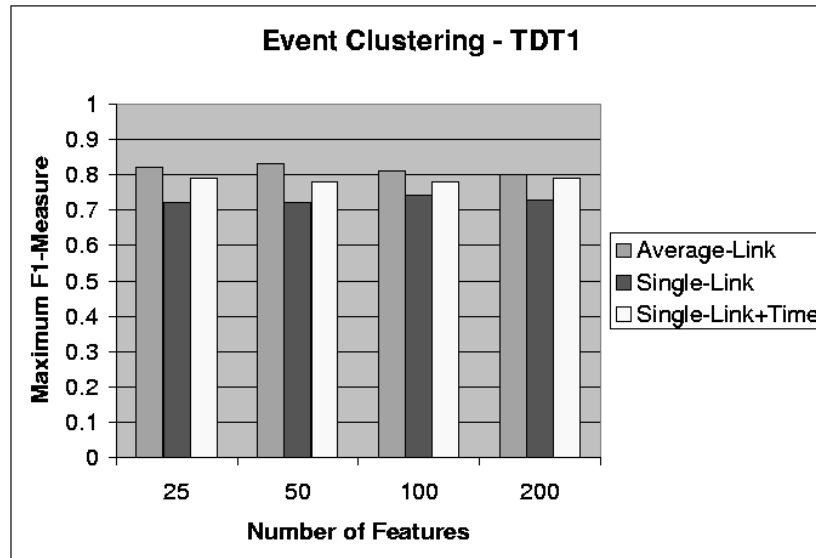


Figure 6.2. Event Clustering: TDT1 corpus.

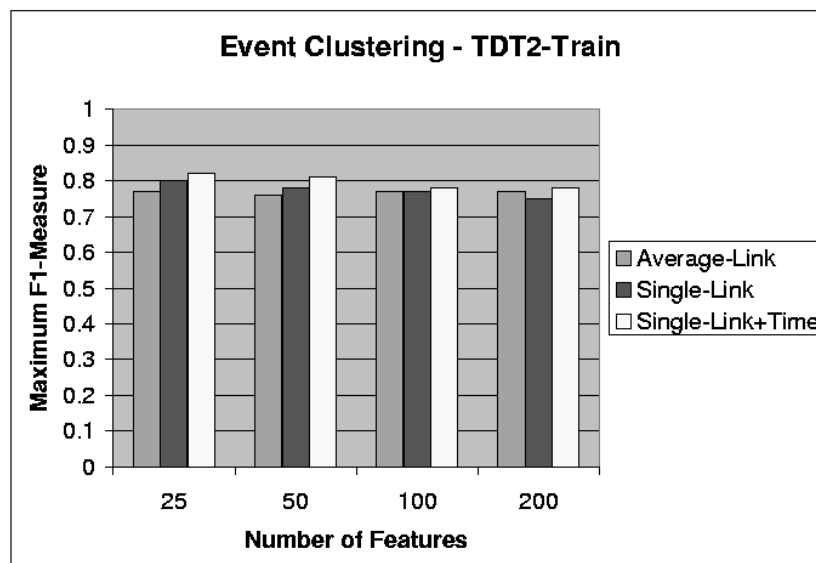


Figure 6.3. Event Clustering: TDT2-Training corpus.

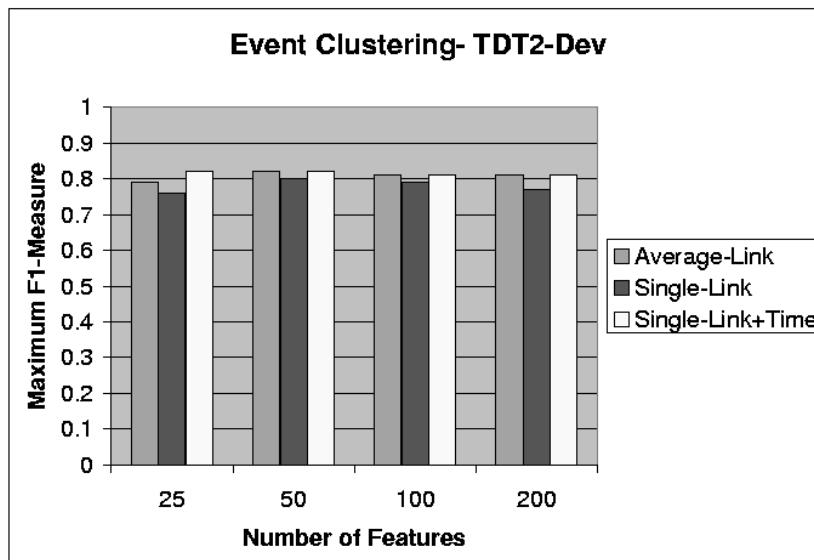


Figure 6.4. Event Clustering: TDT2-Development corpus.

The charts in Figures 6.2 to 6.4 illustrate story-weight (pooled) F1-measure resulting from using the best parameters we found for each corpus. We see that at optimal parameters, the on-line average-link strategy is comparable in effectiveness to on-line single-link+time. Both of these strategies appear to be more effective than the on-line single-link strategy in which the time component of the threshold model is not used. In addition, no particular classifier dimensionality appears to give rise to a significant increase in effectiveness. For example, on the TDT1 corpus using 50-feature classifiers, the on-line average-link approach had an optimal F1 of 0.81, and using 200 features, the on-line single-link+time strategy had optimal F1 of 0.79. These measures are among the best currently reported for on-line clustering on this corpus [98, 5].

Using the on-line single-link strategies on the TDT2-Train, and TDT2-Development corpora, we found that our approaches were affected by the headers and trailers con-

tained in broadcast news programs. The relevance assessments for these corpora indicated that most of these snippets were tagged as miscellaneous text, which is excluded from evaluation but not from processing. For example, the trailer: “up next, your local weather,” results in a classifier with the lexical features ‘local’ and ‘weather’. This small classifier will have high similarity to documents that have even a few occurrences of either feature, and will thus lead to classification errors. In an effort to mitigate this problem, we analyzed the lengths of documents marked miscellaneous in the TDT2-Train and TDT-Development corpora, and determined that documents containing less than 55 words are very likely to be headers or trailers. We therefore classified documents under this size as non-relevant to all events, and assigned them to one cluster not used for comparisons.

Our approach to selecting thresholds for event clustering was to use the parameters that on average gave rise to optimal pooled average F1-measures in the retrospective experiments. This is similar to the histogram approach for tracking discussed in Chapter 5. For on-line single-link+time we set $\theta = 0.26$ and $\beta = 0.001$; for on-line single-link we set $\theta = 0.33$ and $\beta = 0$; and for average-link $\theta = 0.15$ and $\beta = 0$. The results from predictive experiments on the TDT2-Evaluation corpus are listed in Table 6.1.

Table 6.1. Detection results on TDT2-Evaluation corpus.

On-line Strategy	F1-Measure		TDT2 Cost	
	SW	TW	SW	TW
single-link+time	0.61	0.71	0.0056	0.0062
single-link	0.54	0.71	0.0077	0.0072
average-link	0.58	0.68	0.0108	0.0062

In Table 6.1, pooled or story-weighted (SW) measures as well as the mean or topic-weighted (TW) measures are reported for both F1 and TDT2 cost. The on-line single-link+time approach appears to cluster events more effectively than the on-line single-link strategy, which suggests that using the time component of the threshold

model is effective. In addition, the single-link+time approach appeared to be the most effective in terms of pooled F1-measure and even more so in terms of pooled TDT2 cost. However, the average-link approach resulted in the same topic-weighted cost as the single-link+time approach, which suggests that both methods are effective for event clustering. In the following section, we report our TDT2 detection results, where we applied the same parameter estimation process but optimized for story-weighted cost instead of F1-measure. In Section 6.6, we compare our results to those of other event clustering systems.

6.5 TDT Detection Experiments

In TDT, on-line event clustering is referred to as the *detection task*. The stream is processed with or without a deferral period (DEF) that specifies the number of news programs that can be processed before making document/cluster decisions. A score and cluster number are produced for each document before the next document is read, or before the deferral period ends. Effectiveness measures are recorded for the best cluster that resulted for each of the known events, and for TDT, *best* is the cluster that minimizes the TDT2 cost function defined by Equation 3.1 (page 34). The cost function is a linear combination of miss and false alarm rates. In this section, we discuss our results for predictive experiments using the TDT methodology.

The effectiveness measures in this section were produced by the National Institute of Standards and Technology (NIST). We had access to the first three corpora and event assessments described in Tables 3.1 and 3.2 (page 33). The TDT2-Evaluation corpus was used for the evaluation in which we did not have the relevance assessments for the events being evaluated.

As part of our TDT efforts, we tested on-line single- and average-link comparison strategies. We used the same optimization approach as described above, and found values for θ and β that yielded optimal story-weighted cost using the ASR versions

of the TDT1, TDT2-Train, and TDT2-Development corpora. We used the mean of the parameters across the three corpora, and applied them to the TDT2-Evaluation corpus for both automatic speech recognition and manual closed caption transcription (CCAP) versions of the evaluation corpus. The newswire sources (NWT) were the same for both versions.

We found different parameters for each strategy when optimizing for cost instead of F1-measure. In the experiments that follow, for the on-line single-link+time approach we set $\theta = 0.22$ and $\beta = 0.001$, for single-link $\theta = 0.3$ and $\beta = 0.0$, and for average-link $\theta = 0.1$ and $\beta = 0.0$. We found that at optimal parameter settings, lower cost was obtained for some events using more features, but for other events, fewer features were more effective. In the experiments that follow we used 50-feature classifiers, which appeared to work well on the TDT2-Train and TDT2-Development corpora for both on-line single- and average-link approaches.

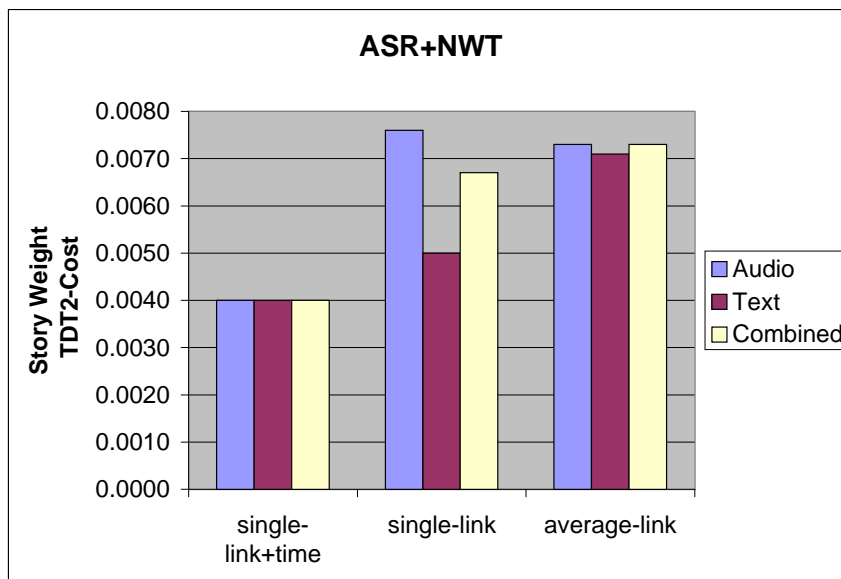


Figure 6.5. TDT Detection: TDT2-Evaluation corpus (Story-weighted cost for ASR+NWT source).

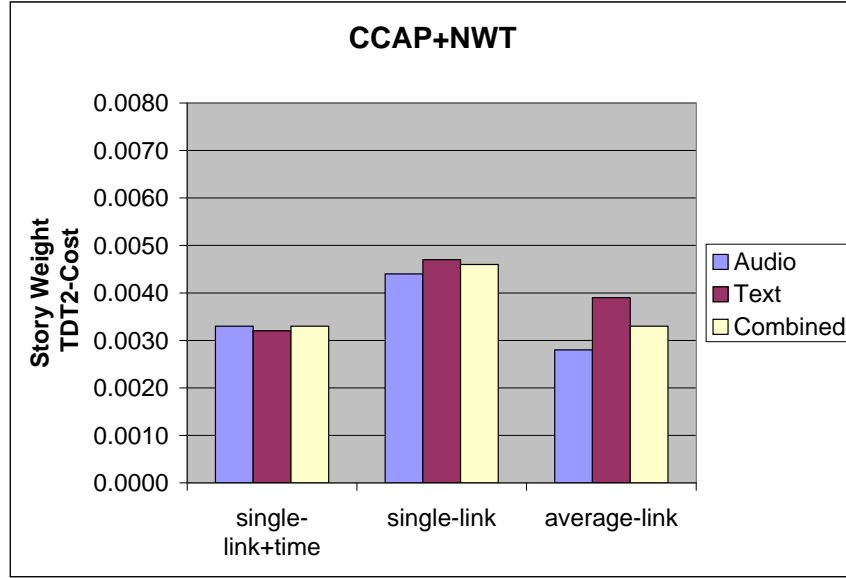


Figure 6.6. TDT Detection: TDT2-Evaluation corpus (Story-weighted cost for CCAP+NWT source).

The data from our predictive TDT detection experiments are listed in Figures 6.5 and 6.6. The charts include separated story-weighted TDT2 cost for audio (ASR or CCAP) and text (NWT) testing documents. The data suggest that the on-line single-link+time strategy is the most effective clustering approach for both audio and text sources when applied to the ASR+NWT version of the TDT2-Evaluation corpus. The other approaches appeared to benefit from the CCAP source (Figure 6.6), in which on-line average-link provided equally low cost as on-line single-link+time. In contrast to our tracking results, all the clustering approaches we tested were more effective on the CCAP source than on the ASR source, which suggests that the ASR technology impacts the effectiveness of event clustering.

