

SEQUENTIAL PATTERNS AND TEMPORAL PATTERNS FOR TEXT MINING

By

Apirak Hoonlor

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: COMPUTER SCIENCE

Approved by the
Examining Committee:

Dr. Boleslaw K. Szymanski, Thesis Adviser

Dr. Mark Goldberg, Member

Dr. Mohammed J. Zaki, Member

Dr. William A. Wallace, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2011
(For Graduation August 2011)

© Copyright 2011
by
Apirak Hoonlor
All Rights Reserved

CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vii
ACKNOWLEDGMENT	x
ABSTRACT	xii
1. Introduction	1
1.1 Extracting Features from Text Document	3
1.1.1 Sequential Patterns in Text Mining Applications	6
1.1.1.1 Surface Text Pattern	8
1.1.2 Pattern Extraction for Online Document Sequences	11
1.2 Thesis Outline	14
2. Background and Related Works	15
2.1 Text Mining and Its Applications	15
2.1.1 Pre-processing	16
2.1.1.1 Text Pre-processing	16
2.1.1.2 Document Indexing	21
2.1.2 Pattern Mining and Results Evaluation	26
2.1.2.1 Categorization	26
2.1.2.2 Clustering	28
2.1.2.3 Information Retrieval	30
2.1.3 Text Mining Applications	32
2.2 Sequential Pattern Mining	33
2.3 Concepts of Burstiness in Text Mining and Related Works	36
3. Recursive Data Mining	39
3.1 Introduction	39
3.2 Preliminaries	41
3.3 Recursive Data Mining	42
3.3.1 General Framework	43
3.3.2 Preprocessing	44
3.3.3 Pattern Generation	45

3.3.4	Pattern Significance	46
3.3.4.1	Chi-square Test	47
3.3.4.2	Document Frequency Test	47
3.3.4.3	Information Gain Test	48
3.3.4.4	Mutual Information Test	48
3.3.4.5	Unigram Test	49
3.3.5	Dominant Patterns	50
3.3.6	Settings parameters for RDM	53
3.4	Recursive Data Mining for Text Categorization	54
3.4.1	Training Phase	54
3.4.2	Testing Phase	54
3.5	Experiments and Results	55
3.5.1	Data Preparation and Experimental Setup	56
3.5.2	Comparative Study of Significance Tests	58
3.5.3	Sequence Manipulating Step	59
3.5.4	Performances of RDM	63
3.5.5	Effect of Parameter Changes	67
3.6	Conclusion	68
4.	Burstiness in Text Mining	71
4.1	Introduction	71
4.2	Preliminaries	71
4.3	Burst Detection	72
4.4	Temporal Correlated Term Detection	73
4.4.1	Temporal Correlated Term on ACM Dataset	78
4.4.1.1	The Bursty Words in the Bursty Period	79
4.4.1.2	Terms with high Temporal Bursty Correlation Score	82
4.5	Burst Detection for Document Clustering	82
4.5.1	Bursty Features for Document Clustering	85
4.5.1.1	Kleinberg’s Algorithm:	86
4.5.1.2	Discrepancy Model of Burstiness (Max-1):	86
4.5.2	Bursty Distance Measurement	87
4.5.3	Experiments	89
4.5.3.1	Synthetic Dataset	91
4.5.3.2	News Article Data sets	92

4.5.3.3	Additional Real-life Datasets	96
4.6	Conclusion	98
5.	Conclusions and Future work	100
5.1	Burstiness for Recursive Data Mining	101
5.2	Surface Text Patterns	101
	BIBLIOGRAPHY	104

LIST OF TABLES

2.1	Results on using stopwords and stemming on Text categorization base on F-score. We compare across two types of features: words and bi-grams. For each feature, we use occurrence (binary) and number of occurrences (frequency) as feature values.	20
2.2	Document clustering using K-mean clustering	29
2.3	Document clustering using density based clustering	29
2.4	A term-document incidence matrix, where matrix element in row i and column j is 1 if an actor, or actress, in row i appears in the movie in column j	31
3.1	Dataset for Role Identification task	57
3.2	Comparison experiment of five significance tests: Unigram test (Uni), Document Frequency test (DF), Information Gain test (IG), Mutual Information test (MI), Chi-Square test (Chi). The F-measure value in bold font is the highest F-measure of the corresponding role.	58
3.3	Comparison experiment of three abstraction level of input sequence: sequence of characters, sequence of words, and sequence of clustered words. The F-measure value in bold font is the highest F-measure of the corresponding role.	61
3.4	The performances of RDM, Bigram and Trigram models on binary classification of the role identification tasks. The f-scores are used as the evaluation means	64
3.5	Results of paired t-test. RN stands for RDM with NB and RS stands for RDM with SVM	65
3.6	Results of paired t-test. RN stands for RDM with NB, NB3 stands for NB model using approximated sequential pattern of length 3, and NB4 stands for NB model using approximated sequential pattern of length 4	66
4.1	Top 10 bursty correlated words, listed in order of the bursty ranking, in the burstiest period of the 10 most frequent words.	81
4.2	The temporal correlated term in the first four-bursty years of five most frequent keywords.	83
4.3	Synthetic Dataset Syn1	92

4.4	Synthetic Dataset Syn2	93
4.5	Synthetic Dataset Syn3	93
4.6	Synthetic Dataset Syn4	94
4.7	RSS feed dataset (Entertainment news 03/01/2009 - 08/31/2009 and Political news 10/01/2008 - 12/31/2008)	95
4.8	20 Newsgroup Dataset	97
4.9	Reuters Dataset	98
4.10	Nature Dataset	98
4.11	ACM Dataset	99

LIST OF FIGURES

1.1	Ask.com - The screen shot of Ask.com with “When was Albert Einstein born?” as the input query.	9
1.2	Ask.com - Q&A system. The screen shot of current Beta version of Q&A system at Ask.com.	10
1.3	Google Trends: The screen shot captures the result of query “Hillary Clinton” on Google Trends.	12
1.4	BlogPluse	13
2.1	Example of the bag-of-word text representation with the occurrence of word as feature value.	21
2.2	Example of the bag-of-word text representation with the number of occurrence as feature value.	24
2.3	Two-state finite machine to detect the bursty period of a given term in the data collection.	37
3.1	Pattern Generation Step. ($l_w = 6, max_gap = 2$)	46
3.2	Sequence Re-writing Step	51
3.3	Removing Noisy Tokens for Long Range Patterns	52
3.4	Comparison of three word clustering methods on various size of clusters for RDM applications on role identifications. WC stands for distribution clustering, POS stands for part-of-speech tagging and FR stands for frequency clustering. RN and RS stand for RDM with NB and RDM with SVM respectively.	63
3.5	Binary Classification – RMSE Comparison	66
3.6	Classification Probability over Unseen Message Folder	68
4.1	The burstiness values of “Aspiration == std” and “Aspiration == turbo” queries on the automobile data set using horse power to arrange, on the x-axis. For y-axis, it is the burstiness, which we defined using probabilities.	74
4.2	The burstiness values of “fuel type == diesel” and “fuel type == gas” queries on automobile data set using horse power to arrange, on the x-axis. For y-axis, it is the burstiness, which we defined using probabilities.	74

4.3	Number of search queries input to Google: ‘Hillary Clinton’, ‘Hillary Clinton’ and ‘Secretary of State’ (HC SS), ‘Hillary Clinton’ and ‘Senator’ (HC Senator), and ‘Hillary Clinton’ and ‘Candidate’ (HC Candidate)	75
4.4	Burstiness score, using Equation 2.3, of the queries: ‘Hillary Clinton’, ‘Hillary Clinton’ and ‘Secretary of State’ (HC SS), ‘Hillary Clinton’ and ‘Senator’ (HC Senator), and ‘Hillary Clinton’ and ‘Candidate’ (HC Candidate)	77
4.5	Temporal bursty correlate scores of various search query following Equation 4.1	78
4.6	The number of records found each year from year 1990 to 2010 on ACM datasets.	79
4.7	Numbers of articles in San Francisco Call newspaper containing words ‘victoria’ and ‘death’ per month from Jan. 1900 to Dec. 1901.	85
5.1	The Q&A framework that automatically generated surface text patterns from the provided questions.	103

ACKNOWLEDGMENT

I have received supports from many people throughout my years as a Ph.D. student at RPI. First, I sincerely thank my advisor, Professor Boleslaw K. Szymanski for his continuous support. He has always prepared me not only for my success in a Ph.D. program, but also for success in my future research work. He has inspired and encouraged me throughout my graduate program at RPI. He opened window of opportunities for me to think critically about research ideas, to test our hypotheses, and to effectively present them. He always has insightful suggestions to our research. Finally, I like to thank him for sharing his interesting stories from history, literature, science, and his personal experiences during our meetings – they were nothing short of amazing.