The results from the experiments above are summarized in Table 6.2. As expected, optimizing for story-weighted cost instead of F1-measure lead to lower cost with respect to the values we listed in Table 6.1. In addition, the average-link ap-

proach appeared to be more effective for topic-weighted (TW) evaluation, which is the average TDT2 cost across events.

Table 6.2. TDT Detection: TDT2-Evaluation corpus.

On-line Strategy	ASR+NWT		CCAP+NWT	
	SW	TW	SW	TW
single-link+time	0.0040	0.0064	0.0033	0.0059
single-link	0.0068	0.0077	0.0047	0.0058
average-link	0.0074	0.0055	0.0033	0.0044

When we analyzed the clusters resulting from the TDT2 evaluation for each strategy, the most salient distinction observed was the number of clusters the different strategies produced. After processing the ASR+NWT stream, which contained 22,445 documents using the boundaries provided, the on-line average-link strategy produced 1221 clusters. However, the on-line single-link strategy produced 8206, and on-line single-link+time produced 7515 clusters. The results using the closed caption source (CCAP+NWT) indicated roughly the same trend in that the on-line single-link strategies produced nearly 6 times as many clusters as the on-line average-link strategy. However, the CCAP+NWT results in Table 6.2 indicate that the on-line single-link+time strategy has the same clustering effectiveness as the on-line average-link strategy despite the large difference in the number of clusters produced. These results suggest that the evaluation methodology, which is based on the best cluster for each event, is not sensitive to stories about the same event appearing in many clusters. In TDT3, new evaluation measures for the detection task are being considered that may provide more insight into the quality of the resulting clusters.

6.6 Cross-System Comparison for TDT2 Detection Problem

In this section we discuss the results from the TDT2 detection task evaluation. In what follows, we compare our results to those from other TDT research sites. We restrict our comparison to sites that provided data to NIST prior to the TDT2

submission deadline. The detection systems were developed independently, and the overall competitive-collaborative effort resulted in several corroborated results from different approaches to the detection problem.

A common element across systems was the use of a word-cooccurrence model. Some systems used a *tf·idf* text representation similar to the one described in this dissertation, while others made use of statistical language modeling approaches. We summarize the salient features of the BBN, CMU, Dragon, IBM, and UPENN detection systems as follows:

BBN The BBN system used an incremental k-means algorithm. They experimented with probabilistic document similarity metrics and more traditional vector-space metrics. They show that combining these metrics is effective for detection. As documents are processed and added to previous clusters, parameters are re-estimated and cluster centroids are reformulated. They also tested the use of a deferral period for which they found no improvement in terms of story-weighted cost. [92]

CMU The group from Carnegie Mellon University used incremental agglomerative clustering that successively merged clusters using a group-average comparison approach. They used the SMART retrieval system and a *tf·idf* document representation. The experimental focus for their system was to test the effects of using a deferral period. The deferral period gives rise to a lookback technique where comparisons are limited to a fixed size window of time. They show small gains from using a deferral period. [15]

DRAGON Dragon Systems describes their detection system in terms of a Beta-Binomial Mixture Model. This is a language modeling approach where they calculate the probability distribution for a word appearing a given number of times in a document of a particular length. As the documents are processed,

the parameters for the distribution of each word are re-estimated. They use distributions from a relatively small lexicon of between 20K and 60K words. They compare their binomial model to a more expensive multinomial model which results in a small improvement for average TDT2 cost. [51]

IBM The IBM system used a Vector Space clustering approach. Several preprocessing steps were applied to each document before comparing it to existing cluster centroids. These steps included part-of-speech tagging, stemming, and feature extraction of unigram and bigram adjacent nouns. They used a *tf·idf* document representation, and compared documents to cluster centroids using a similarity measure from the Okapi retrieval system. [27]

UPENN The system from the University of Pennsylvania is based on a single-link agglomerative clustering approach using a *tf·idf* representation for documents. A cosine similarity function was applied to vectors of document weights. Unlike the other systems, they did not use stemmed lexical features. [79]

Tables 6.3 and 6.4 list the official results reported by NIST for the TDT2 detection task. The target source condition was the ASR version of the evaluation corpus, and we report both story- and topic-weighted TDT2 cost. The TDT sites had access to the first three corpora and event judgements described in Tables 3.1 and 3.2 (page 33). The TDT2-Evaluation corpus and set of events was used for the evaluation, and the sites did not have the relevance assessments for the events before the submission deadline. The TDT cost function was used for evaluation, and the target condition was based on DEF=0, and thus, cluster decisions were made for each document before reading the next.

In general, the effectiveness reported for all the systems is very good. Many of the systems that used *tf·idf* weights also used an explicit time component or incremental-*idf* calculation, which we have found to have a similar property to a time component

caused by decreasing *idf* weights as more of the stream is processed. It is therefore not surprising that overall effectiveness was not significantly different between these systems. However, it is surprising that the on-line single-link approaches worked as well as agglomerative and average-link approaches in light of past observations by Willet [94] and Voorhees [90] who have found that single-link clustering is less effective than average-link for cluster-assisted retrieval. However, both on-line single-link and average-link methods appear to be effective for the TDT2 detection task.

Table 6.3. NIST evaluation of TDT2 detection task: TDT2-Evaluation corpus (ASR+NWT).

System	ASR+NWT		Description
	SW	TW	
BBN	0.0040	0.0047	Incremental K-Means
UMASS	0.0040	0.0064	Single-Link+Time
DRAGON	0.0045	0.0048	Beta-Binomial Mixture Model
IBM	0.0046	0.0042	Agglomerative
UPENN	0.0070	0.0063	Single-Link
UMASS	0.0073	0.0055	Average-Link (not official)
CMU	0.0077	0.0057	Incremental Agglomerative

Table 6.4. NIST evaluation of TDT2 detection task: TDT2-Evaluation corpus (CCAP+NWT).

System	CCAP+NWT		Description
	SW	TW	
UMASS	0.0033	0.0059	Single-Link+Time
UMASS	0.0033	0.0044	Average-Link (not official)
BBN	0.0034	0.0043	Incremental K-Means
CMU	0.0068	0.0049	Incremental Agglomerative

The TDT2 systems were more effective using the CCAP version of the evaluation corpus than the ASR version. A comparison of the results from Table 6.4 to those of Table 6.3 suggest that clustering is more effective when using the cleaner closed caption data.

As previously mentioned, evaluation for each event is based on the one cluster that gives rise to minimal TDT2 cost. This best-fit approach hides cluster fragmentation

issues. Recall that our single-link+time approach produces roughly 7500 clusters, while BBN reported that its system produced roughly 3000 clusters for the same corpus. While these different clusterings result in the same story-weighted cost, the utility of returning several thousand clusters to a user is questionable. More experiments are necessary to determine which approach is producing the correct granularity of clusters, and those experiments would likely require exhaustive relevance judgments. However, we believe that document clustering is likely to be a more useful component technology when it is part of a solution to a user-oriented classification task such as new event detection.

6.7 Conclusion

In this chapter, we tested approaches to on-line event clustering using a text classifier approach to the problem. The data suggest that the on-line single-link+time cluster comparison strategy is more effective than other approaches when automatic speech recognition data is clustered. When cleaner closed caption transcriptions are used, the on-line average-link strategy appears to be as effective as on-line single-link+time. We believe the on-line single-link+time strategy works well because it captures the periodicity and overall temporal relationship between news stories that may not exist in collections from other information domains. Our results suggest that modeling the temporal relationship between documents is useful when clustering broadcast news data.

We also found that our optimization approach using both F1-measure and TDT2 cost resulted in effective classification. This approach was used for the tracking problem, where we searched for optimal parameters across training and development corpora, and used the mean of these values when performing predictive experiments. When we applied the approach to the TDT detection task, our overall implementation compared favorably to other TDT systems for both the ASR+NWT

and CCAP+NWT source conditions. In addition, the TDT detection results suggest our text representation and classification approach was effective.

The question that arises is whether clustering can help new event detection. In the context of clustering news, the problem of new event detection is to find the first story in each cluster, in other words, the stories that become cluster seeds. The results from this chapter suggest that, on average, our approach to on-line event clustering puts most of the stories about an event in one cluster. In the next section we evaluate on-line clustering as a basis for new event detection, and determine how well different comparison methodologies produce cluster seeds.

CHAPTER 7

NEW EVENT DETECTION

The problem of *new event detection* is to identify the stories in a stream of news that contain discussion of a new event, that is, an event which has not been previously reported. In this chapter, we present on-line solutions to new event detection in which the system indicates whether the current news story contains or does not contain discussion of a new event before processing the subsequent story. Our approach to new event detection is based on event tracking with one relevant document, which was discussed in Chapters 4 and 5. In addition to newswire, we apply our approaches to the domain of broadcast news, which includes data from television and radio transmissions. In what follows, we describe the details of our algorithm and experimental results using the corpora and evaluation methodology developed as part of the Topic Detection and Tracking (TDT) research initiative.

The motivation for our approaches to the problem is to incorporate the salient properties of broadcast news. In particular, we identify the property of *time* as a distinguishing feature of this domain. We posit that modeling the temporal relationship between documents should result in improved classification. Our event clustering results in Chapter 6 suggested that this hypothesis is true, and we showed that our approach to event clustering is more effective when this temporal relationship is modeled.

Another property of news is that its content includes the names of the people, places, dates, and things, i.e., the *who*, *what*, *when*, and *where* that are the focus of an event. Our intuition is that the words in proper noun phrases are important to

include in a classifier when using a word-cooccurrence model for text classification. We test this intuition by augmenting the classifier formulation process described in Section 4.1.2.1 with a natural language parser that finds proper noun phrases in each document. Our results suggest that identifying these phrases leads to improved classification.

There are many lexical cues in language that indicate a new event, which we do not model, but which are used by humans. For example, when you are asked if the document in Figure 4.2 (page 49) discusses a new event, you may be inclined to say “yes” because of your interpretation of the words in the document. You might also be correct in saying “no” if you know that the event was covered by an earlier broadcast.

Simulating the way humans interpret novelty in news is a provocative concept; however, it may not necessarily lead to more accurate detection than the system we have implemented. For example, a human trying to detect new events over a significant period of time would begin to forget the details of each event, because the brain does not store a representation for every snippet of news seen and heard.¹ Nonetheless, humans have the ability to generalize, and in order to detect new events in text, a representation with the property of generalizing to unseen documents is necessary.

In what follows, we use the same representation we used for tracking, and the same classification model we used for event clustering. The contents of each document is formulated into a classifier which is compared to subsequent documents. Our approach to new event detection is that if no classifier comparison results in a positive classification decision for the current document, then the current document has content not previously encountered, and thus it contains discussion of a new event.

¹Case studies by experimental psychologists reveal a few anomalies where the opposite is true, that is, the subject has a memory in which an object is identifiable only if that exact object has been seen before, but does not appear to have the cognitive ability to generalize [76]. This type of memory is easier to model mathematically than one with the capacity to generalize.

The main difference between our approach to new event detection and event clustering is that the emphasis is placed on finding the start of each event, not grouping documents by events.

7.1 New Event Detection Algorithm

We have developed a single-pass algorithm for new event detection using the text representation described in Chapter 4. Our implementation is based on the query syntax and ranked-retrieval engine of the Inquiry retrieval system [12]. Our algorithm processes each new document on the stream sequentially, as follows:

1. Formulate a classifier representation for the document's content.
2. The new classifier's initial threshold is its similarity to the document from which it was formulated.
3. Re-estimate existing threshold over time using appropriate values for constants θ and β in the threshold model.
4. Compare the new document against existing classifiers in memory.
5. If the document does not result in a positive decision score with any existing classifier, flag the document as containing a new event.
6. If the document results in a positive decision score, flag the document as not containing a new event.
7. (Optional) Add the document to the document lists of the classifiers for which it had a positive decision score.
8. (Optional) Reformulate the existing classifiers using the updated document lists.
9. Add the new classifier to memory.

We represent the content of each document, which we assume discusses some event, as a classifier. If any existing classifier results in a positive classification of the current document, the document is assumed to discuss the event represented by the classifier; otherwise the current document contains a new event. We discuss the implementation details of the algorithm below.

7.2 Implementation

The classifier formulation process is the following: For each document we formulate a fixed-length classifier from the n most frequent words in the document excluding stopwords. This process is the same as our static classifier formulation process for tracking (Section 4.1.2.1) using $Nt = 1$ relevant training documents. We have found that classifiers formulated from one document are relatively effective for tracking, and that the formulation approach also works well using different comparison strategies for event clustering.

During processing, each classifier's threshold is potentially recomputed when each new document arrives on the stream. We use the threshold model from our event clustering implementation, which incorporates a time component that increases thresholds over time. For any classifier formulated at time i , its threshold for a document arriving at a later time j is

$$threshold(q_i, d_j) = 0.4 + \theta * (sim(q_i, d_i) - 0.4) + \beta * (date_j - date_i), \quad (7.1)$$

where $sim(q_i, d_i)$ is the similarity value between the classifier and the document from which it was formulated (Equation 4.6, page 51), and the constant 0.4 is an Inquiry parameter. The value of $(date_j - date_i)$ is the number of days between the arrival of document d_j and the formulation of classifier q_i . The values for θ and β control

new event classification decisions, and our method for finding appropriate settings are discussed later in this chapter.

We use decision scores when deciding whether a new event has arrived. If any decision score is positive as a result of comparing the current document to an existing classifier, then we assume the document does not discuss a new event. In what follows, the score we used between a document arriving at time j when compared to an existing classifier q_i formulated at an earlier time i is

$$decision(q_i, d_j) = sim(q_i, d_j) - threshold(q_i, d_j), \quad (7.2)$$

which is the same decision score we used for event clustering in Chapter 6. Decision scores greater than 0 imply that documents d_i and d_j are similar in content, and thus document d_j does not discuss a new event.

7.3 New Event Detection Experiments

In this section, we discuss the results we obtained from the algorithm described in Section 7.1. For these experiments, we developed our system using data from the TDT1, TDT2-Train, and TDT2-Development corpora, and we evaluated our system on the TDT2-Evaluation corpus (Section 3.1). We report values based on TDT2 cost and F1-measure, which are described in more detail in Section 3.2.1. In the following experiments we optimized for F1-measure in order to compare these data to those we obtained in previous experiments [5, 4].

The results in this section are based on the evaluation methodology for new event detection developed for the TDT Pilot Study (Section 3.2.3). In the next section, we discuss our methodology for parameter estimation. In Section 7.3.2, we evaluate the effectiveness of using event clustering as an approach to new event detection, and in Section 7.3.4, we evaluate the use of proper nouns in our classifier formulation process.

7.3.1 Parameter Estimation Experiments

Our approach to parameter selection was to use the means of the parameters that gave rise to optimal effectiveness during training and development. We found the parameters that gave rise to optimal new event detection effectiveness over the events with known judgements in the TDT1, TDT2-Train, and TDT2-Development corpora, and we used the mean of these parameters when evaluating predictive experiments.

Using classifiers with different numbers of features led to different parameter settings, and we found that the parameters were most stable using 50-feature classifiers. In what follows, we used $\theta = 0.2$ and $\beta = 0.0005$ when time is factored into the threshold model, and $\theta = 0.3$ and $\beta = 0$ otherwise. The corpora and events that were used for development appeared to have optimal effectiveness at the same β and, on average, the same θ .

The effectiveness using 50-feature classifiers and the parameters we found in our optimization process described above are listed in Tables 7.1 - 7.4 below. The results for the TDT2 corpora in these tables are based on using newswire and automatic speech transcription sources (ASR+NWT).

Table 7.1. New Event Detection: TDT1 corpus using 50-feature queries.

skip	# of Docs	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
0	1124	40%	1.27%	60%	52%	0.56	0.0205
1	1099	44%	1.12%	56%	54%	0.55	0.0197
2	1074	48%	1.14%	52%	50%	0.51	0.0208
3	1051	57%	1.17%	43%	45%	0.44	0.0227
4	1028	41%	1.19%	59%	52%	0.55	0.0199
5	1006	64%	1.22%	36%	40%	0.38	0.0247
6	984	50%	1.25%	50%	48%	0.49	0.0222
7	962	50%	1.06%	50%	50%	0.50	0.0204
8	942	53%	1.08%	47%	47%	0.47	0.0211
9	923	68%	1.00%	32%	40%	0.35	0.0234
10	904	78%	0.90%	22%	33%	0.27	0.0244
Pool	1008	53%	1.13%	47%	48%	0.47	0.0217
Mean		54%	1.13%	46%	47%	0.46	0.0218

Table 7.2. New Event Detection: TDT2-Train corpus using 50-feature queries.

skip	# of Docs	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
0	3969	46%	3.20%	54%	13%	0.21	0.0405
1	3934	65%	3.21%	35%	9%	0.14	0.0444
2	3900	52%	3.31%	48%	10%	0.17	0.0427
3	3869	40%	3.36%	60%	12%	0.20	0.0409
4	3839	63%	3.39%	37%	8%	0.13	0.0459
5	3809	55%	3.31%	45%	9%	0.16	0.0434
6	3780	52%	3.25%	48%	10%	0.16	0.0422
7	3753	65%	3.25%	35%	7%	0.12	0.0449
8	3727	54%	3.43%	46%	9%	0.15	0.0444
9	3701	58%	3.40%	42%	7%	0.13	0.0450
10	3677	58%	3.53%	42%	7%	0.12	0.0463
Pool	3814	55%	3.33%	45%	9%	0.15	0.0436
Mean		55%	3.33%	45%	9%	0.15	0.0437

Table 7.3. New Event Detection: TDT2-Development corpus using 50-feature queries.

skip	# of Docs	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
0	599	60%	1.74%	40%	50%	0.44	0.0291
1	574	40%	1.81%	60%	55%	0.57	0.0257
2	554	42%	2.06%	58%	50%	0.54	0.0286
3	535	44%	1.93%	56%	50%	0.53	0.0278
4	517	61%	1.60%	39%	47%	0.42	0.0279
5	499	50%	1.44%	50%	50%	0.50	0.0241
6	485	36%	1.49%	64%	56%	0.60	0.0217
7	471	71%	1.53%	29%	36%	0.32	0.0293
8	457	69%	1.80%	31%	33%	0.32	0.0315
9	444	54%	1.39%	46%	50%	0.48	0.0244
10	431	58%	2.15%	42%	36%	0.38	0.0327
Pool	506	53%	1.73%	47%	48%	0.47	0.0275
Mean		53%	1.72%	47%	47%	0.47	0.0275

Tables 7.1 - 7.4 list the effectiveness of our approach using 50-feature classifiers across the 11 passes through the corpus as described in Section 3.2.3 above. Recall that a skip value of n implies that relevant documents 1.. n were removed from the stream for each event, and the goal was to detect the $(1 + n)$ -th document for each

Table 7.4. New Event Detection: TDT2-Evaluation corpus using 50-feature queries.

skip	# of Docs	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
0	1312	65%	1.72%	35%	35%	0.35	0.0298
1	1278	54%	1.76%	46%	37%	0.41	0.0280
2	1250	76%	1.71%	24%	22%	0.23	0.0320
3	1225	65%	1.91%	35%	26%	0.30	0.0318
4	1202	71%	1.86%	29%	21%	0.24	0.0325
5	1181	60%	1.64%	40%	30%	0.34	0.0280
6	1161	61%	1.40%	39%	30%	0.34	0.0259
7	1143	78%	1.60%	22%	18%	0.20	0.0312
8	1125	64%	1.53%	36%	23%	0.28	0.0279
9	1111	64%	1.55%	36%	23%	0.28	0.0280
10	1097	71%	1.85%	29%	17%	0.21	0.0324
Pool	1189	66%	1.69%	34%	26%	0.30	0.0297
Mean		66%	1.68%	34%	26%	0.29	0.0298

event. Hence, a skip value of 1 implies that the second document was the goal, and so on. Pooled averages are based on responses across the available events. The pooled average is story-weighted in the sense that events with more relevant documents are more heavily weighted. The effectiveness measures are stable for the first few skip values, but become worse at higher values because fewer events are included in the pass.

The effectiveness reported in the first row of these tables (*skip* = 0) is for the actual task of new event detection. We summarize these results in the Table 7.5 below.

Table 7.5. Cross-Corpora Comparison of New Event Detection

Corpus	# of Events	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
TDT1	25	40%	1.27%	60%	52%	0.56	0.0205
TDT2-Train	35	46%	3.20%	54%	13%	0.21	0.0405
TDT2-Dev.	25	60%	1.74%	40%	50%	0.44	0.0291
TDT2-Eval.	34	65%	1.72%	35%	35%	0.35	0.0298

The data suggest that we find between 35%-60% of the documents discussing new events at relatively low false alarm rates of 1.27%-3.2%. The predictive experiment on the TDT2-Evaluation corpus is listed in the fourth row. The data indicate that our parameter estimation process led to an F1-measure that was within the range of those obtained on the TDT2-Train and TDT2-Development corpora.²

The effects of modeling the temporal relationship between documents in our threshold model are illustrated in Figure 7.1³. Each point represents effectiveness at a particular combination of θ and β (Equation 7.1), obtained from threshold parameter searches on the TDT1 corpus.

In Figure 7.1, the points in the graph that are closer to the origin reflect higher classification accuracy. The points connected by a line represent effectiveness of the threshold model when $\beta = 0$, i.e., when time is not modeled. However, there is always a parameter setting for θ , where $\beta > 0$ is more effective. In general we found that across the corpora effectiveness is optimal when the time component was used in the threshold model.

7.3.2 Clustering Approaches to New Event Detection

The approach we use for new event detection does not cluster documents in any way. In our algorithm, a document contains a new event if the document does not result in a positive decision score with respect to any of the existing classifiers using Equation 7.2 (page 106). It is evident; however, that our new event detection algorithm is similar to the *on-line single-link+time* strategy for event clustering that

²The TDT cost measures in Tables 7.1 - 7.5 are worse than those obtained by a heuristic that decides that no document contains a new event. For example, this simple approach would yield an F1-measure of 0.0, but a cost of 0.0200 using the TDT2 cost function parameters applied to Equation 3.1 (page 34). It should be noted that the cost function parameters were intended for evaluating TDT tracking and detection data, and that different parameters are currently being considered in TDT3 to address this issue.

³Note that Figure 7.1 is not a DET graph, and that the axes are on different scales.

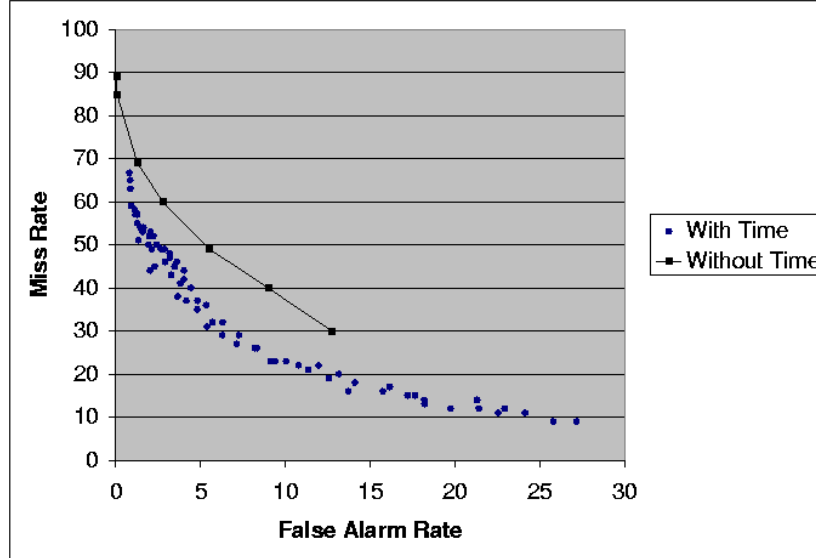


Figure 7.1. Effects of varying threshold parameters θ and β .

was evaluated in Chapter 6. In that chapter, it was determined that different comparison strategies resulted in different groupings of documents. This implies that the number of clusters, and thus the number of new events identified, are different. If we view the instantiation of a cluster seed as tantamount to detecting a new event, we find that the on-line single-link strategies tend to return more new events than the on-line average-link strategy at optimal parameter settings, which would appear to be advantageous in terms of misses, but not for false alarms. In the sections that follow, we compare event clustering strategies and compare the effectiveness of document/cluster comparison strategies as approaches to new event detection.

7.3.2.1 Retrospective Experiments

In this section, we evaluate our new event detection system varying the comparison strategies between documents and classifiers, and varying the dimensionality of the classifiers. We ran an analogous set of experiment to those we ran for event clus-

tering in Section 6.4, in which we compared on-line single-link strategies and on-line average-link strategies. Recall that in clustering, comparison values are determined for each cluster using the maximum or average decision scores resulting from the current document being processed. The classifier for the current document becomes a member of the cluster with highest comparison value. If no existing cluster results in a positive comparison value, then the document’s classifier initiates a new cluster and the document is assumed to discuss a new event.

We found optimal parameters for classifiers of dimensionality 25, 50, 100, and 200 using the optimization process described in the previous section. The effectiveness measures reported are based on finding the actual new event, i.e, results based on *skip* = 0. Results for the TDT1, TDT2-Train, and TDT2-Development corpora that led to maximum F1-measure on each corpus are listed in Figures 7.2 to 7.4 below.