I would also like to express my gratitude to Professor Goldberg, Professor Wallace, and Professor Zaki for agreeing to be part of my committee. I had the wonderful opportunity of being in the ONR research group with Professor Goldberg, and Professor Wallace. They always have interesting ideas and suggestions for the group. I especially enjoyed working for Professor Goldberg as the teaching assistant for the Computability and Complexity class. It has been a wonderful experience. I also thank Professor Zaki for his involvement with our research. With his research ideas and knowledge, he has been a key contributor to our research. I had a pleasure of taking his Data Mining class. It has been a very informative class, and I learned a great deal from it.

I also like to thank all my friends in the Computer Science Department. I need to give special mentions to Vineet, Jierui, Konstantine and Hasan. I have enjoyed working with Vineet Chaoji on many projects. He had substantial contributions in RDM project. He has been great collaborators on other projects with great research ideas, validation and presentation approaches. I especially enjoyed our long friendly meeting with discussion varying from class, research, to life in general. I also have enjoyed exchanging ideas and working with Jierui Xie. He has helped me with many of my research problems, ranging from Matlab source codes, to designing

algorithms. It was always a pleasure having him as a classmate, officemate, and friend. Konstantin Mertsalov and Mohammad Al Hasan also provided contributions to my research. Konstantin has provided me with his datasets, and interesting ideas in my research. Hasan has always been there to help whenever I need supports with his programs and research. I would also like to acknowledge other colleagues and friends - Medha, Cagri, Geng, Jiao, and Sahin. I would also like to thank the Computer Science Department, its faculty as well as its staff, especially Terry Hayden and Chris Coonrad. My graduate life in the department and the graduate school would have been miserable without their efforts.

I would like to express my gratitude to all friends in the US, especially for Anongpat and my RPI Thai students group: Charn, Chinawut, Natt, Natthapong, Panitkwan, Pilasinee, and Praowpan for their continuous supports. They have been a strong support through all my ups and downs in life. I would like to thank Thai government, for providing me with the DPST scholarship for my undergraduate and my early graduate educations. Without this scholarship, I would never be where I am now. I also acknowledge partial support through the ONR Contract N00014-16-1-0466 and the Army Research Laboratory Cooperative Agreement Number W911NF-09-2-0053. However, the views and conclusions contained in this document are mine and should not be interpreted as representing the official policies, either expressed or implied, of the Thai Government or the U.S. Government, no official endorsement should be inferred or implied.

Lastly but most importantly, I would like to thank my family members. Especially, my mother (Sulux), my aunt (Karunee), my uncle in law (Charoen), and my grandparents (Leard and Sai), without their supports, I would not have traveled half way around the world to study since my high school year.

ABSTRACT

With the current growth rate of URLs, as a community, we are at the age of online information overload and for many other domains, such as Internet, web services, data analysis, and the like – they have been for quite sometimes. Text mining has been a key research topic for online information retrieval and information extraction. In this thesis, we studied two concepts in pattern extraction for text mining tasks: sequential pattern mining and bursty information.

In sequential pattern mining, our interests stemmed from a text mining problem of recognizing a group of authors communicating in a specific role within an Internet community. The challenge is to recognize possibly different roles of authors within a communication community based on each individual exchange in electronic communications. Depending on the exchange parties, the message can vary in length, contain different style of writing, and contain multiple topics, making the standard text mining approaches less efficient than in other applications. An example of such a problem is recognizing roles in a collection of emails from an organization in which middle level managers communicate both with superiors, subordinates and among themselves.

For this problem, we present *Recursive Data Mining* (RDM), a sequence pattern mining framework for text data. RDM allows certain degree of approximation in matching patterns – necessary to capture non-trivial features in text datasets. The framework minimizes the size of the combinatorial search space using statistically significant tests, such as information gain, mutual information, and minimum support. RDM recursively and hierarchically mines patterns at varying degrees of abstraction. From one abstraction level to another, the framework removes “noisy” tokens to allow long range patterns to be discovered at higher levels. We used a hybrid approach, in which the RDM discovered patterns are used as features, to build a classifier. We validated RDM framework on the role identification task using Enron email dataset. Specifically, we used RDM to categorize the senders of email content into their roles in Enron as CEO, Vice-President, Manager and Traders. The results

showed that a classifier that used the patterns discovered by Recursive Data Mining performs well in role identification.

Our interests in the concept of bursty information originated from the temporal nature of social roles of each individual. Over a life time, a person can be associated with variety of roles ranging from a leader of an arm group, a prisoner, a Nobel Prize winner to a president of a country. Such information is often embedded in, and can be extracted from, the temporal text patterns associated with an individual. Previously, the term frequency was the main quantifier for trend analysis and visualization. In recent years, due to their abilities to detecting changes in patterns, the concepts of bursty information have been used to extract temporal patterns from text streams. We proposed two complementary burstiness frameworks to extract temporal correlated patterns from text stream. The first framework is proposed for the extraction of bursty patterns in the bursty period of a given pattern. The second framework is proposed for the extraction of temporally correlated patterns at each time steps. We use these frameworks to analyze ACM dataset. Specifically, we used them to find out the following: (i) when certain research topics received high interests from Computer Science research communities, and during which time, what were the related topics often associated with them, (ii) for each topic, which other topics are temporally correlated with it at a given time.

As we studied the burstiness concepts, we realized that they can be applied to other text mining tasks. We proposed a *bursty distance measurement* for creating a distance matrix in text clustering task. We experimented with our framework on synthetic data, online news article data, and additional real-life datasets. The experiments showed a substantial improvement on event-related clustering on online news article data for our framework. Also, our framework generally performed better than other existing methods. In the future, we intend to embed the bursty information into our RDM framework and use it to trace the changes, or lack thereof, in the sequential patterns and hierarchical structures of text streams. We also want to see if we can incorporate more information into sequential patterns using the idea of surface text pattern.

CHAPTER 1

Introduction

There is at least one piece of information in every document. Moreover, a group of similar documents also contain collective information, just as a group of movie reviews can tell how the public feels about certain movies. The process of extracting such information from text documents is often referred to as text mining. As its importance grew, the research on text mining developed impressively over the last decade, drawing researchers from the areas of computational linguistics, data mining, information retrieval, machine learning, and statistics. Besides the general data mining tasks such as categorization and clustering, text mining tasks extend to document retrieval, document summarization, entity extraction and modeling, information extraction and tracking, and sentiment analysis. Text mining applications are used by everyone in many areas ranging from academic research, Biomedical, and marketing applications to spam mail blocking. The prominent field of application for text mining is the Internet. Since its introduction, people have started posting electronic documents on the web. In 2008, Google reported in an Official blog that they processed at least 1 trillion unique URLs [111]. With the current growth rate of online information, all aspects of text related applications have become the important tools to help extract online information, handling and organizing electronic documents, searching and ranking documents, etc. In this thesis, we study and present sequential pattern extraction and temporal pattern extraction methodologies for online text documents. In particular, we propose *Recursive Data Mining* framework to extract sequential patterns and use the definition of *burstiness* to extract temporal patterns from online text documents. We also show how we use these

Portions of this chapter previously appeared as: V. Chaoji, A. Hoonlor, and B.K. Szymanski, "Recursive data mining for role identification in electronic communications," *International Journal of Hybrid Information Systems*, vol. 7, pp. 89–100, Apr. 2010.

Portions of this chapter previously appeared as: A. Hoonlor, and X. Zhu. (2006, Apr.). Unsupervised learning for surface text recovery in QA system, Computer Science Department, University of Wisconsin - Madison, WI. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.4787>, (Date Last Accessed on Jun. 2, 2011)

concepts on text mining tasks for online text documents.

Similar to data mining, the text mining tasks can be viewed as a three-step process: preprocessing, pattern mining, and result evaluation [48, 115]. The preprocessing step consists of preprocessing raw texts and representing each text in a form of feature vector. A learning algorithm cannot process the raw electronic document directly. Thus, we must perform document indexing procedure that maps a raw text document into a feature vector format. Prior to document indexing, a text document often contains words that can lead to lower performance in a learning model, such as misspelled words, abbreviated words, and words with and without stemming. This process is referred to as text preprocessing. After we perform the preprocessing on the electronic documents, we need to define a set of features to represent the text documents. A *feature* refers to an attribute which represents a certain characteristic of the data. By looking at a text document, we can think of several characteristics that we can use as features to represent the text document – a list of all words, a list of the selected keywords, and a list of phrases in the text, etc. By extracting every feature for every document, we create a feature vector space for the text corpus. Next, we extract informative patterns during the pattern mining process using various text-learning algorithms depending on the intended task. These algorithms analyze the data, and present the extracted information. Finally, in the evaluation step, depending on the tasks, the visualization on data, and the retrieved documents are return as the extracted information. In the next chapter, each of the three processes is discussed in detail.

The challenge in text mining problems is the size of the dataset and the result qualities. In the first, the numbers of documents can be so large that an ad-hoc manual approach is either requiring too much processing time, or simply not feasible due to limited resources. For example, in web categorization, with the 1 trillion unique URLs, even if a human expert can tag one URL per 5 seconds, it would take roughly 159 years of non-stop work to complete the task. Of course, that number is calculated with the assumption that no new URL will be generated in that time frame. The second challenging task is the result qualities of the text mining jobs, which are important since they can potentially affect the user satisfaction of millions

of people.

One good example for the previous challenges is the search engine where the users always want to get the search results as fast as they can. In the recent event on May 14th 2009, Twitter's services were filled with complaints and comments regarding Google search engine which had become extremely slow, or even inaccessible, for millions of users around the globe [99]. If Google search engine becomes even a few seconds slower in presenting results, users will likely use other search engines which present results faster with similar search qualities. On a smaller scale, consider a text categorization engine engaged in a newsgroup host to recommend news articles to its million subscribers; a bad categorization job would recommend the wrong articles, resulting in enormous user dissatisfaction. Next, we provide brief statements on feature extraction problems and our contributions to improve the qualities of the text mining tasks.

1.1 Extracting Features from Text Document

The problem of understanding data and its characteristics has attracted keen interest of scientists from early years. Specifically, we are interested in the characteristic that captures a certain "style" that discriminates between different categories of data. These characteristics have been widely used to analyze and create the underlying generative model for each data category. The data is characterized in term of a set of *features*, also referred to as patterns or attributes. The definition of a feature is closely tied to the nature of the data. For example, for text data set, a feature can include keywords. For high dimensional genetic datasets, the principle eigenvectors can serve as the features. Within a pattern recognition system, *feature extraction* identifies features relevant to the application at hand. Feature extraction broadly consists of two sub-tasks – *feature construction* (e.g. PCA, LSI, etc.) and *feature selection* [39], each addressing one of the two main challenges within feature extraction. The first challenge arises from the presence of a large amount of noise in the data which results in construction of ineffective features. The second challenge arises from the large number of features usually generated in the construction process. Feature selection methodologies are applied to rank features based on

an optimality criteria – such as information gain, kernel and novelty detection – and just the top-ranked features are used to filter out noises, avoid the curse of dimensionality, and enhance generalization capabilities [31].

With the proliferation of communication technology in the form of mobile technologies and email, data generated through human interaction can be effectively collected. Human interactions are unique in the sense that every interaction carries not only semantic content defining its meaning but also a unique word and pattern-of-words structure characteristic of its author (or speaker). In addition to the noisy and high dimensional [51] nature of human interaction data (which is common to any real life data), it is plagued by ambiguities introduced by the language. These challenges underline the need for effective feature extraction for tasks such as *authorship assessment*. As the name suggests, authorship assessment is a form of text categorization which aims to determine the author of a fragment of human interaction. Note that authorship assessment assumes that the communicating “style” of an author is time and audience invariant. In other words, the implicit cues in the communication do not change based on the entity at the receiving end of the communication.

In document pre-processing step, choosing the set of relevant features for a specific text mining tasks is a very important and difficult task. In the early stage of text mining, words are often used as features [49, 54]. We called such an approach *vector space model* (VSM), also known as the *bag-of-word* representation (explained in the next chapter). Figure 2.1 illustrates how a document is represent using a bag-of-word style. Depending on how the values of each word feature is assigned, the bag-of-word document representation can be applied in many text mining tasks. In [93], a Bayesian network model using simple word features is able to classify a junk email with high precision and recall (97.1% and 94.3% respectively). Although VSM is a very robust and effective document representation, there are limitations.

The first problem with VSM using words as feature is that it ignores the information imbedded in the order of words. For example, following the bag-of-word style, the sentences “A cat chases a rat” and “A rat chases a cat” have the same exact bag-of-word representation “a-2, cat-1, chases-1, rat-1”. However, the mean-

ings of the two sentences are different. The applicable applications based on word features are limited. Although text mining applications using a bag-of-word style are found to be successful in [49, 82], it cannot be applied to applications such as machine translation, automatic summarization, analysis of information flow, and speech recognition. For example, an automate-natural-spoken-dialog application often encounters sounds which cannot be converted correctly to a word. If the speaker said “I’d like to make a collect call”, but the sound of the last word “call” tailed off, a system using phrases as feature could still make an accurate guess that the last word that follows “I’d like to make a collect...” is “call”. However, if the word order was ignored such that the system only saw a single word, “collect”, the system would be less likely to make a correct guess that “call” was the following word. The solution proposed to attack this problem was the “sequential pattern” extraction. A sequential pattern, a sequence of ordered items, is used to preserve and capture the information from the ordered of the words. Many language models have been proposed for extracting sequential-pattern-based features from documents. In [113], statistical information of phrases and correlated words are used as features for text mining task. In [45], full logical translation of the document is used to create the relations of all objects, subjects and actions found in the document. Charniak illustrated that the combination of context free grammar and statistical information can be used to create a parser that eliminates ambiguities in sentences [17]. Logic and grammatical based language models are often used in information extraction task where the understanding of sentence provides crucial information in the documents [11, 17, 45].

The second problem of VSM is that the current feature values used for document representation, such as binary value, term-frequency, and term-frequency and invert document frequency, are not optimized for *text stream*. The text stream is the sequence of documents sorted according to time. Examples of text stream are Blog posts, emails, online news articles, RSS feed and Tweets. Kleinberg [60] introduced “bursty” concept as a quantifier for temporal patterns in text stream. In [42], the bursty information is included as the weight for feature values for document representation. This framework is called “bursty feature representation”. He et al.

showed that the bursty feature representation improved the performance of document clustering [42]. In [62], the word features are used to create document index for burst-aware search. The burst-aware search is able to retrieve the news article related to the given events with high precision. In the past several years, researchers in text mining community have shown increasing interests on both issues.

Next, we introduce the concept of sequential patterns extraction for capturing the imbedded information in the sentences. We also discuss the applications of sequential patterns mining in the surface patterns extraction from online documents. Finally, we introduce the feature extraction on online documents sequences using bursty information as an alternative solution of the second problem.

1.1.1 Sequential Patterns in Text Mining Applications

In the past several years, the sequential patterns have gained interests in text mining community. The key reasons of using sequential patterns based approach for text mining applications are as follows:

1. As compared to a feature based on single words, a feature based on sequence pattern captures temporal relationships between words and phrases.
2. A language model based on sequential patterns has more expressive power than those based on words.
3. A sequence pattern based approach allows one to control the level of flexibilities in the feature extraction process.
4. While checking for the presence of a feature (pattern in this case), a score reflecting the degree of matching can be assigned. Non-pattern based methods assign a strict binary score based on the presence or absence of the feature. This is specifically important for data that is affected by noise in the form of spelling mistakes, use of abbreviations, etc.

As previously mentioned, there are more and more text documents posted on the website everyday. The problem is that the text documents written in any language. Without a perfect-automated-universal language translation machine,

text mining applications are either domain specific or requiring domain knowledge on languages that the documents are in [28, 45]. The advantage of language models such as N-gram language model, Hierarchical pattern model, and Sequential pattern model is the minimum knowledge requirement of the natural language [80, 101, 109]. Some languages, such as Japanese and Thai, words are written without any white space in between. Thus, the common requirement is the tokenization method to extract sequence of word from a document. Although, one can forgoes such requirement by use other characteristics of the documents. Keselj et al. explored the idea of using the N-gram model on sub-word level in text mining problem [58]. By creating N-gram model on the byte level of the document, they were able to extract patterns without any prior knowledge of the language of the document.

On the other hand, there are useful language applications such as WordNet, EuroNet, Part-of-speech tagging, etc. that can enhance text mining performances. In [33], WordNet has been used to improve text retrieval. One of Google projects is to use word clustering application to automatically expand the user's search query to multiple queries of the same meanings. In Google Sets [37], upon receiving the query "cooking class", the search engine will display the results of the similar items such as "cooking school", "culinary school" and "cooking course".

In this thesis, we present an alternative framework called "Recursive Data Mining" (RDM) for extracting features from text documents. The original term of Recursive Data Mining is introduced in [103] as a tool to extract statistical information from sequential users' command for masquerade detection. The idea of capturing hierarchical pattern from the sequence of tokens is a useful for text mining. However, the actual presented features and the extracted sequential patterns recovered in that work are not suited for text mining task for the following reasons. First, the set of features used in [103] do not have discriminating power for applications such as text clustering and text categorization. Second, the sequential patterns found by [103] do not allow approximate matches (see next chapter for all formal definitions of patterns). As the text document often contains typo, abbreviation, synonyms, etc., the "approximated pattern" is a very useful and, in many cases, crucial concept in text mining. For example, the pattern "President Obama" is not

an exact match of the pattern “President Barack Obama”. However, both patterns refer to the same person.