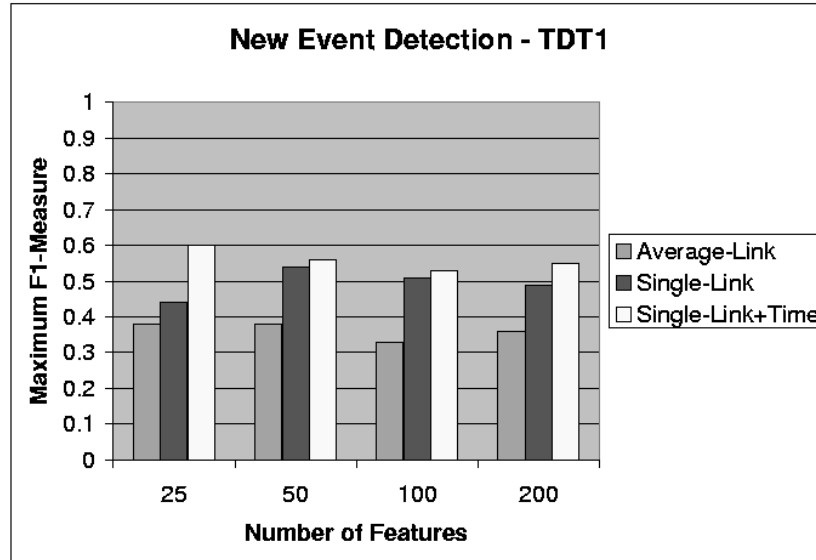


Figure 7.2. New Event Detection: TDT1 corpus

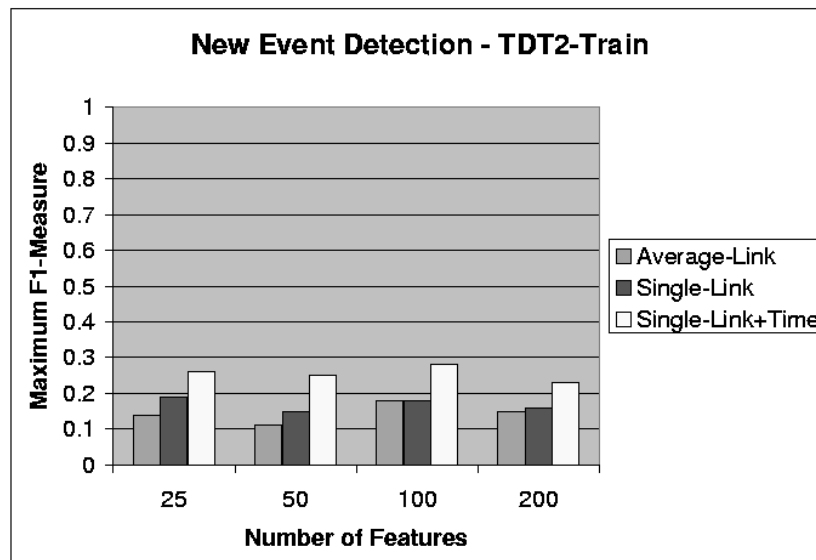


Figure 7.3. New Event Detection: TDT2-Train corpus

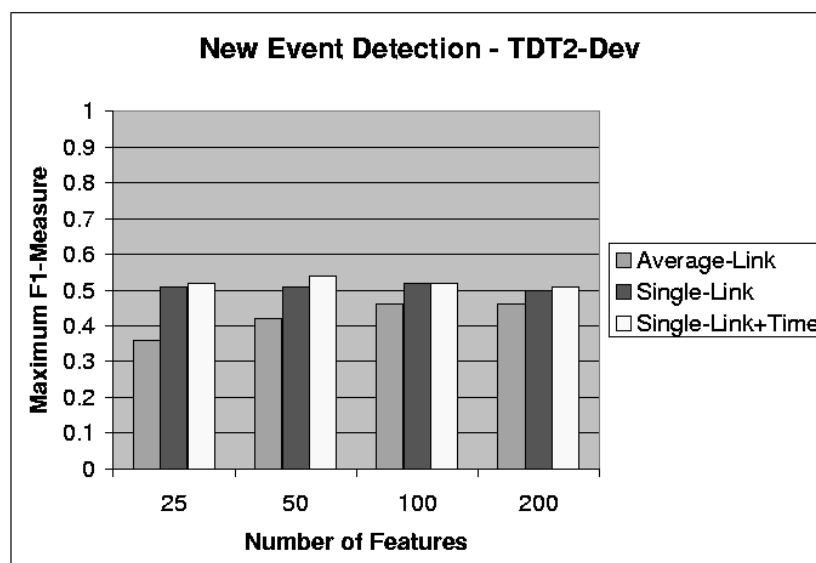


Figure 7.4. New Event Detection: TDT2-Development corpus

At optimal average effectiveness, new event detection is similar using different numbers of features for classifiers. However, an event-level comparison across dimensionality revealed that some classifiers benefit from using fewer features while others benefit from more. This analysis also suggested that choosing the appropriate dimensionality for each event would yield over 50% improvements in F1-measure on the TDT2-Train and TDT2-Development corpora. We have yet to find a methodology for determining the appropriate dimensionality for an individual event automatically, and we look forward to studying this problem in our future work.

The data from Figures 7.2 - 7.4 suggest that the on-line single-link strategy is on average better at detecting new events when $\beta > 0$, that is, when the time component of the threshold model is used. The on-line single-link strategies, in general, appear to be more effective than the average-link strategy using optimal parameters. However, all the strategies had lower accuracy on the TDT2-Train corpus. We believe this was caused by the “Monica Lewinsky Scandal” and the “Asian Financial Crisis”, which were two events that contained heavy news coverage comprising 10% of the documents in the corpus.

7.3.2.2 Predictive Experiments

We ran predictive experiments on the ASR and closed caption (CCAP) versions of the TDT2-Evaluation corpus. We used the parameters that on average gave rise to optimal effectiveness in the retrospective experiments using 50-feature classifiers. In the following analysis, we use Detection Error Tradeoff (DET) curves as well as the 11-pass evaluation methodology that is described above. The DET curve for the predictive experiment using the ASR data is listed in Figure 7.5, and the pooled-average hard decision results are listed in Table 7.6.

The DET graph in Figure 7.5 illustrates that the on-line single-link strategies have lower curves and thus more favorable error rate tradeoffs than the on-line average-

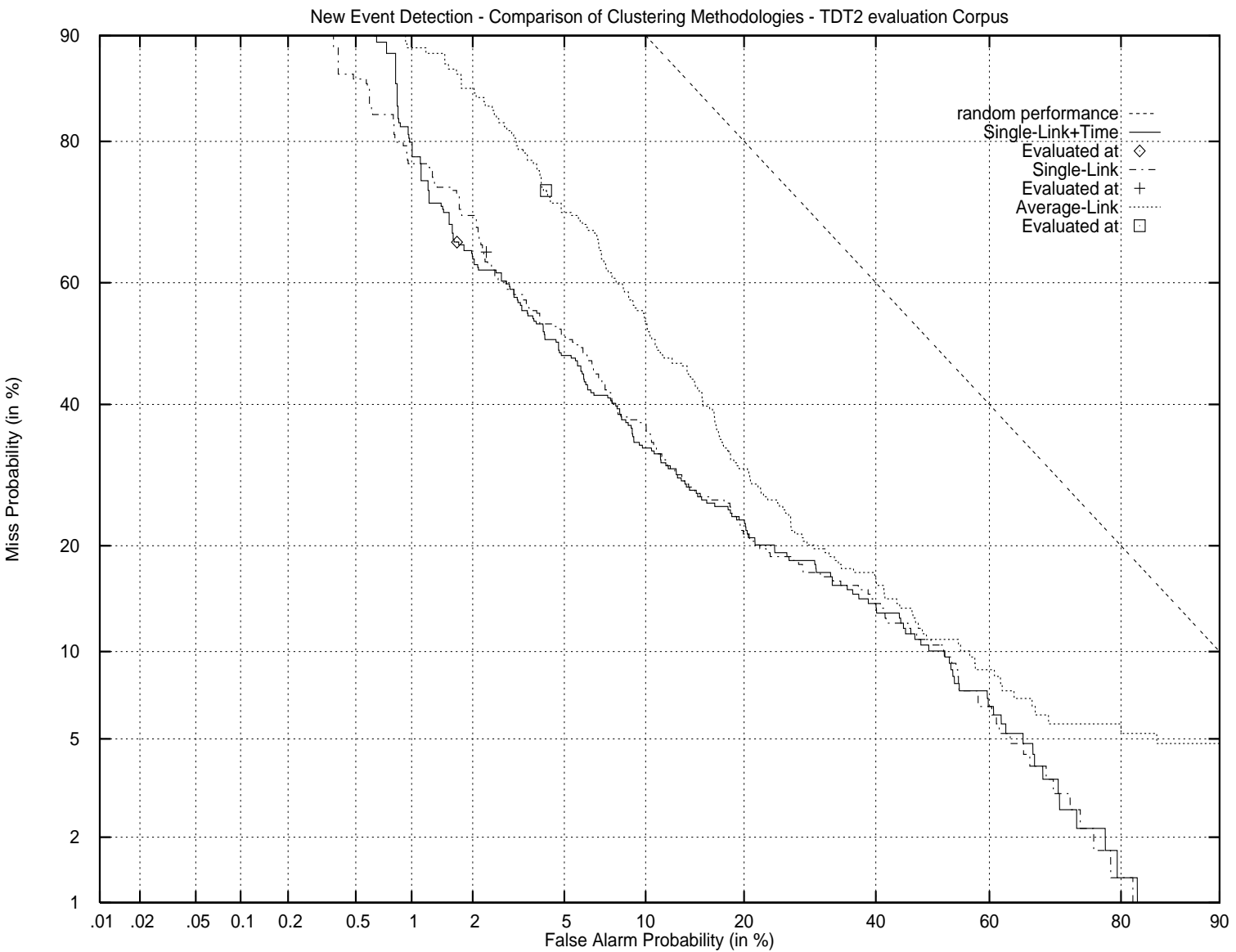


Figure 7.5. Comparison of clustering strategies for New Event Detection (TDT2-Evaluation corpus, ASR+NWT).

Table 7.6. Comparison of clustering strategies for New Event Detection (Pooled averages, TDT2-Evaluation corpus, ASR+NWT).

On-line Strategy	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
single-link+time	66%	1.69%	34%	26%	0.30	0.0297
single-link	63%	2.31%	37%	22%	0.28	0.0352
average-link	73%	4.22%	27%	10%	0.15	0.0559

link strategy. The improvement from the time component of the threshold model appears to be less distinguished than the results from the retrospective runs would have suggested. Nevertheless, from Table 7.6, on average we see a 7.1% improvement in F1-measure, and a 15.6% improvement in TDT2 cost when the temporal relationship between stories is modeled. This suggests the desirability of using the time component of the threshold model for an actual application.

We also tested the comparison strategies and parameter settings that gave rise to good event clustering in Chapter 6. In the following experiments we evaluate event clustering as the new event detection task. The threshold parameter for these experiments are those obtained from the optimization process for event clustering using F1 as the target utility measure. In the following experiments, we used $\theta = 0.26$ and $\beta = 0.001$ for the on-line single-link+time strategy, and $\theta = 0.33$ and $\beta = 0$ for the on-link single-link strategy. For on-line average-link, $\theta = 0.15$. The results for these experiments are listed in Table 7.7.

Table 7.7. New Event Detection using good parameters for Event Clustering (Pooled averages, TDT2-Evaluation corpus, ASR+NWT).

On-line Strategy	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
single-link+time	41%	7.50%	59%	12%	0.20	0.0818
single-link	35%	10.09%	65%	10%	0.17	0.1059
average-link	65%	6.53%	35%	9%	0.14	0.0769

The parameters for the on-line clustering strategies were different, and generally lower for new event detection, and we see an improvement in miss rates, but a sig-

nificant decrease in false alarm rates when applying the lower effective clustering parameters. The improvement in F1 and the decrease in cost resulting from using parameters optimized for new event detection are listed in Table 7.8.

Table 7.8. Percent improvement using New Event Detection threshold parameters.

On-line Strategy	F1	TDT2 Cost
single-link+time	50.0%	-63.7%
single-link	64.7%	-66.8%
average-link	7.1%	-27.3%

The data in Tables 7.6 to 7.8 suggest that on average, new event detection improved when different threshold parameters were determined explicitly for the task. These data suggest that good clustering strategies do not necessarily lead to effective new event detection. Recall that the on-line average-link strategy was shown to be an effective event clustering strategy, but does not result in finding many new events. In addition, the on-line single-link strategies need different parameter settings for event clustering and new event detection.

7.3.3 Impact of ASR technology

In the following experiments, we evaluate the effects of using text from the automatic speech recognition (ASR) process. The ASR process has an expected word error rate of 15%. We substituted the broadcast news documents with automatic transcriptions (ASR) with cleaner data containing closed caption manual transcriptions (CCAP). The newswire (NWT) documents remained the same. We used the same parameters for the CCAP+NWT data that were used on the ASR+NWT data above. The DET curves resulting from this experiment are in Figure 7.6, and the pooled-average results from the 11-pass evaluation methodology are listed in Table 7.9 below.

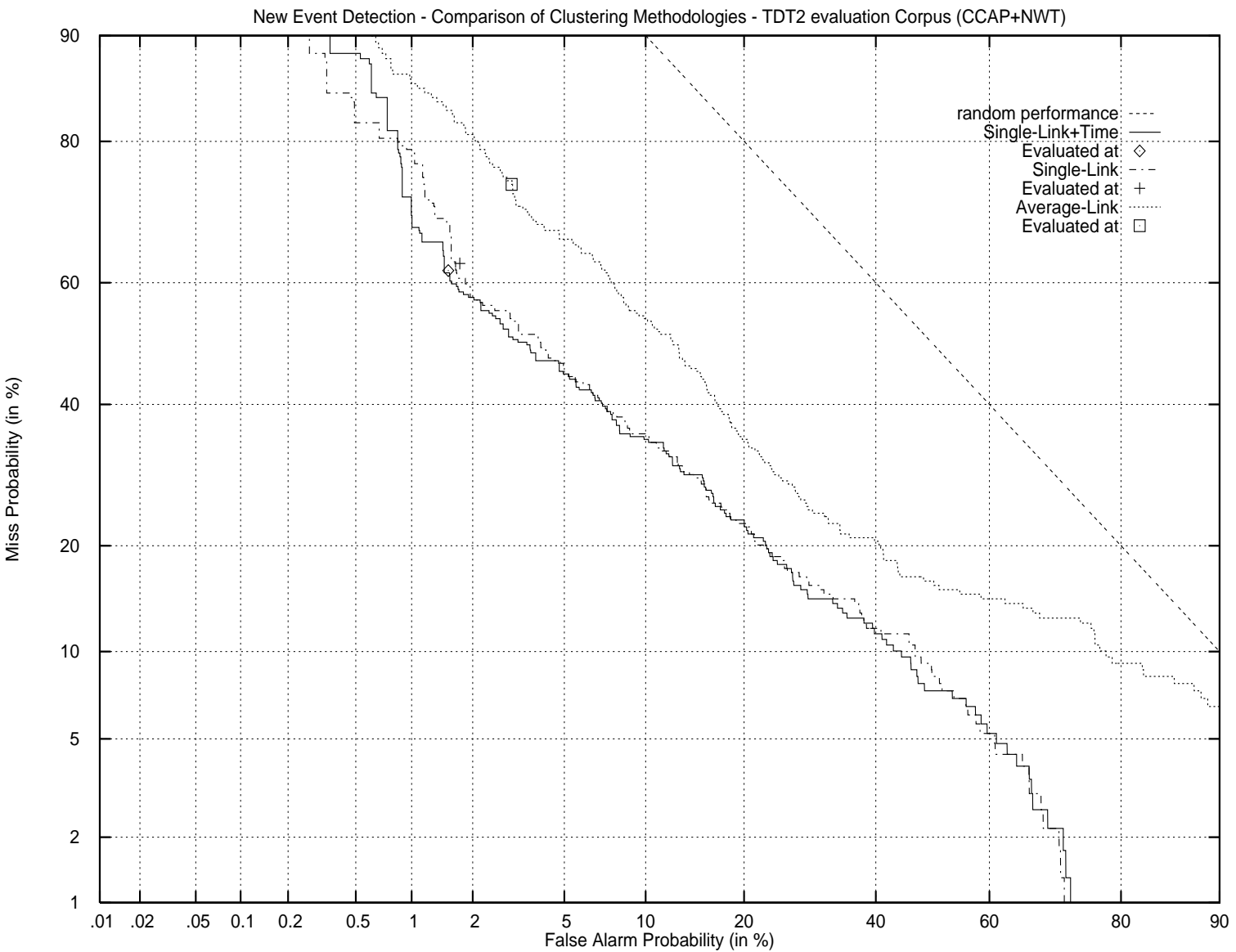


Figure 7.6. Comparison of clustering methodologies for New Event Detection (Pooled averages, TDT2-Evaluation corpus, CCAP+NWT).