To extract approximated sequential pattern, we made the adjustments to RDM and re-introduced RDM as a language model, and novel sequential pattern mining framework for text mining applications in [15, 16]. The key contributions of RDM approach made in our works are as follows:

- While the other published techniques work on a fixed size window, our framework allows arbitrary size patterns to be discovered.
- RDM can discover approximate patterns from text documents.
- RDM uses statistical techniques, such as log likelihood ratio tests, information gain, and mutual information methods to identify significant patterns.
- RDM can capture patterns at various levels of abstraction, as well as “long range patterns”, using its hierarchical model.
- RDM does not only require little domain knowledge, but it also allows language tools such as part-of-speech tagging, and word clustering to enhance performance.

1.1.1.1 Surface Text Pattern

Usages of sequential patterns are not limited to classical task in text mining such as document classification. Sequential pattern can be used to extracted information for question answering system. Sometimes users search the data banks, and web, not looking for the documents, but for the specific information that such documents contain. For example, one might just want to know “When was Abraham Lincoln born?”, or “How long is Mississippi river?”. The question answering system accepts such questions and returns the corresponding answers in natural language format. Ask (Ask.com) is an online search engine website that also provides an open domain question answering system. As its named suggested, the user can input question as a normal search query. For example, if we submit the query “When was Albert Einstein born?”, Ask would search, extract and provide the answer as

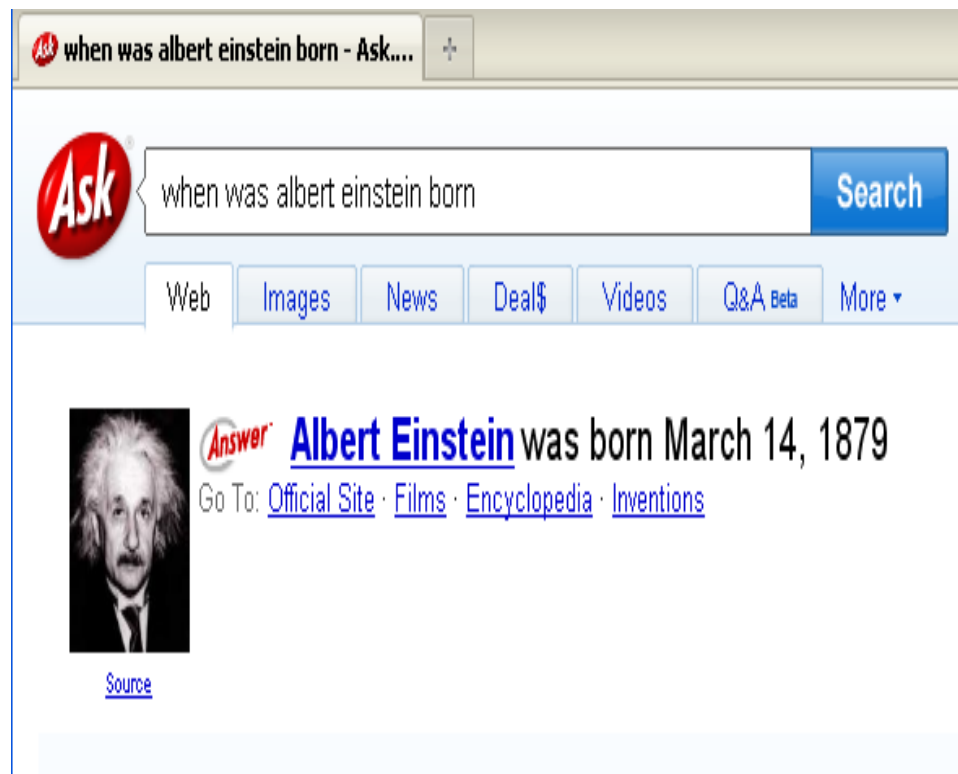


Figure 1.1: Ask.com - The screen shot of Ask.com with “When was Albert Einstein born?” as the input query.

shown in Figure 1.1. Google has also included similar system in its search engine, where the answer of the question is also provided at the top of the search page. Recently, Ask introduced a version of its new question and answering system. Instead of simply search for URLs of the topic related to the question, the new system of Ask utilizes its search results on the user provided answers [5]. Figure 1.2 shows the screen shot of this new system.

To build a question answering system, one often need to use the applications from information retrieval, information extraction, machine learning, language modeling and data mining. Thus, question answering system has received attentions from these fields, and became a testing ground of how one can combine all these applications. More recently, the open domain system used external information extraction methodologies to find the answer. One of these methodologies is surface text patterns - a special type of sequential pattern which composes of words and special user defined tokens. It has been shown by several question answering sys-

The screenshot shows the Ask.com Q&A system interface. At the top, there is a search bar with the question "who is hillary clinton" and buttons for "Search Q&A" and "Search the Web". Below the search bar, there are navigation tabs for "Web", "Images", "News", "Deal\$", "Videos", "Q&A Beta", and "More". The main content area is titled "Topic: Who Is Hillary Clinton?" and includes a prompt: "Not finding your answer? Try searching the web for **who is hillary clinton**".

On the left side, there are sponsored results for "News on Hillary Clinton" from YellowPages.com, "What are they building?" from conezones.com, and "Hillary Clinton" from russiatoday.com. On the right side, there are "Related Topics" such as "Hillary Clinton for Pre: No to Hillary Clinton", "New York State Senatc", "Who's Running for Pres Presidents", "Famous Women", "Hillary Rodham Clinto", "Barack Obama", "John Edwards", "Senator Hillary Clinton", "Bill Clinton", and "Rudy Giuliani". There are also "Reference Topics" on Reference.com, including "Hillary Clinton for Pres Biography Hillary Clint", "Hillary Clinton Accom", "Hillary Clinton Gallery", "Hillary Clinton's Politic", "Democratic Party Platfc", "New York State Senato", "Who's Running for Pres", "Hillary Rodham Clinto", and "Barack Obama".

The "Top Answers" section contains three answers:

- Answer 1: "Born: 26 October 1947 - Birthplace: Chicago, Illinois - Best Known As: The former First Lady who ran for president - Name at birth: Hillary Diane Rodham Hillary Rodham Clinton was elected to the U.S. Senate in 2000. She is also the ... http://wiki.answers.com/Q/Who_is_Hillary_Clinton See entire page >
- Answer 2: "He has been reduced to a nobody is an insignificant office. Obama could not have planned it better. He will have no competition when he runs for reelection. <http://answers.yahoo.com/question/index?qid=2009071311...> See entire page >
- Answer 3: "This is sad. You posted your hateful opinion, but were unable to parse and research the facts of Simon's piece. Hillary's background has been in supporting women and children. These are facts - not opinions - do your homework before you ..."

Figure 1.2: Ask.com - Q&A system. The screen shot of current Beta version of Q&A system at Ask.com.

tems that, for certain types of questions, answers can be extracted from documents using such sequential patterns. For example, for the birth-day-type question “When was X born?”, the typical answers can be found in sentence such as “Lincoln was born in 1809”, and “Benjamin Franklin - Date of Birth: 17 January 1706”. These sentences suggest that the patterns “< NAME > was born in < ANSWER >” and “< NAME > - Date of Birth: < ANSWER >” can be used to find the correct answers. The surface text pattern has done surprisingly well in pinpointing the answer. In previous work by Ravichandran and Hovy [89] and Dumais et al [30], the surface text patterns learning is setup as the offline semi supervised learning with the bootstrapping method, which is required the uses of training data. In this thesis, we discuss an ongoing research of an automate system which recovers the surface text pattern from the training corpus for information extraction purposes.

1.1.2 Pattern Extraction for Online Document Sequences

Many data types are naturally in sequence – brainwave, sale transaction. Moreover, any timestamp type dataset can be considered as sequential data set. Detecting irregularity in sequential data set is not something new. An irregularity pattern can provide very crucial information depending on this type of the data sets. For example, irregularity in brainwave defined as a spike in the electroencephalography can imply sleeping, experiencing pain, or brain death of a person. In [114], Wilson and Emerson provided a good review and comparison studies of various spike detection methods to help electroencephalographers find spikes faster. In credit card fraud detection, detecting abrupt changes of pattern in large data sets can flag a suspicious transaction which in turn helps detecting credit card fraud. Curry et al. introduced a novel method in [25] that can detect changes on payment transaction of credit card.

By presenting text documents in sequence according to their timestamps, we can consider them as document sequence. The detection of changes of pattern in document sequence is referred to as burst detection. We are not only interested in the sudden-change in patterns of words' trends, but also in how the relationship between two words changes over time. For example, in case of Hillary Clinton, her changes in political status can be observed from the changes in temporal patterns related to her name – from 1993 to 2001, her name was associated with the first lady of the United States, from 2001 to 2009, she was referred to as the United States Senator for New York, and since 2009, she has had the title of the United States Secretary of State next to her name.

In recent years, for information retrieval and information search applications, the burst detection from queries is an issue of interests. There are many applications which try to capture such burst.