Table 7.9. Comparison of clustering methodologies for New Event Detection (Pooled averages, TDT2-Evaluation corpus with Closed Caption source).

Type	Miss Rate	F/A Rate	Recall	Prec	F1	TDT2 Cost
single-link+time	61%	1.53%	39%	31%	0.35	0.0271
single-link	61%	1.73%	39%	29%	0.33	0.0291
average-link	73%	3.02%	27%	14%	0.18	0.0442

The on-line single-link strategies were more effective than the on-line average-link strategy for new event detection for both the CCAP and ASR sources. Furthermore, using the on-link single-link+time strategy gives rise to the best DET curve in Figure 7.6, and the most effective pooled-average results for F1-measure and TDT2 cost in Table 7.9. In addition, a comparison of the ASR and CCAP DET curves in Figures 7.5 and 7.6 indicates that the on-line average-link strategy benefits more than the on-line single-link strategies when the ASR data is replaced with CCAP data.

In Table 7.10, we summarize the percent improvements in effectiveness realized by replacing ASR sources with CCAP sources. For both F1-measure and TDT2 cost, the CCAP sources result in relatively high percent improvements in effectiveness over the ASR sources for both on-line single- and average-link strategies. In general, we would expect the ASR technology to give rise to more out-of-vocabulary (OOV) words than the manual transcription process, and thus we expect the OOV words to be the cause of an increase in classification error. This analysis suggests that the ASR technology hinders new event classification using our approaches.

Table 7.10. Percent improvement using CCAP vs. ASR transcriptions.

On-line Strategy	F1	TDT2 Cost
single-link+time	16.7%	-9.4%
single-link	17.8%	-17.3%
average-link	20.0%	-20.9%

7.3.4 Can Natural Language Phrases Increase Effectiveness?

A retrieval system that uses single-word features has its drawbacks. Consider a retrieval task where a user is interested in information about *Harley Davidson Motorcycles*. Documents containing the word “Davidson” without the word “Harley” become unwanted relevant candidates using weighted independent words. Boolean combinations of words introduce ambiguity between the concept they are intended to represent and other concepts that contain the same words. For example, a user interested in documents about *The World Bank* is not necessarily interested in retrieving documents about the merger that created *the world’s largest bank*. In these examples, specifying the query as a phrase would assist in discriminating between relevant and non-relevant documents. Previous research in document classification extends the feature space by extracting natural language phrases and more general *multiword features*.

The utility of multiword features and their effects on retrieval have gotten mixed reviews in the previous literature. Lewis has documented some of the earlier work pertaining to representation and ambiguity issues arising from the use of phrases. He shows that “[t]he optimal effectiveness of a text representation based on using simple noun phrases...will be less than that of a word-based representation” [47]. In addition, TREC routing and filtering systems using multiword features do not appear to significantly outperform systems that use only single-word features [68].

These negative conclusions regarding multiword features are offset by several positive results that have been reported. For example, Fagen reports 2.2%-22.7% improvements in average precision using phrasal indexing [29]. Strzalkowski and Carballo [81] describe improvements in ad hoc retrieval using natural language phrases. Boolean features comprising multiple words have been used to improve precision by Hearst [37]. Papka and Allan [58] showed that in the context of massive query expan-

sion (queries with several hundred features), retrieval effectiveness improves by using proximity operators on pairs of words.

There are several methods for extracting phrases. N-gram models based on mutual information metrics are used to find sets of adjacent words that are likely to cooccur within sentences [9]. Part-of-speech tagging using pre-specified syntactic templates or more complex natural language parsing [29, 88] gives rise to related multiple words comprising noun, verb, and prepositional phrases. Riloff and Lehnert [66] use information extraction techniques that build multiword features as an integral part of their message understanding system.

In what follows we attempt to embed a statistical natural language parser into our new event detection system. The parser was developed by Eugene Charniak at Brown University [17], and it is based on a Hidden Markov Model that predicts productions for a context-free grammar of the English language. The parser was trained using the Penn Tree-Bank of hand parses for the Wall Street Journal, and obtains recall and precision of roughly 77% on Tree-Bank experiments [18].

We use Charniak’s parser for its ability to detect proper nouns and dates. In effect, our goal was to select features associated with the *who*, *what*, *where*, and *when* of the document. Figure 7.7 shows some examples of proper nouns that we were able to extract from the parse of a document discussing the O.J. Simpson murder trial. The lists in Figure 7.7 were organized manually since the parser did not distinguish between people, places, things, and dates, but grouped them all into two tags: one for singular proper nouns, and one for plural proper nouns. The phrases were obtained by finding words in a document containing a proper noun part-of-speech tag. When a tag was found, we produced the phrase within which the word occurred by selecting the words in the subtree spanned by its parent.

We tried a series of experiments on the TDT1 data where we parsed each document and extracted proper nouns and dates. Our hypothesis is that if proper noun phrases

Who o.j. simpson, nicole brown simpson, ronald goldman, a los angeles police spokeswoman, a los angeles police detective, shapiro, jose camacho, allen wattenberg, michele kestler.

What los angeles television station knbc, the week tabloid national enquire, ross cutlery, the national enquire offer, ford bronco, the los angeles police crime lab, wattenberg and wattenberg,

Where chicago, o' hare international airport, a chicago hotel, the brentwood section, los angeles airport.

When thursday night, thursday, june 12, the first time thursday, friday, tuesday, the july 4 holiday weekend, thursday, may 3.

Figure 7.7. Proper Nouns Produced by Natural Language Parser

and dates are important lexical features, then increasing their weight in a classifier should result in an effectiveness increase for new event detection, and decreasing their weights should result in an effectiveness decrease.

We augmented the classifier formulation process using single-word lexical features. After formulating a classifier for each document, we identified the features that mapped to words in the proper noun phrases contained in the document from which it was formulated. We initially set the feature weight using the usual weight assignment, and then we increased and decreased the feature weights only for words that appeared in the proper noun phrases collected from the parse.

Figure 7.8 shows the DET curves for an experiment using the TDT1 corpus and 200-feature classifiers. We determined that 47.5% of the words that were contained in the proper noun phrases extracted from the parse appeared in the classifiers. In addition, 21.2% of the lexical features in the classifiers mapped to words in the proper noun phrases. In general, we see that classification improves at false alarm levels below

5% when the weights of the lexical features in the proper noun phrases are doubled. When the weights of these features are halved, classification tends to get worse at false alarm levels below 5%. Unfortunately this improvement does not appear to be stable throughout the entire curve. In the graph, the curves cross for different weight settings, which suggests that one weight setting is not more robust than another.

We also observed that effectiveness was reduced when all the words associated with proper noun phrases were removed from the classifier by setting their weights to zero. We did not see that more gains resulted from quadrupling weights. We also tested this approach using 25-feature classifiers, and found no apparent improvement from increasing weights in the features mapping to the words of proper noun phrases.

Our attempts to use the parser on the TDT2 corpora failed. The major problem was that the automatic speech recognition data lacked the punctuation needed for the parser to work. The closed caption data, on the other hand, had too much punctuation. We found many spurious periods that appeared to be used to indicate a pause in the flow of the news story. This additional punctuation also prevented the parser from working.

Several feature selection issues associated with natural language phrases became apparent from this experiment. For example, we found that many proper noun phrases were already represented in the classifier. Another problem is that in broadcast news there are many references to news correspondents' names. Reporters use their own names for communication cues within live coverage to signal transitions, so a reporter's name that appears often in the text is likely to be selected for the classifier. It could be that a reporter's name indicates the type of coverage, for example, Wolf Blitzer often covered politics from the White House, but in general, names of reporters are not good features, and should be removed from a classifier. In addition, there are many dates and proper noun phrases that have no significant bearing on the content of the story, so using all of these features in an oblivious manner will

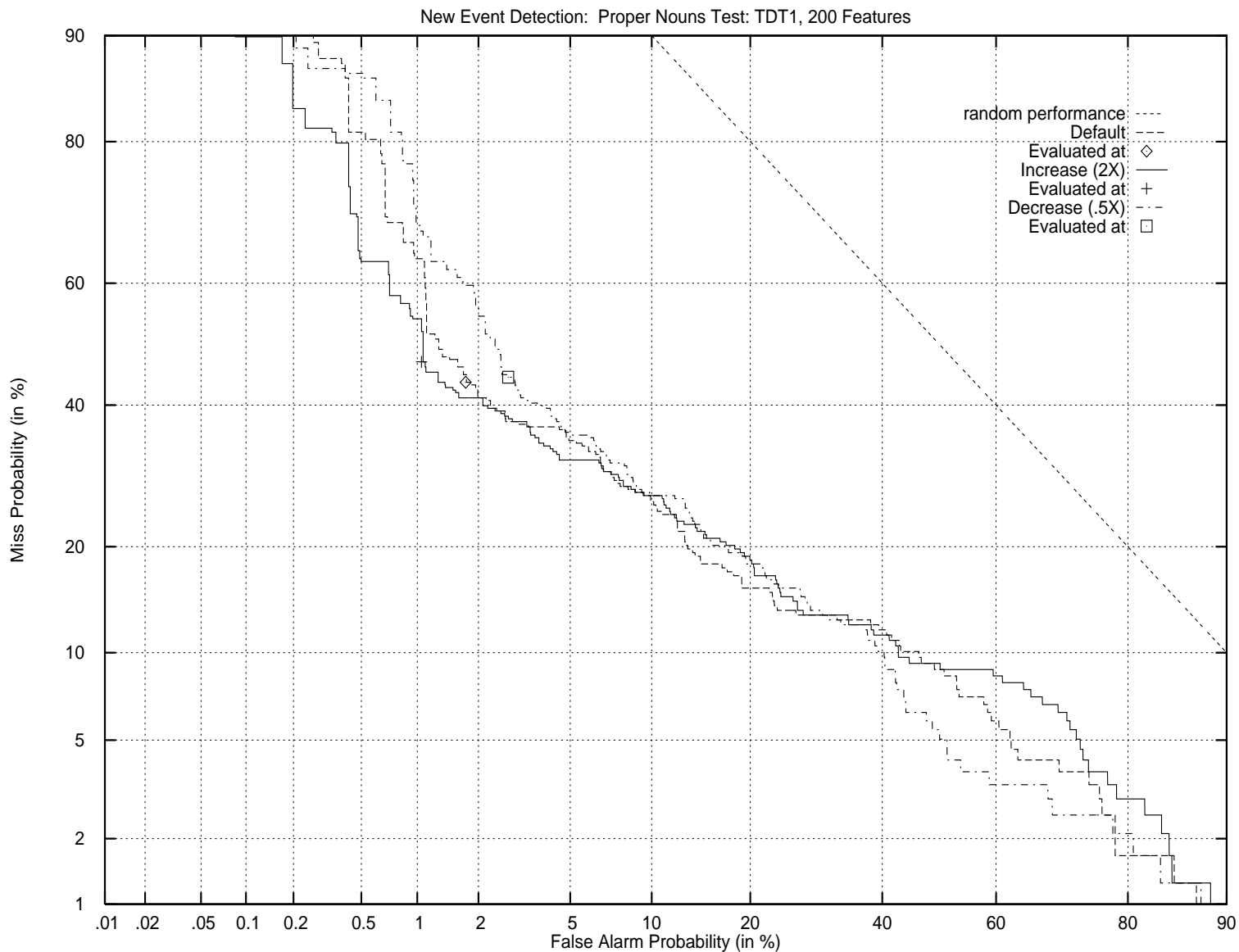


Figure 7.8. Impact of Proper Noun Extraction on Error Rates

most likely hurt rather than help effectiveness. Even if these problems were solved, we believe that a more effective use of the parse data needs to be established in order to show significant gains and justify the direct application of this technology to our new event detection system.

7.4 TDT Pilot Study

In this section we compare our approach to new event detection to implementations from other TDT sites. The data presented in this section are results from the TDT Pilot Study, and include an evaluation of systems from Carnegie Mellon University (CMU), and from Dragon Systems (DRAGON). It should be noted that CMU and Dragon Systems have not since worked on the problem of new event detection, and we therefore use the results and descriptions of the systems discussed at the TDT Pilot Study Workshop [5]. In what follows, we compare their approaches to the system we implemented for the Pilot Study, which is similar to our current implementation.