1. Google Trends: Upon receiving a user to query and time period of interests, Google Trends displays how frequently the query appears in Google news as well as how many times it was searched in those time periods [38]. It also lists the news that appears on the local maximum peak of search volume index in the interested time period. Figure 1.3 displays the results of query “Hillary

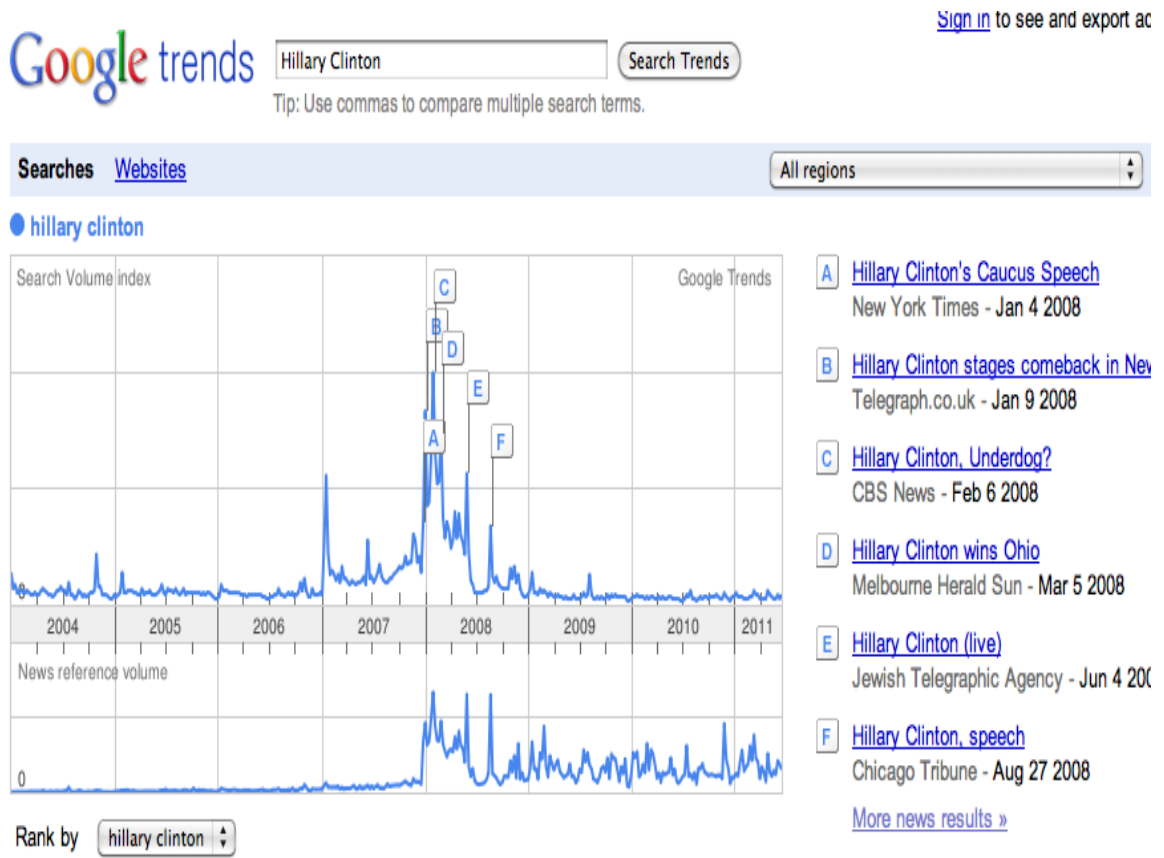


Figure 1.3: Google Trends: The screen shot captures the result of query “Hillary Clinton” on Google Trends.

Clinton” on Google Trends. Note that, Google Trends listed on the right the news of Hillary Clinton during her presidential campaign, but it fails to notice the spike in January 2007 where she announced the formation of a presidential exploration committee for the presidential election in 2008. Google Trends is a very useful tool in detecting events, and users’ interests over a given period of time.

2. BlogPulse: BlogPulse is an analyzing tool that detects the trends of phrases in the user blog [35]. Its functionalities regarding the trends are very similar to that of Google Trends, but BlogPulse focuses on the posts from Blogs. It provides top bursty phrases, and bursty new story of the days. It also keeps track of the usages of the phrases over times. Upon search query, BlogPulse can give the Trend search results as well as the conversation tracker of the

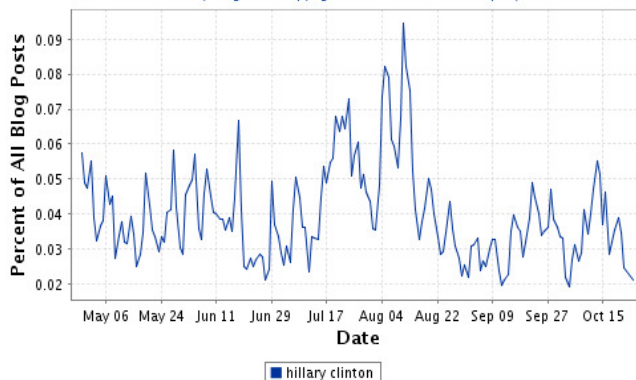


Tools Overview
Trend Search
Featured Trends
Conversation Tracker
BlogPulse Profiles

Home > Tools

Trend Search

Generated by BlogPulse Copyright 2009 The Nielsen Company.



➤ **Search results:**
[hillary clinton](#)

➤ **Drill:**
 Hover over lines on the click through to date-sp

➤ **Want more?**
 Nielsen Online's [MyBuz](#) tool analyzes all forms of generated media (CGM newsgroups, message discussion forums. [See or more information](#), or free at 1-888-634-1222

➤ **BlogPulse Trend Search**

① Trend Search Term(s) ex=("digital camera"or digicam) <input type="text" value="hillary clinton"/>	Display Label ex=Digital Camera <input type="text" value=""/>
--	--

Figure 1.4: BlogPulse

query. Figure 1.4 is the screen shot of the trend search result of query “Hillary Clinton”.

The usages of bursty features on document sequences are not limited to only trend or outlier detections. There are other works that discuss a few methods for burst detection and their usages in other text mining tasks. In [62], Lappas et al. introduced the framework of burst detection in document sequence, and redefined a term “burstiness” using numerical discrepancy. They also presented a search framework which took the term in the indexing and ranking process. He et al. used Kleinberg’s algorithm to improve the topic clustering on document sequences [42]. In this thesis, we study the burst detection for pattern extractions. Our contributions to temporal pattern extraction in this thesis are the following:

1. Two complimentary frameworks to extract temporally correlated patterns using bursty information.

2. A case study of data analysis using various burstiness analysis methods.
3. An alternative definition of “burstiness” of patterns which takes both local and global information into account.
4. A bursty distance measurement for document clustering that uses both burstiness score and bursty intervals.

1.2 Thesis Outline

In Chapter 2, we give a brief survey of text mining discussing in detail the preprocessing and pattern mining step, and sequential pattern search in text mining. Chapter 2 also provides a comprehensive introduction to the concepts of burstiness. Chapter 3 presents the recursive data mining algorithm for sequential pattern mining, and proposes the language model based on those sequential patterns. We illustrate the uses of RDM on text categorization task, specifically “role identification”. We also illustrate the use of text preprocessing methods such as word clustering to enhance the performance of classifier based on sequential patterns found by RDM. Chapter 4 provides an alternative definition of burstiness of sequential patterns and the definition of temporal correlation of sequential patterns. A framework for event detection using the temporally correlated sequential patterns is also presented. Chapter 5 discusses future works. Specifically, we describe surface text pattern mining framework for information extraction, and the web application for information-aware search. We also give the conclusion in Chapter 5.

CHAPTER 2

Background and Related Works

For this chapter, we give an overview of Text mining and its applications, focusing on the text mining concepts used throughout this thesis. We also discuss the sequential pattern mining and the concepts of bursty in text documents as well as their related works.

2.1 Text Mining and Its Applications

Text Mining is the process of extracting hidden information from text documents. There was a lot of discussion regarding what aspect of data mining we should consider as text mining. For example, in contrast to seeing Text categorization as part of Text mining due to their analogies, some believed that text categorization should not be part of text mining. In [43], Hearst argued that since the author of the document knew which class the document belonged to, text categorization by itself did not provide new information. Hence, it should not be part of text mining. However, in the same article, Hearst stated that text categorization was a great text mining tool to assist finding hidden patterns in text documents (see also [23]). In this section, rather than adding another debate of what should be considered as text mining, we provide a discussion for each framework of text mining process: pre-processing, pattern mining and results evaluation (see [9, 48, 115] for an in depth survey of text mining).

Portions of this chapter previously appeared as: V. Chaoji, A. Hoonlor, and B.K. Szymanski, "Recursive data mining for role identification in electronic communications," *International Journal of Hybrid Information Systems*, vol. 7, pp. 89–100, Apr. 2010.