7.4.1 CMU and Dragon System Approaches

Carnegie Mellon University embedded the SMART retrieval engine in their system. They used a clustering strategy with a *detection threshold* that governed the minimum document-cluster similarity score required for the system to label the current document as containing a new event. They also used a *combining threshold*, which was the minimum score required for adding a document to an existing cluster. Time was incorporated in the detection decision by limiting comparison to documents that appeared within a constant window size of time from the document being processed. They reported that experiments using a cluster representation between dimensionality 50 and 100 yielded the best results. They also reported that experiments using no prototyping yielded better results than those using prototyping.

The Dragon system used a language modeling approach of single-word (unigram) frequencies for cluster and document representations: their document representation did not use *tf·idf* scores, which were used by our system and the CMU system. Dragon’s cluster comparison methodology is based on the Kullback-Leibler distance measure [5]. The modification included a decay term which decreased the similarity measure for clusters containing documents closer in sequence to the current document on the stream. In addition, they used a pre-processing step in which an iterative *k-means clustering algorithm* was used to build 100 background models (clusters) from an auxiliary corpus. In their decision process, a document is considered to contain a new event when it is closer to a background model than to an existing story cluster.

7.4.2 Cross-System Comparisons

The new event detection results using the data and the systems described above are presented in Figure 7.9. The DET curves show the UMASS system has miss rates that are lower than the other systems between false alarm rates of 1% and 10%. Below the 1% false alarm rate, the UMASS and CMU systems outperform the Dragon system. At the 10% level of false alarms the systems converge, and at 30%, the UMASS system experiences higher miss rates than the CMU and Dragon systems.

Table 7.11. Effectiveness measures for systems presented at the first TDT workshop.

System	Miss Rate	F/A Rate	Recall	Precision	F1
UMASS	50%	1.34%	50%	45%	0.45
CMU	59%	1.43%	41%	38%	0.39
DRAGON	58%	3.47%	42%	21%	0.28

A comparison of average effectiveness using the reported measures are listed in Table 7.11. The UMASS system has a miss rate that is 18% lower than the other systems at lower false alarm levels. These data suggest that our text representation and classification model are relatively effective for new event detection. The difference

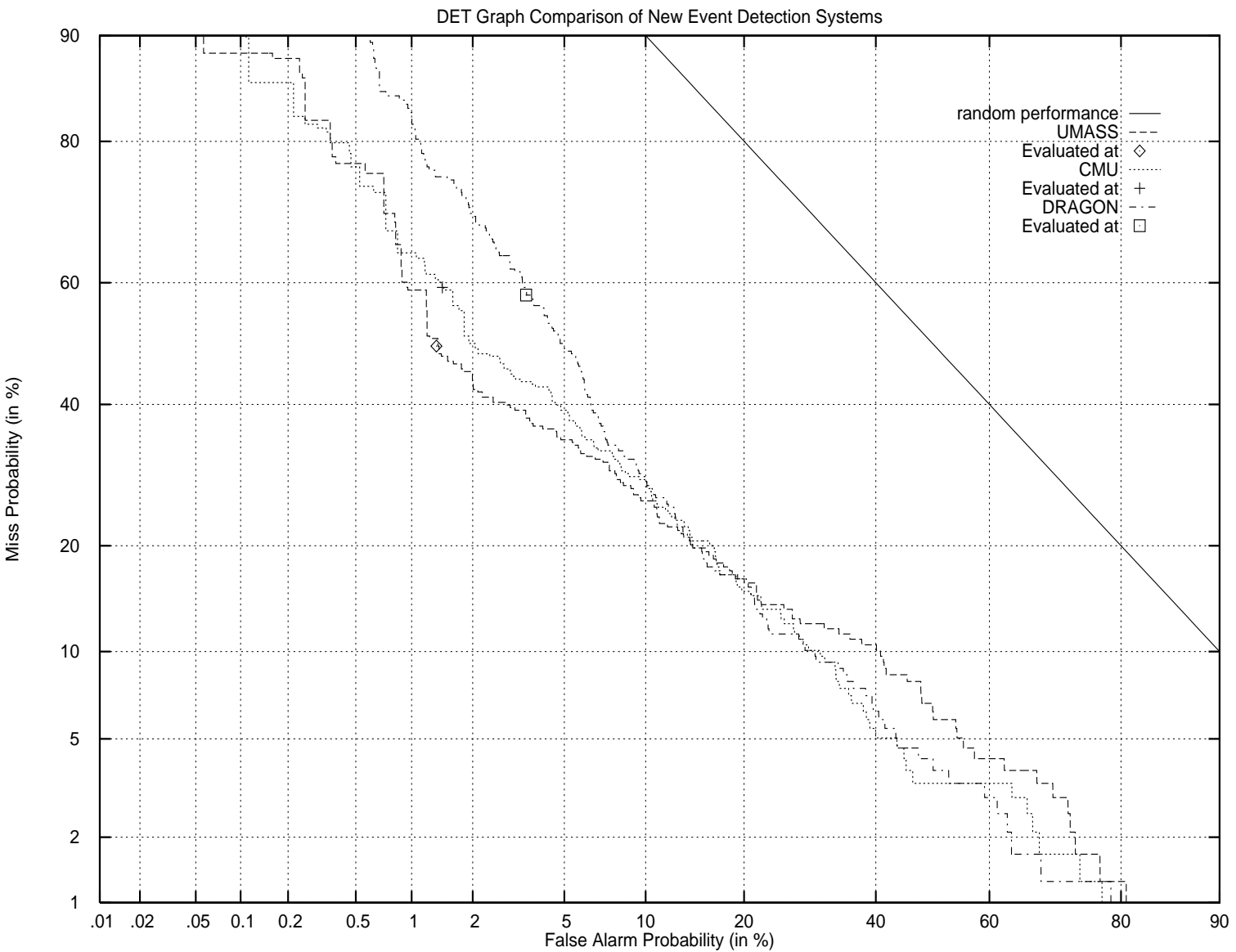


Figure 7.9. DET graph for systems presented at the first TDT workshop.

in effectiveness indicates that the UMASS system, on average, correctly detected two additional new events. This improvement is not significant based on a sign test across event-level effectiveness.

7.5 Discussion of Approach to New Event Detection

In this chapter, we presented our algorithm for new event detection, and analyzed extensions to our classification model that incorporate the properties of broadcast news. In particular, we showed a method for modeling the temporal relationship between documents, and a method for incorporating proper noun phrases into the classifier formulation process. Our results suggest that these extensions can be modeled efficiently and lead to improved classification accuracy for new event detection.

We evaluated the use of on-line clustering as an approach to new event detection, and we found that the on-line single-link+time strategy appeared to be more effective than the other on-line clustering strategies tested. Our analysis suggests that good strategies and parameter settings for event clustering do not necessarily result in effective new event detection. In addition, we showed that the ASR technology negatively affects classification.

7.5.1 The Good News

The effectiveness of our approach to new event detection is sufficient for certain applications. We find that on the TDT2 data, we correctly classify 35%-60% of the new events from broadcast news with relatively low false alarm rates. At the lowest reported false alarm rate of 3.2%, our results suggest that a user will need to read only 650 of the 20,000 documents available over a two-month period. This implies that our system can significantly reduce the workload of someone searching for new events in large amounts of news.

7.5.2 The Bad News

Though our approach appears to be comparable to other approaches to new event detection, classification using these methods is far from perfect. From the TDT2 assessment process, we know that assessors of the same event were in agreement on 90% of the assessments [31], so we would like to see miss rates of 10% at much lower false alarm levels than ones we measured. A major problem is that even with several extensions to the general word-cooccurrence model, the improvements we are experiencing from the various techniques are incremental. With the exception of modeling the temporal relationship between documents, other approaches, including average-link clustering and additional feature selection, did not appear to have a significant impact on new event detection effectiveness.

Another major problem is the limitation of the word-cooccurrence model. When we analyze system misses, which occur when documents containing new events are labeled as “not new”, we find that at low dimensionality, misses occur because important event-level features are not appearing in the classifier, or are not sufficiently weighted in the classifier. For example, we found the first story about the “Crash of US Air Flight 427”, resulted in a positive decision score from a classifier formulated for an earlier document about the “Crash of US Air Flight 1016”. The distinguishing lexical feature is the flight number, which does not always have a high term frequency in a relevant document. Figure 7.10 contains the query syntax of a classifier that results from one of the US Air plane crashes, which often resulted in positive decision scores for documents discussing other plane crashes. We had similar problems with other disaster events. For example, the first story discussing the “Oklahoma City bombing” resulted in a positive decision score from a classifier formulated from a document discussing the earlier “World Trade Center bombing”. At higher dimensionality, the two bombing events were separable, but the airline crashes were not.

However, as we mentioned previously, choosing the correct dimensionality automatically for a classifier is a hard problem.

```
q104 = #WSUM( 1.0
1.175688 accident
1.125646 crash
1.070033 plane
0.935901 cause
0.935901 investigate
0.935901 look
0.852374 air
0.852374 aircraft
0.852374 survivor
0.752039 usair );
```

Figure 7.10. General “US Air plane crash” classifier.

Other problems appear to be associated with events that are heavily covered in the news. These events often span several hundred days and contain significantly more documents than other events in the corpora. The more documents discussing an event, the more its effectiveness is weighted in the pooled-measures and DET curve, and also the greater prior probability that false alarms will occur. For example, errors in the “Monica Lewinsky Case”, which has the most documents of all the events in the TDT2 corpora, are a major cause of poor effectiveness on the TDT2-Train corpus.

We also noticed that some domains, such as courtroom coverage, led to errors. For example, using the best parameters on the TDT1 corpus, the system could not distinguish between documents from the “O.J. Simpson Trial” and documents pertaining to other court cases. Different events in the same country are also problematic. For example, documents related to various events in Bosnia caused our system to miss “Carter’s Visit to Bosnia”. These examples indicate that the system was unable to detect certain events that are discussed in the news at different levels of granularity.

CHAPTER 8

CONCLUSION

We have implemented and evaluated solutions to the news classification problems of *new event detection*, *event clustering*, and *event tracking*. The results presented in this work are based on problem definitions, evaluation methodologies, and data developed by the Topic Detection and Tracking (TDT) project. We used the same text representation for all three problems, which was based on the Inquiry [12] retrieval engine, and extended Inquiry’s functionality with classification processes that incorporate the properties of broadcast news as a major component. In the following sections, we summarize our research contributions, and discuss our conclusions based on the experimental results presented in this work.

8.1 Summary of Research Contributions

The research presented in this dissertation furthers the understanding of issues relating to unsupervised and supervised document classification problems. The problem of new event detection has not been the focus of research prior to the TDT Pilot Study, and to the best of our knowledge, this work is the first thorough investigation of the problem. We introduced a single-pass classification algorithm for new event detection, and we presented methods that model the properties of broadcast news which extend previously studied text representations. Our results suggest that our single-pass algorithm is a good basis for new event detection, and we find that modeling the properties of broadcast news leads to improved classification effectiveness.

We also tested previous approaches to document clustering and evaluated their effectiveness in the context of event clustering for the domain of broadcast news. In general, previous approaches to clustering are based on retrospective solutions, where all the data are available before clustering begins. We re-evaluated single-link, average-link, and complete-link clustering strategies, but used them in an on-line environment, in which a cluster is determined for the current document before looking at the next document. We introduced a classifier-based approach to the problem, and presented a threshold model that incorporates the temporal relationship between news stories. Our results suggest that an on-line single-link clustering strategy extended with a time component is more effective than other comparison strategies for both event clustering and new event detection.

The representation we used for new event detection and clustering is derived from a classifier formulation process previously used for supervised text classification problems such as filtering and routing. We tested previous approaches to classifier formulation, but applied them to the TDT tracking problem. Much of the related work in this area is based on news story classification by topic. Here, we evaluated several approaches, and applied them to on-line event-based document classification. This work furthers our understanding of feature selection, feature weight assignment, and threshold estimation issues for on-line environments. The major focus of our tracking experiments was to evaluate the impact of relatively few training samples on classification accuracy. We introduced a theoretical framework for estimating classifier thresholds that is useful for understanding the bias inherent in threshold estimates using small numbers of relevant training documents.

8.2 Discussion of Experimental Results

New event detection is an abstract document classification task that we have shown can be reasonably solved using a single-pass approach. Our algorithm for this

problem is based on a notion of *event identity*. Our approach is to represent the events discussed in each document with a query syntax and a threshold, which together form a classifier for the content of the document. When a document appearing on the news stream is not classified as a positive instance by any of the existing classifiers, we assume the document discusses a new event. If the document is classified as a positive instance, then we assume the document is similar in content to a previously processed document, and therefore does not discuss a new event. Our results indicate that we find between 35%-60% of the documents discussing new events at relatively low false alarm rates of 1.27%-3.2%.

Our approach to new event detection is very similar in nature to on-line clustering algorithms. The step in a clustering algorithm in which the decision is made to assign a document to an existing cluster or to initiate the formation of a new cluster is tantamount to deciding whether a document discusses a new event. With this in mind, we explored different cluster comparison strategies which included on-line single-, average-, and complete-link strategies. Each of these strategies gives rise to a different on-line clustering of the documents, and thus a different set of new events. We found that our implementation of complete-link did not work well for either new event detection or event clustering. While the average-link approach worked well for clustering, but not for new event detection, the on-line single-link+time strategy worked best for both tasks.

The motivation for exploiting the property of time was based on an analysis of the TDT corpora which suggested that documents closer together on the stream were more likely to discuss the same event than documents further apart. Time was incorporated as a component of our threshold model, which resulted in improved effectiveness for the on-line single-link strategy when applied to new event detection and event clustering.