A. Hoonlor, B. K. Szymanski, M. J. Zaki, and V. Chaoji, "Document Clustering with Bursty Information," *Computing and Informatics*, Bratislava: Slovak University Press, (In Press).

Portions of this chapter previously appeared as: A. Hoonlor, and X. Zhu. (2006, Apr.). Unsupervised learning for surface text recovery in QA system, Computer Science Department, University of Wisconsin - Madison, WI. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.4787>, (Date Last Accessed on Jun. 2, 2011).

2.1.1 Pre-processing

In pre-processing, we convert a document into a feature vector. Similar to the issue of considering text categorization as part of text mining, the communities have different views on how the pre-processing step should be defined [9, 48, 97, 115]. For this thesis, we break down the pre-processing into two parts: text pre-processing and document indexing. A text document often contains words that can lead to lower performance in a learning model. Words that lead to lower performance in a learning model are often referred to as “noisy” words [15]. Misspelled words, abbreviation words, and common words – such as “is”, “or”, and “a” – are often considered noise words. Such a noise word does not contain information that we can use to help in classification. We must handle these words depending on an application and a learning algorithm. This process is referred to as text preprocessing. After we perform the text preprocessing on the electronic document, we can proceed to the document indexing procedure of a document and turn it into a feature vector format required by the learning algorithm.

2.1.1.1 Text Pre-processing

Three common text preprocessing methods that can handle the issue of noise word and help improve the overall performances of the learning algorithms for various text mining tasks such as categorization, clustering and information extraction are: stop words, stemming process, and pruning.

1. **Stop words:** A stop word, also known as stopword, is a word that is filtered during the preprocessing of text. For example, words such as “is”, “are”, “you” and “me” can appear in any text document. The longer the document is the greater chance of encountering them. These words affect the model performance on the text mining task such as text categorization and text clustering. For the stop words removing process, the user must provide a “stop list” – a list of words that are eliminated from the representation of documents. Usually, the words on the stop list are the nondiscrimination, closed-class words [54]. The closed-class term refers to a word class that no new word can normally be added. Determiners, conjunctions, pronouns and

adpositions are examples of closed-class terms. These words often carry little to no semantic weight. Thus, they are unlikely to help with text mining tasks. Eliminating the stop words saves a sizable space in the inverted index files used to map from terms to the documents that containing them. Due to this reason, the concept of stop words is very useful in text categorization task as shown in [76].

The disadvantage of the stop list is that it removes every phrase that contains words on the stop list. For example, one of the stop lists used for experiments in [16] contains “ask, i, to, work, you”. If we apply this stop list to the phrase “I ask you not to work”, all we have left is the word “not”. It must be noted that the list is very dependent on the dataset and the task. For example, in sentiment classification, if the word “not” is included in the stop list, after this stop list is applied to “Good will hunting is a great movie” and “Good will hunting is not a great movie”, the semantic meanings of these phrases are the same. However, if the task is to categorize documents into “movie” and “food” categories, removing the word “not” from one phrase, or not, is irrelevant.

2. **Stemming:** The terms “process”, “processing” and “processed” derive from the root word “process”. The question is, in a document, should we consider these words separately, or should we collapse them into a single root form. The stemming process addresses this exact issue. For example, if we apply the stemming algorithm on “A data processing program processes a document”, we have “A data process program process a document”. The stemming algorithm refers to the above process – that reduces the inflected and/or derived words to their stems. In practice, the words with the same stem root share a single summed frequency count. For stemming, the user does not have to provide the list of stems. The stemming process can be done automatically by using various methods. One of the popular stemming algorithms is the Porter stemmer. The Porter stemmer uses a series of simple cascaded rewrite rules to find the stem of each word [54].

The major advantage of this process is that if a document contains words or phrases that are the morphological variants of the terms, the learning model can still classify the document into the correct category. However, the disadvantage of this process is that it throws away useful distinctions. For example, a document that contains a word “stocks” is probably in a different class of a document that contains a “stockings”. According to the Porter stemmer, the word “stocks” and “stockings” have the same stem. Thus, both documents appear to be in the same class because of the stemming. Due to this reason, stemming is not always useful in text categorization. However, it is often a practice that stemming is applied to the dataset during the validation steps to see if it helps improve the classifiers [97].

3. **Pruning:** Pruning, in Machine Learning, refers to an action of removing non-relevant features from the feature space. In text mining, pruning is a useful preprocessing concept because most words in the text corpus are low-frequency words. According to the Zipf’s law, given some corpus of natural language texts, if we rank the words according to their frequencies, the distribution of word frequencies is an inverse power law with the exponent of roughly one [112]. This implies that, in any training corpus, the majorities of the words in the corpus appear only a few times. A word that appears only a few times is usually statistically insignificant – low document frequency, low information gain, etc. Moreover, the probability of seeing word, that occurs only once or twice in the training data, in the future document is very low.

In categorization, the pruning often yields the smaller size of the feature space, a smaller classification model and a better performance on testing dataset, because of the irrelevant of low frequency words to the text categorization task [51, 70]. The minimum frequency of a word to be included in the vocabulary is varied in each training corpus, and defined by the domain expert. In the next chapter, we find the local optimum minimum frequency using the tuning technique. In text clustering, words that appear in a few documents have very low mutual information and are not useful features for clustering task. In practice, to reduce the dimension of feature space, only top keywords

are selected to represent the feature space in text clustering, and documents which do not contain these keywords are often ignored [41]. However, in an application such as document retrieval where each word occurrence is important, pruning is omitted.

As discussed, these text preprocessing methods are very dependent on the datasets and the natures of text mining tasks. To show the effects of stopwords and stemming on a text mining task, we performed a comparative study on the following binary text categorization task on movie review dataset [82] and SRAA dataset [73]. For this purpose, we performed stemming, and removed stop words, on unigram and bi-gram feature types without using any feature selection. We used the standard stop list to remove the stopwords from the documents [104]. For stemming, we used Porter Stemming algorithm in Snowball project [86]. In Table 2.1, the results on the movie review and SRAA datasets show that stopwords and stemming did not always improve the performance of classifiers based on the F-score. The stemming process improved the performances of NB and ME classifiers on the both datasets, but the F-score only increase by at most 0.02. A stop list did not improve the performance of NB and ME classifiers. The stopwords and stemming also have a small impact on bigram features. For example, on the dataset, where a stopword helps improving the F-score of a classifiers the improvement is only by 0.015 at most.

The stopwords and stemming have various impacts depending on datasets, choices of feature values and the machine learning algorithms. For example, in Table 2.1, if we used the word features with binary attribute values, the stemming algorithm improved the results of NB classifiers on the movie database. However, under the same settings, it reduced the average F-score of the ME classifiers. In practice, one needs to use the tuning dataset to check if the stemming and/or the stopwords will improve the performance of the classifiers or not. The major problem with the stemming and stopwords are that they are language dependent. For instance, the stemmer and stopwords for English language cannot be applied on Spanish document. For stemmer, there are efforts such as snowball projects and Hummingbird lexical stemming to create multilingual stemmers [105]. As for stopwords, we can detect them using mutual information. Mutual information measures

Table 2.1: Results on using stopwords and stemming on Text categorization base on F-score. We compare across two types of features: words and bigrams. For each feature, we use occurrence (binary) and number of occurrences (frequency) as feature values.

Dataset	Preprocessing alg.	Word feature			
		Binary		Frequency	
		NB	ME	NB	ME
Movie	Stemming	0.814	0.838	0.805	0.837
	Stopword	0.812	0.854	0.798	0.831
	Stopword & Stemming	0.814	0.838	0.801	0.837
	None	0.812	0.854	0.798	0.831
SRAA	Stemming	0.88	0.832	0.917	0.882
	Stopword	0.921	0.899	0.92	0.884
	Stopword & Stemming	0.922	0.89	0.917	0.882
	None	0.921	0.899	0.904	0.9
Dataset	Preprocessing alg.	Bigram feature			
		Binary		Frequency	
		NB	ME	NB	ME
Movie	Stemming	0.839	0.821	0.83	0.811
	Stopword	0.841	0.816	0.831	0.809
	Stopword & Stemming	0.839	0.821	0.831	0.811
	None	0.840	0.816	0.831	0.809
SRAA	Stemming	0.88	0.832	0.917	0.882
	Stopword	0.868	0.823	0.866	0.823
	Stopword & Stemming	0.88	0.832	0.879	0.831
	None	0.868	0.823	0.92	0.884

the dependencies between two features [74].

We can use mutual information to identify words that occurs independently of categories, i.e. if words have the mutual information value of 0, then they are occurring independently of any category [74]. For an empirical proof, we conducted a small study on word clustering and mutual information on the movie review dataset for text categorization task. We found that stop words, such as “is”, “and”, and “but” had the mutual information value of 0. Thus, if the computational resources are not limited, we suggest that instead of using a fixed stop list, the mutual of all words should be used as indicators. Note that, mutual information can also be used for feature selection in text clustering task (see [41]).

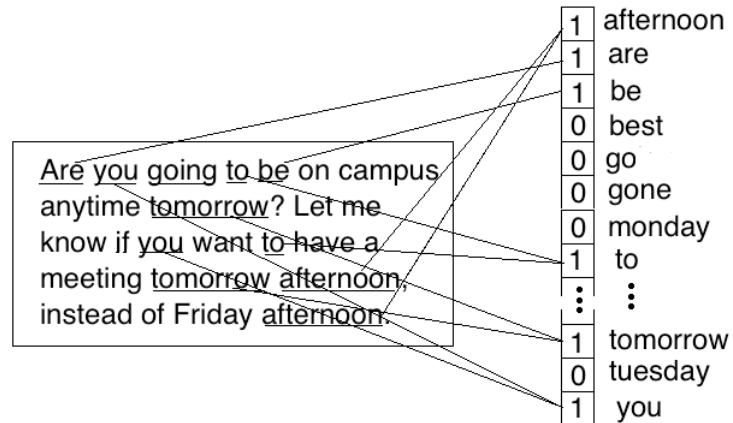


Figure 2.1: Example of the bag-of-words text representation with the occurrence of word as feature value.

2.1.1.2 Document Indexing

A document, in its raw format, is not suitable for many analysis tools such as WEKA, Matlab, etc. To the best of our knowledge, there is no current learning algorithm that can process raw text document directly. Even in document retrieval applications, text documents are indexed for fast access. Hence, we need to represent text document in an acceptable input format for the current database, learning algorithms and/or target applications. The current learning algorithms and text mining applications accept the data in feature vector format [97]. In order to convert a text document into a feature vector format, we have to answer two questions: what the set of features of documents is, and what all possible values of each feature should be. For example, if we ignore all the sentence structures and the order of the words that appear in the document, one simple feature set of a document is the set of all words in that document. The common feature value for such word feature is the occurrence of the word in a document. This style of text representation is referred to as bag-of-words text representation, as shown in Figure 2.1. Bag-of-words has a curse of dimensionality problem when the training corpus contains a large number of distinct words. Also, many of the online documents, such as Blogs, email, and Twitter, can be informal writing. Due to this informality, the documents

containing misspelled words, abbreviated words, and incomplete sentences, can add a large number of features if they are not handled prior to the conversion from a text document to the corresponding feature vector.

According to [52], most text mining applications use four categories of the features in text representation: sub-word level, word level, multi-word level, and semantic level. In Chapters 3 and 4, we used word level and multi-word level representation for text mining tasks. In word level, we can use each distinct word as a feature to represent the documents. The distinct word is considered a keyword of the document collection if it satisfies a user defined definition. Depending on various applications the definition of keyword also varies. In many text mining studies [51, 94], the common assumption made was that a word with sufficient frequency count, also called “minimum support”, in the documents of a collection was a keyword. By setting the threshold of sufficient frequency for a keyword to a value such as five, we can filter out the low frequency words such as misspelled words. In [82], the authors used the domain knowledge to create a set of distinct adjectives and considered them as keywords. In [41], the words whose mutual scores ranked in the top 100 were used as keywords for clustering task.

The word level feature is the most common features in text mining tasks. However, it loses all other information regarding the relation and position of the words. Consider the senses of the word “bat” in these two examples: “Bruce Wayne fears a bat” and “Babe Ruth is at bat for the Yankees”. For an English speaker, it is obvious that the word “bat” refers to an animal in the first sentence, while it refers to the action of striking a baseball in the second sentence. The computational linguistics refers to the problem – that a word has a number of distinct senses depending on the given sentences – as the word sense disambiguation (WSD). In text categorization, we encounter WSD when we use words as features. For example, if we consider the task of categorizing documents into either an animal newsgroup or a sport newsgroup, the word “bat” would not be a relevant feature for classification model. On the other hand, the occurrence of “bat animal” and “bat baseball” would be great features for this classification model. The latter refers to multi-word level text representation. In text categorization, empirical results show that multi-

word level document indexing improves the performances of text classifiers [13, 49]. In document retrieval, the search query is often expanded to phrase with similar meaning, e.g. a query “cooking school” can be expanded to “cooking class”, and “culinary school”. In the NLP and IR communities, one well-known multi-word level language model is an N-gram model. The N-gram model uses the sequence of $1, 2, \dots, N-1$ words to predict the N^{th} word. The N-gram model has two important behaviors: (i) the N-gram model relies on its training corpus, and (ii) the accuracy of N-gram model increases as the value of N increases [54].

Although, sub-word level and semantic level have limited uses in text mining, their representations are very interesting. One common sub-word level representation is the syllable feature vector. How we pronounce each syllable of a word can give a different meaning to a spoken text. Thus, if we want to capture the meaning of the spoken text, we should consider using the order of syllables as text representation. This text representation according to the order of syllables is used to represent the spoken text in a speech recognition application such as the phone-based model by King et al. [59]. For the semantic level of text representation, one example is the cause-level dependency. For this type of feature, a sentence is parsed into cause-level structure, and assigned syntactic-valence values to clause constituent. Specifically, we assign subject, direct object, oblique and adjunct in the sentence. A well-known example for why this style of text presentation is necessary in such task – given by the famed linguistic, Noam Chomsky in 1957 – is “the sentence colorless green ideas sleep furiously” whose grammar is correct, but whose “meaning is nonsensical” [4]. Other level abstraction of text representation would not be able to capture the nonsense from this sentence.

For feature values, due to their simplicity and showing good empirical results in many datasets, three attribute values commonly used in text classification task are number of occurrence (frequency), occurrence of term, and term-weighted value such as term frequency-inverse document frequency [51, 82]. The example for the number of occurrence is given in Figure 2.2. As for the occurrence of term, there are two possible feature values, zero and one. The feature value of one implies that the feature appears in the document. The feature value of zero says otherwise. The text

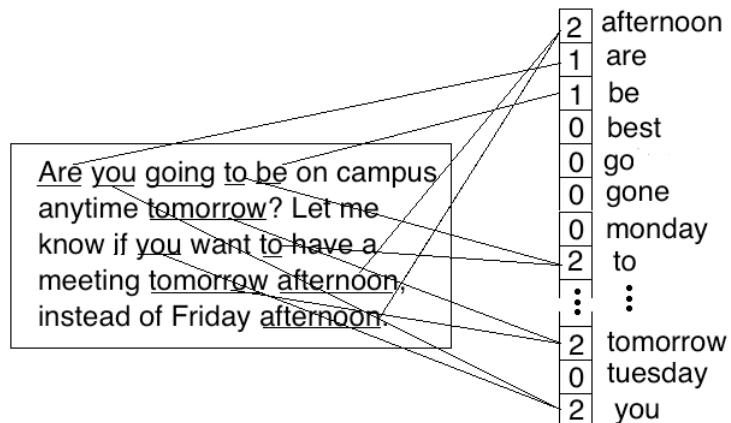


Figure 2.2: Example of the bag-of-words text representation with the number of occurrence as feature value.

representation of the document in left hand side of Figure 2.1 using the occurrence of term as feature value is shown in right hand side of Figure 2.1.

Term weight refers to a term weighting approach where a term should have higher weight if that term is a relevant term in documents of interest. There are many variations of the term weighting schemes. The most popular one is known as term frequency-inverse document frequency (TF-IDF) weighting scheme. Term weights, such as TF-IDF, were used for document indexing to improve their performances in many text mining tasks [97, 51, 50]. There are two components in TF-IDF, $TFIDF(t, d)$, – the term weight $TF(t, d)$ and the inverse document frequency $IDF(t)$ [94]. $TF(t, d)$ is the number of times a term t occurs in a document d . $TF(t, d)$ captures the idea that if a term t is a keyword in document d , then the term weight of t in d should be high. $IDF(t)$ is defined as

$$IDF(t) = \log\left(\frac{|D|}{DF(t)}\right) \quad (2.1)$$

where $|D|$ is the number of documents in the training corpus, and $DF(t)$ is the number of documents that contain t . On the other hand, a high value of $IDF(t)$ indicates that t appears in only a few documents, while a low value of $IDF(t)$

indicates that t appears in many documents. $TFIDF(t, d)$, the TF-IDF value of a term t and a document d , is defined in Equation 2.2.

$$TFIDF(t, d) = TF(t, d) \cdot IDF(t) \quad (2.2)$$

Hence, a high weight of $TFIDF(t, d)$ can be caused by either a high frequency term t in d , a small number of documents that contains t relatively to the whole corpus, or both. This weighting scheme can be used as a threshold to effectively filter out common words or phrases that appear throughout the corpus. Finally, a document d can be represented as a vector $\vec{d} = (TFIDF_{t_1}, TFIDF_{t_2}, \dots, TFIDF_{t_{|V|}})$ where $TFIDF_{t_i}$ is the $TFIDF(t_i, d)$ value of the term t_i in document d .