In addition, we explored a process that extracts the lexical features that capture the *who*, *what*, *when*, and *where* contained in news. The processes involved a natural language parser that was used to extract proper noun phrases and dates from the stories in the news stream. Unfortunately, the parser did not work on the TDT2 corpora due to missing or inaccurate punctuation in the automatic and manual transcriptions of the radio and television broadcast news sources. We were able, however, to experiment with the parser on the TDT1 corpus, and the results suggested that many of the words in the proper noun phrases were already in the classifiers that were being formulated. When we increased the weights of the classifier features that appeared in proper noun phrases and dates, classification accuracy improved, and when we decreased their weights, classification accuracy declined. This suggests that identifying proper noun phrases and dates can help new event detection; however, in order to extract them from data with punctuation errors, a more robust extraction process is needed.

The automatic speech transcription (ASR) process affected not only the natural language parser, but also the effectiveness of our approach to new event detection and event clustering. We found that effectiveness improved between 9.4% to 20.9% for both single- and average-link approaches when the ASR data was replaced by manual Closed Caption (CCAP) transcriptions. The impact of the ASR technology was minimal for event tracking. The ASR process was developed by Dragon Systems, which is currently among the best recognition processes available. However, the system has been reported to produce word error rates of 15% on similar data. A reduction in word error rates should lead to improved new event detection as well.

The majority of our experimental runs for all three problems involved an exhaustive search for system parameters, including the parameter specifying the number of lexical features to use when formulating classifiers, i.e., classifier dimensionality, as well as the parameters for our threshold model. Our current solution for new event

detection uses constant dimensionality and two global threshold parameters. Our experiments indicated that a good set of values existed for these parameters in 80% of the events tested. In addition, we found that if it had been possible to determine optimal dimensionality automatically, we would have realized an over 50% effectiveness improvement in the classification of new events.

Unlike ranked retrieval, true text classification tasks require that a system make hard decisions about a document's relevance. Therefore, threshold parameter estimation becomes an integral part of the classification approach. We found that the threshold parameters that optimized pooled average effectiveness measure were similar across the TDT corpora, and thus an averaging of optimal parameters resulted in comparably effective classification relative to other systems evaluated at TDT2. However, when we compared optimal threshold parameters that worked well for event clustering against those we determined for new event detection, we found that good parameters for new event detection were consistently lower than good parameters for event clustering. We therefore believe that a good clustering approach does not necessarily lead to a good solution for new event detection. Additional evidence that suggests this hypothesis is true was our application of on-line average-link clustering, which was effective for event clustering, but not for new event detection.

We presented several experiments for the event clustering problem, where we compared single-pass clustering solutions that included on-line single-link and average-link cluster comparison strategies. The data suggest that augmenting on-line single-link clustering with a time component was the most effective approach when using automatic and manual transcriptions for broadcast news sources. Other clustering experiments suggested that average-link is comparable to single-link+time when audio sources for news are manually transcribed. However, our single-link+time approach resulted in the lowest story-weighted cost realized by the systems evaluated at TDT2,

and we suggest using single-link+time for on-line clustering of broadcast news because it is both efficient and effective.

We viewed the tracking problem as an instance of on-line document classification, and used extensions to techniques that have been previously shown to work well for the related problem of document filtering. We compared variations of a static classifier formulation process that included expansion with multiword features and weight-learning steps. In addition, we tested adaptive classifier formulation, which has the effect of including new features in the classifier over time. We found insignificant effectiveness improvements using these variations, which in most cases require significant computing resources. The main problem with the weight learning approaches is that classifiers and thresholds formulated using the static approach with few relevant training instances often separated their training data, and thus further improvements from supervised weight learning were limited. The adaptive technique appeared to work well on the target evaluation condition ($Nt = 4$), but proved less robust than the static approach for $Nt < 4$. We suggest using the static classifier formulation approach because it is relatively fast, effective, and uses fewer parameters than the other approaches.

In addition, we introduced a theoretical framework for automatic threshold parameter estimation for our static classifier formulation process. We view the threshold as a statistic of the incoming data stream that is estimated using an optimization process for a pre-specified utility measure applied to the training data. We defined the notion of bias in terms of threshold estimators, and illustrated that the amount of bias increases when fewer relevant training samples are used. We present two approaches that learn threshold estimation bias and result in effective estimates of parameters.

In retrospect, the comparisons between our approaches and those of other TDT participants indicate that different views of the tasks lead to different retrieval models which however result in similar effectiveness. Our systems, in general, were compara-

ble in effectiveness to the best systems for each of the problems, and we attribute this to our efforts in parameter estimation. In addition to similar overall effectiveness, the common element among the systems is the underlying model of word-cooccurrence used to determine when two documents discuss the same event. We believe this model is the key to the event classification solutions, and that improvements in effectiveness will come more from modeling the properties of events than from modifying existing retrieval models.

CHAPTER 9

FUTURE WORK

Our approaches to on-line new event detection, clustering, and tracking answered several questions regarding the use of existing Information Retrieval representations in on-line classification environments. Our work on parameter estimation using few relevant documents suggests that we can obtain classification accuracy relatively close to optimal effectiveness using exhaustive parameter searches; however, the accuracy of our approaches even at optimal parameter settings is far from perfect.

One area of concern for all three classification problems was selecting the dimensionality and the types of lexical features to use in a text classifier. Our plan for future work involves improving the feature selection and extraction methodologies of our approach to new event detection. In several experiments, we found that classifiers did not contain all the features that distinguish an event from its more general topic. For example, distinguishing lexical features such as a flight number and accident location were not necessarily included in each classifier formulated from a plane crash story. Other problems included over-weighting a feature, such as Bosnia, in classifiers that were used to track events about more specific news coverage. We plan to extend our classifier representation with a model that distinguishes event-level from topic-level features. Other aspects of feature selection to develop include finding a more effective use of the proper noun phrase data, which should lead to additional effectiveness gains and justify the direct application of natural language parsing technology to our new event detection system.

Our results suggest that cleaner data improves classification effectiveness. In the new event detection and event clustering experiments, we saw improvements in all cluster comparison strategies using the cleaner closed caption data. In addition, removing broadcast news header and trailer snippets resulted in improved effectiveness for both problems. There would appear to be several applications for text pre-processing steps that may result in further improvements to classification accuracy, such as removing reporters' names and other text that is specific to the broadcast news program. Another example is a processing step that would resolve some of the word errors resulting from the automatic speech recognition process.

There appear to be several applications for event classification, and the interest in Topic Detection and Tracking (TDT) is increasing with each phase of TDT. We expect several new approaches to the problems we discussed here to emerge from this research effort, and we look forward to future TDT comparisons, which may help us understand how improvements in text classification can be made.

What remains to be seen is whether or not this technology can be used to solve other abstract text classification problems. One such problem we plan to investigate involves classifying stories as "good news" or "bad news" with respect to the prospects of financial instruments such as stocks, bonds, and commodities. Currently, several prediction models exist that find patterns in price fluctuations. However, these models do not incorporate the news and events that are responsible for the actual price movements. From a research perspective, this domain may be suitable for the approaches that we have pursued in this dissertation. For example, a proper noun extraction technique could be used to match news stories to related underlying financial instruments. If these matches are accurate, then the short term price movements of the related instruments could be used as relevance feedback to train classifiers to detect the types of news stories that result in increases or decreases in prices. In addition, there could be several possibilities for incorporating domain knowledge into the

classification process. For example, news about a company reporting strong earnings is not necessarily an instance of “good news”. A model of expected versus reported earnings would need to be included in the classification process, which should result in improved stock price predictions. In addition to the financial markets, we are hopeful that this technology will be useful for applications in other areas.

APPENDIX A

The following sections list the events that were annotated for the TDT corpora. Some of the events did not have relevance judgments and were excluded from the relevance judgement listing in Table 3.2.

1 Events in TDT1 Corpus

1. Aldrich Ames Spy Case
2. The Arrest of ‘Carlos the Jackal’
3. Carter in Bosnia
4. Cessna Crash on White House Lawn
5. Salvi Clinic Murders
6. Comet Collision with Jupiter
7. Cuban Refugees Riot in Panama
8. Death of Kim Jong II
9. DNA Evidence in OJ Trial
10. Haiti Ousts Human Rights Observers
11. Hall’s Helicopter Down in N. Korea
12. Flooding in Humble, Texas
13. Breyer’s Supreme Court Nomination

14. Nancy Kerrigan Assault
15. Kobe Japan Earthquake
16. Detained U.S. Citizens in Iraq
17. New York City Subway Bombing
18. Oklahoma City Bombing
19. Pentium Chip Flaw
20. Quayle's Lung Clot
21. Serbians Down F-16
22. Serb's violation of Bihac Safe Area
23. Faulkner's Admission into the Citadel
24. Crash of US Air Flight 427
25. World Trade Center Bombing

2 Events in TDT2-Train Corpus

1. Asian Economic Crisis
2. Monica Lewinsky Case
3. Peruvian Anti-torture Laws
4. McVeigh's Navy Dismissal & Fight
5. Upcoming Philippine Elections
6. Israeli Palestinian Raids

7. Fossett's Balloon Ride
8. Casey Martin Sues PGA
9. Karla Faye Tucker
10. Mountain Hikers Lost
11. State of the Union Address
12. Pope Visits Cuba
13. 1998 Winter Olympics
14. African Leaders and World Bank President
15. Current Conflict with Iraq
16. \$1 Million Stolen at World Trade Center
17. Babitt Casino Case
18. Bombing Alabama Clinic
19. Cable Car Crash
20. China Airlines Crash
21. Tornado in Florida
22. Diane Zamora
23. Violence in Algeria
24. Shevardnadze Assassination Attempt
25. Shoplifter's Hand Amputated
26. Oprah Lawsuit

27. Unearthing of Pharoah's Tomb
28. Mary Kay LeTourneau
29. Buffett Buys Silver
30. Pension for Ms. Schindler
31. John Glenn Back in Space
32. Sgt. Gene McKinney
33. Superbowl '98
34. David Satcher Confirmed
35. Holocaust Museum Resignation
36. Reverend Lyons Arrested
37. Quality of Life Campaign, New York City

3 Events in TDT2-Development Corpus

1. LaSalle Boat Found
2. India Parliamentary Elections
3. Tello (Maryland) Murder
4. Grossberg Baby Murder
5. Asteroid Coming?
6. Dr. Spock Dies
7. National Tobacco Settlement

8. Mt. Cook Climbing Accident
9. Great Lake Champlain?
10. Viagra Approval
11. Jonesboro Shooting
12. Boehner-Gingrich Chat Taped
13. JJ the Whale
14. Thai Police Lt. Arrested
15. Strike in Germany
16. Capps Replacement Elections
17. Albright to Canada
18. Boeing Discrimination Suit
19. James Earl Ray's Retrial?
20. World Figure Skating Champs
21. Guinness Beer Gag
22. UCONN Spring Weekend
23. POW Memorial Museum
24. Kenya Boosts Tourism
25. Mandela Visits Angola
26. Bird Watchers Hostage
27. Race Relations Meetings

28. Rats in Space!

29. Marcus Allen Retires

4 Events in TDT2-Evaluation Corpus

1. Spanish Dam Broken

2. Debellia Treatment Cures Cancer?

3. Carter Family Reunion

4. India Begins Nuclear Testing

5. Israeli-Palestinian Talks (London)

6. Tony Awards '98

7. Mother-Tongue Teaching

8. Nigerian Protest Violence

9. Removal and Restoration of Food Stamps

10. Anti-Suharto Violence

11. Unabomber Trial and Conviction

12. Denmark Strike

13. Akin Birdal Shot & Wounded

14. Human Rights Conference Ethiopia

15. Contaminated Apple Juice Case vs. Odwalla Inc.

16. Abortion Clinic Acid Attacks

17. World AIDS Conference
18. Job Incentives
19. Saudi Soccer Coach Sacked
20. GM Strike
21. NBA Finals '98
22. Anti-Chinese Violence in Indonesia
23. Afghan Earthquake
24. Unwed Fathers' Law
25. German Train Derails
26. Anti-obesity Drug
27. Puerto Rico Phone Strike
28. Nazi-plundered Art
29. Turkish Military Officers Fired
30. Clinton-Jiang Debate
31. Martin Fogel's Law Degree
32. Cubans Returned Home
33. Oregon Bomb for Clinton?
34. Goldman Sachs - Going Public?

BIBLIOGRAPHY

- [1] J. Allan, L. Ballesteros, J. Callan, W.B. Croft, and Z. Lu, "Recent Experiments with Inquiry," *Proceedings of TREC-4*, 49-63, 1995.
- [2] J. Allan, "Incremental Relevance Feedback for Information Filtering," *Proceedings of ACM SIGIR*, 270-278, 1996.
- [3] J. Allan, V. Lavrenko, and R. Papka, "Event Tracking," UMASS Computer Science Department, CIIR Technical Report IR-128, 1998.
- [4] J. Allan, R. Papka, and V. Lavrenko, "On-line New Event Detection and Tracking," *Proceedings of ACM SIGIR*, 37-45, 1998.
- [5] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic Detection and Tracking Pilot Study: Final Report," *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 194-218, 1998.
- [6] L. Ballesteros, W.B. Croft, "Resolving Ambiguity for Cross-Language Retrieval," *Proceedings of ACM SIGIR*, 64-71, 1998.
- [7] B.T. Bartell, "Optimizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval," Ph.D. Thesis, University of California - San Diego, 1994.
- [8] D. Beeferman, A. Berger, and J. Lafferty, "Statistical Models for Text Segmentation," *Machine Learning*, 34: 1-34, 1999.
- [9] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin, "A Statistical Approach to Machine Translation," *Computational Linguistics*, 16(2): 79-85, 1990.
- [10] C. Buckley and G. Salton, "Optimization of Relevance Feedback Weights," *Proceedings of ACM SIGIR*, 351-357, 1995.
- [11] J.P. Callan, W.B. Croft, and S.M. Harding, "The INQUERY Retrieval System," *Database and Expert Systems Applications: Proceedings of the International Conference in Valencia Spain*, 78-83, 1992.
- [12] J. Callan, B. Croft, and J. Broglio, "TREC and TIPSTER Experiments with INQUERY," *Information Processing & Management*, 31(3):327-343, 1994.

- [13] J.P. Callan, "Document Filtering With Inference Networks," *Proceedings of ACM SIGIR*, 262-269, 1996.
- [14] F. Can and E.A. Ozkaran, "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases," *ACM Transactions on Database Systems*, 15(4): 483-517, 1990.
- [15] J. Carbonell, Y. Yang, J. Lafferty, R. Brown, T. Pierce, and X. Liu, "CMU Report on TDT2: Segmentation Detection and Tracking," *Proceedings of the DARPA Broadcast News Workshop*, 117-120, 1999.
- [16] C. Carrick, and C. Watters, "Automatic Association of News Items," *Information Processing & Management*, 33(5):615-632, 1997.
- [17] E. Charniak, "Tree-bank Grammars," Brown University, Department of Computer Science, Technical Report: CS-96-02, 1996.
- [18] E. Charniak, personal communication.
- [19] G. Casella and R.L. Berger, *Statistical Inference*, Duxbury Press, Belmont, CA, 1990.
- [20] C. Cieri, Personal Communication from LDC, 1999.
- [21] P.R. Cohen, *Empirical Methods for Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, 1995.
- [22] W.B. Croft, and D.J. Harper, "Using Probabilistic Models of Document Retrieval Without Relevance Information," *Journal of Documentation*, 5(3): 285-295, 1979.
- [23] W.B. Croft, "A Model of Cluster Searching Based on Classification," *Information Systems*, 5(3): 189-195, 1980.
- [24] D.R. Cutting, D.R. Karger, J.O. Pederson, and J.W. Tukey, "Scatter/Gather: A Cluster Based Approach to Browsing Large Document Collections," *Proceedings of ACM SIGIR*, 298-306, 1992.
- [25] D. Cutting, "Industry Panel Discussion," ACM SIGIR, 1997.
- [26] G. DeJong, "Prediction and Substantiation: A New Approach to Natural Language Processing," *Cognitive Science*, 3: 251-273, 1979.
- [27] S. Dharanipragada, M. Franz, J. McCarley, S. Roukos, and T. Ward, IBM slide presentation by J.S. McCarley at the *The DARPA Broadcast News Workshop*, Herndon, VA, 1999.
- [28] G. Doddington, Presentation Slides at *The DARPA Broadcast News Workshop*, Herndon, VA, 1999.

- [29] J.L. Fagan, *A Comparison of Syntactic and Non-Syntactic Methods*, Ph.D. Thesis, Department of Computer Science, Cornell University, 1987.
- [30] R.P. Goldman, *A Probabilistic Approach to Language Understanding*, Ph.D. Thesis, Brown University, 1990.
- [31] D. Graff, Personal Communication, 1999.
- [32] W.R. Greiff, "Empirical Studies of Query/Document Characteristics as Evidence in Favor of Relevance," *Proceedings of ACM SIGIR*, 11-19, 1998.
- [33] D.K. Harman, "How Effective is Suffixing?" *Journal of the American Society for Information Science*, 42(1): 7-15, 1991.
- [34] D.K. Harman, ed., *Proceedings of Text REtrieval Conferences (TREC)*, NIST Special Publication, 1993-8.
- [35] D. Hawking, and P. Thistlewaite, "Proximity Operators - So Near And Yet So Far," *Proceedings of TREC-4*, 131-144, 1996.
- [36] P.J. Hayes, L.E. Knecht, and M.J. Cellio, "A News Story Categorization System," *Proceedings of the 2nd Conference on Applied Natural Language Processing* (1988), reprinted in *Readings in Information Retrieval*, K. Sparck Jones and P. Willet editors, Morgan Kaufmann Publishing, San Francisco, CA, 518-526, 1997.
- [37] M.A. Hearst, and J.O. Pedersen, "Reexamining the cluster Hypothesis: Scatter/ Gather on Retrieval Results," *Proceedings of ACM SIGIR*, 76-84, 1996.
- [38] D. Hull, *Proceedings of Text REtrieval Conferences (TREC-6)*, NIST Special Publication, 45-67, 1998.
- [39] M. James, *Classification Algorithms*, John Wiley & Sons, New York, 1985.
- [40] H. Jin, R. Schwartz, S. Sista, F. Wall, "Topic Tracking for Radio, TV Broadcast, and Newswire," *Proceedings of the DARPA Broadcast News Workshop*, 199-204, 1999.
- [41] J. Kivinen and M. Wartmuth, "Exponentiated Gradient Versus Gradient Descent for Linear Predictors," UCSC Technical report: UCSC-CRL-94-16, 1994.
- [42] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Proceedings of International Joint Conference on Artificial Intelligence*, 1137-1143, 1995.
- [43] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, 43: 59-69, 1982.
- [44] B. Krovetz, "Viewing Morphology as an Inference Process," *Proceedings of ACM SIGIR*, 191-202, 1993.

- [45] B. Krovetz, *Word Sense Disambiguation for Large Text Databases*, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, 1995.
- [46] A. Leuski, and J. Allan, "Strategy-based Interactive Cluster Visualization for Information Retrieval," to appear in *The International Journal on Digital Libraries*, 1999.
- [47] D. Lewis, *Representations and Learning in Information Retrieval*, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, 1991.
- [48] D. Lewis, and W. Gale, "A Sequential Algorithm for Training Text Classifiers," *Proceedings of ACM SIGIR*, 3-13, 1994.
- [49] D. Lewis, R. Schapire, J. Callan, and R. Papka, "Training Algorithms for Linear Text Classifiers," *Proceedings of ACM SIGIR*, 298-306, 1996.
- [50] L.B. Lombard, *Events: A metaphysical study*, Routledge & Kegan Paul, Boston, MA, 1986.
- [51] S.A. Lowe, "The Beta-Binomial Mixture Model and Its Application to TDT Tracking and Detection," *Proceedings of the DARPA Broadcast News Workshop*, 127-132, 1999.
- [52] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET Curve in Assessment of Detection Task Performance," in *Proceedings of EuroSpeech'97*, 4:1895-1898, 1997.
- [53] P.E. Mayeux, *Broadcast News: Writing & Reporting*, 2ed, Brown & Benchmark Publishers, Guilford CT, 1996, p. 79.
- [54] T.M. Mitchell, *Machine Learning*, McGraw-Hill, Boston, MA, 1998.
- [55] *Proceedings of Message Understanding Conferences*, Morgan Kaufmann, 1988-98.
- [56] R. Papka, J.P. Callan, and A.G. Barto, "Text-Based Information Retrieval Using Exponentiated Gradient Descent," *Proceedings of the 10th Annual Conference of Advances in Neural Information Processing Systems*, 3-9, 1996.
- [57] R. Papka and J. Allan, "On-line New Event Detection using Single Pass Clustering," UMASS Computer Science Technical Report 98-21, 1998.
- [58] R. Papka and J. Allan, "Document Classification using Multiword Features," *Proceedings of ACM International Conference on Information and Knowledge Management*, 124-131, 1998.

- [59] R. Papka, J. Allan, and V. Lavrenko, "UMASS Approaches to Detection and Tracking at TDT2," *Proceedings of the DARPA Broadcast News Workshop*, 111-116, 1999.
- [60] R. Papka, "Learning Query Bias for Improved On-Line Document Classification," *Proceedings of Learning 99*, Snowbird, UT, 1999 (Unpublished).
- [61] J.M. Ponte and W. B. Croft. "Text Segmentation by Topic," *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, 113-125, 1997.
- [62] K. R. Popper, *The Logic of Scientific Discovery*, Harper & Row Publishers, New York, 1968.
- [63] M. Porter, "An Algorithm for Suffix Stripping," *Program*, 14(3): 130-137, 1980.
- [64] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
- [65] F. Provost and T. Fawcett, "Robust Classification Systems for Imprecise Environments," *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 1998.
- [66] E. Riloff and W. Lehnert, "Information Extraction as a Basis for High-Precision Text Classification," *ACM Transactions on Information Systems*, 12(3): 296-333, 1994.
- [67] S.E. Robertson and K. Sparck Jones, "Relevance Weighting of Search Terms," *Journal of the ASIS*, 27(3): 129-146, 1976.
- [68] S.E. Robertson, W. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," *Proceedings of TREC-3*, 109-126, 1994.
- [69] S.E. Robertson, S. Walker, and M. Beaulieu, "Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive," to appear in "*Proceedings of Text REtrieval Conferences (TREC-7)*", NIST Special Publication, 1999.
- [70] J.J. Rocchio, "Relevance Feedback in Information Retrieval," in *The Smart System - Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, N.J., 313-323, 1971.
- [71] W. A. Rosenkrantz, *Introduction to Probability and Statistics for Scientists and Engineers*, McGraw-Hill Publishing, New York, 1997.
- [72] G. Salton, "Relevance Feedback and Optimization of Retrieval Effectiveness," in *The Smart System - Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, N.J., 324-336, 1971.

- [73] G. Salton, "Cluster Search Strategies and the Optimization of Retrieval Effectiveness," in *The Smart System - Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, N.J., 223-242, 1971.
- [74] G. Salton and C. Buckley, "Term-weighting Approaches in Automatic Text Retrieval," *Information Processing & Management*, 24(5): 513-523, 1988.
- [75] G. Salton, *Automatic Text Processing*, Addison-Wesley Publishing Co, Reading, MA, 1989.
- [76] D.L. Schacter, *Searching for Memory: The brain, the mind, and the past*, Basic Books, New York, 1996.
- [77] R. Schank, *Conceptual Information Processing*, Elsevier-North Holland, New York, 1975.
- [78] R. Schapire, Y. Singer, and A. Singhal, "Boosting and Rocchio Applied to Text Filtering," *Proceedings of ACM SIGIR*, 215-223, 1998.
- [79] J.M. Schultz and M. Liberman, "Topic Detection and Tracking using idf-Weighted Cosine Coefficient," *Proceedings of the DARPA Broadcast News Workshop*, 189-192, 1999.
- [80] A. Singhal, M. Mitra, and C. Buckley, "Learning Routing Queries in a Query Zone," *Proceedings of ACM SIGIR*, 25-33, 1997.
- [81] T. Strzalkowski and J. Perez Carballo, "Natural Language Information Retrieval: TREC-4 Report," *Proceedings of TREC-4*, 245-258, 1996.
- [82] R. Swan, J. Allan, and D. Byrd, "Evaluating a Visual Information Retrieval Interface: Asplnquery at TREC-6," presented at *Workshop on Information Exploration at CHI 98*, 1998.
- [83] J. Swets, "Measuring the Accuracy of Diagnostic Systems," *Science*, 240:1285-1293, 1988.
- [84] *Proceedings of the TDT Workshop*, University of Maryland, College Park, MD, October 1997 (Unpublished).
- [85] R.H. Thompson and B.W. Croft, "Support for Browsing in an Intelligent Text Retrieval System," *International Journal of Man - Machine Studies*, 30(6): 639-668, 1989.
- [86] H. Turtle, *Inference Networks for Document Retrieval*, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1990.
- [87] H. Turtle and W.B. Croft, "A Comparison of Text Retrieval Models," *The Computer Journal*, 35(3): 279-290, 1991.

- [88] E. Tzoukermann, J.L. Klavans, and C. Jacquemin, "Effective Use of Natural Language Processing Techniques for Automatic Conflation of Multi-Word Terms: The Role of Derivational Morphology, Part of Speech Tagging, and Shallow Parsing," *Proceedings of ACM SIGIR*, 148-155, 1997.
- [89] C.J. van Rijsbergen, *Information Retrieval, 2ed.*, Butterworths, London, 1979.
- [90] E. Voorhees, *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval*, Ph.D. Thesis, Department of Computer Science, Cornell University, Ithaca, N.Y., 1985.
- [91] E. Voorhees, "Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness," *Proceedings of ACM SIGIR*, 315-323, 1998.
- [92] F. Walls, H. Jin, S. Sista, and R. Schwartz, "Topic Detection in Broadcast news," *Proceedings of the DARPA Broadcast News Workshop*, 193-198, 1999.
- [93] S. Wegmann, P. Zhan, I. Carp, M. Newman, J. Yamron, and L. Gillick, "Dragon Systems 1998 Broadcast News Transcription System," *Proceedings of the DARPA Broadcast News Workshop*, 277-280, 1999.
- [94] P. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing & Management*, 24(5): 577-597, 1988.
- [95] J. Xu and W.B. Croft, "Query Expansion Using Local and Global Document Analysis," *Proceedings of ACM SIGIR*, 4-11, 1996.
- [96] J. Xu, *Solving the Word Mismatch Problem Through Automatic Text Analysis*, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, 1997.
- [97] J. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt, "Topic Tracking in a News Stream," *Proceedings of the DARPA Broadcast News Workshop*, 133-138, 1999.
- [98] Y. Yang, T. Pierce, and J. Carbonell, "A Study on Retrospective and On-Line Event Detection," *Proceedings of ACM SIGIR*, 28-36, 1998.
- [99] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," to appear in the *Journal of Information Retrieval*, 1999.
- [100] C. Zhai, P. Jansen, E. Stoica, N. Grot, and D.A. Evans, "Notes on Threshold Calibration in Adaptive Filtering," to appear in *Proceedings of Text REtrieval Conferences (TREC-7)*, NIST Special Publication, 1999.
- [101] J. Zobel and A. Moffat, "Similarity Measures Explored," Department of Computer Science, RMIT Technical Report: CITRI/TR-95-3, 1995.