There are numerous works that deal with extracting patterns and features from unstructured raw text data. One hot research topic is an author identification problem. Experts in the fields of information retrieval, natural language processing, data mining and statistical learning have focused on a diverse set of techniques for feature extractions and feature selections to solve the author identification problem. Linguists use statistical techniques to obtain a set of significant words that would help identify authors. In 1964 Mosteller and Wallace [77] solved the Federalist Papers problem by identifying 70 “function words” and applying statistical inference for the analysis. In the Natural Language Processing (NLP) community, the Hidden Markov Model greatly influenced many techniques used for speech recognition such as a stochastic part-of-speech tagging program and a noun phrase parser proposed in [19]. In [40], the authors improved the performance of part-of-speech tagging by assigning a syntactic category to each word in a text document using Machine learning techniques. For feature selection, Guyon and Elisseeff provided a comprehensive introduction to techniques in [39]. Besides text mining, many other applications – such as automatic painting classification [26], detection of malicious alterations of images [91], grouping proteins and genes into classes based on function, location and biological process in bioinformatics [121] and characterizing the behavior of an individual or a group of individuals in e-Commerce [119] – also benefit from feature extraction and feature selection.

2.1.2 Pattern Mining and Results Evaluation

After a text document is converted to a feature vector format, the pattern must be identified and analyzed to help extracting hidden information. There are various pattern mining algorithms available depending on the nature of text mining task. In Chapters 3 and 4, we explore three distinct groups of algorithms in pattern mining: categorization, clustering and information retrieval. Since each group of algorithms requires different evaluation methodologies, we also discuss the appropriate evaluation methodologies of each group.

2.1.2.1 Categorization

Text categorization is the task of assigning the predefined label to the unlabeled text document. In general, a document is categorized to one class. However, as a document can contain multiple subjects, there are many variations in text categorization. For example, if there are two classes of documents: “politics” and “movie”. Given a news article about a documentary movie about politics, then it is not cleared how we should label such document. This type of categorization problem is considered classification with multiple labels. One of the common classification model used to solve this multi-label text classification is a mixture model based classifier. Such model can be trained using expectation-maximization (EM) methods [71]. Recently Chang et al. introduce a new weighted indexing technique for mixtures of linear classifiers that can handle multi-label text classification tasks [14]. Blei et al. used the latent Dirichlet allocation (LDA) as the generative topic model to perform text categorization tasks [10]. There are many other interesting applications of the text categorization, some of which are listed below.

1. **Sentiment Classification:** Sentiment classification is a special type of non-topic-based text categorization, where the predefined classes are the overall sentiments of the documents. For example, the positive and negative sentiments are used as class labels for movie reviews dataset in [82]. Sentiment classification has been used for customer review analysis, and summarization of opinions on webpage, such as newsgroup, forum and blog [66].
2. **Authorship identification:** Problems of detecting authorship of text, such

as identifying the authorship, detecting plagiarism and authorship assessment, have become more demanding in recent years. In [107], the authors suggested and introduced various text categorization methodologies for these tasks. In [15], text categorization was used to identify the social role of the authors.

3. **Web classification:** One possible way to classify the web is to use the text content in the webpage. In [70], text categorization methods were used for web classification. In [23], after performing classification on documents, sets of ontology were extracted from the classes of website to use as knowledge base for other information extraction tasks.

Categorization methods used in machine learning and data mining communities can be used for text categorization. Easily implemented methods such as Naïve Bayes, and K-nearest neighbors have been applied in text categorization task with high performance on various corpora [70, 117, 119, 118]. In [51], support vector machine (SVM) was applied to text categorization task. SVM showed that they work well when almost all features were relevance. In [82], NB, SVM and ME showed that they performed equally well in the setting of sentimental classification. Sebastiani provided a survey of machine learning methodologies used for text categorization [97]. The feature vectors used in these work are often in the form of bag-of-words. The usages of more complex classification language models such as N-gram, and hierarchical text categorization methods were proposed in [80, 84]. One limitation of models such as N-gram, shown in [51], was the curse of dimensions because text corpus contains a large number of words that had high mutual information score. Thus, features which compose of multiple words were often limited by the size of the features. Note that early work in text categorization had been done under supervised learning settings. The test error rate, precision and recall values, and the receiver operating characteristic (ROC curve) were, and still are, the common means of performance evaluation. Such scores were collected using hidden test dataset, and cross validation tests [74]. For the comparison between different classification models, paired t test was often used to show statistically significant different between pair of models [74].

2.1.2.2 Clustering

Text Clustering is the task of assigning a text document to a group such that the documents in the same group are similar while two documents from distinct group are different. Most of the methods used for text clustering are clustering methods used in machine learning. Three main clustering methods often used for text clustering are k-means clustering, fuzzy c-means clustering, quality threshold clustering [74, 54]. K-means clustering is a well known clustering methods where predefined number of clusters is given, and that each object can belong to only one cluster. Fuzzy c-means are K-means clustering that allows an object to be assigned to multiple clusters. While Fuzzy c-means clustering is interesting methods to handle text clustering, the multi-class clustering is well beyond the scope of this thesis. We encourage the readers to look at [74] for references in fuzzy c-means. Quality threshold clustering is different from k-means and c-means clustering in that they do not cluster objects into a predefined number of clusters. Quality threshold clustering clusters objects into the same group if they are similar to each other more than a provided threshold. The resulting number of clusters using quality threshold clustering was much larger to than those of k-means clustering based methods. Recently, graph-theoretic clustering method had received a lot of interests. In [41], an approximated method of clustering with lower bound on similarity was used to cluster text documents.

Similar to text categorization, the common features used in text clustering are again words. However, not only that a word can be ambiguous in many languages, but it can also be relevant to text clustering task as they do in text categorization task. Also, the size of word feature space is large, if all words are used as features. The resulting clusters of the documents often form a sphere like structure. These problems often lead to difficulties in clustering. We conducted a small experiment by clustering 2775 document in 3 of the 20newsgroup dataset. We used all words as features and performed the k-mean clustering and the density based clustering (a form of quality threshold clustering). The results, shown in Table 2.2, indicate that k-mean clustering assigned almost all documents into one class, showing a problem of clustering data with curse of dimension problem. For density based

clustering, it is natural to have the number of assigned clusters more than that of the actual clusters, depending on the predefined density threshold. In Table 2.3, show the clustering results based on the density threshold that we set in order to have at least 3 large clusters. While it clustered the documents from different classes into different clusters better than k-mean clustering, density based clustering still clustered documents from multiple categories together in the same clusters. This illustrated that the same set of English words can be used in many topics. Hence, word feature must be selected carefully. Otherwise, two documents with the same word maybe clustered together when they should not.

Table 2.2: Document clustering using K-mean clustering

True Cluster Assignment	Clustered Assignment		
	First set	Second set	Third Set
Cluster 1	1	798	0
Cluster 2	15	965	5
Cluster 3	3	980	8

Table 2.3: Document clustering using density based clustering

True Cluster Assignment	Clustered Assignment					
	1st	2nd	3rd	4th	5th	Noisy Points
Cluster 1	33	745	0	0	0	21
Cluster 2	145	605	174	0	43	18
Cluster 3	199	0	662	115	0	15

In [41], a set of words with high mutual information was used as feature. Text clustering has been used widely in many areas. In the study of focus group interviews, Dransfield et al. used text clustering to analyze interviews regarding the quality of pork meat, collected from the selected group of interviewed subjects [29]. The evaluation process of the results of clustering is subjected to user judgments. To evaluate the cluster, one must set an evaluation benchmark, or “gold standard” [69]. Then, we can use tests such as Purity test to statistically measure how much the clustered results varies from the gold standard. Purity test measures collectively how many documents in each clusters is considered of different type of document than that of the majority (see [69]). Other tests include F-measure, Normalized mutual information and Rand Index.

2.1.2.3 Information Retrieval

Information Retrieval is a research field which overlaps with many other fields, such as natural language processing, information science, cognitive science, text mining, etc. The general definition of information retrieval is the science of retrieving information from given dataset, which can be in form of online documents, local file systems or relational databases. Signal gave a brief overview of information retrieval in [100]. In [69], Manning et al. gave an introduction of information retrieval, from traditional Boolean retrieval, document indexing and its compression, to usages of language model and evaluation in information retrieval. The overlap between information retrieval and text mining is referred to as text retrieval. Text retrieval process has a distinction from the general pattern mining process that is it requires a user query. Upon receiving a user query, Text retrieval system will first retrieving relevant documents. Then, the system will evaluate the retrieved documents and either extracting necessary information, or sort the documents according to their relevant to the query.

Traditional text retrieval method is considered Boolean retrieval method. The method begins by indexing the document using bag-of-word method to create an indexed matrix \mathcal{D} . \mathcal{D} of $n \times d$ size where n is the number of documents and d is the number of indexed words in the dataset. $\mathcal{D}(i, j) = 1$ implies that document i^{th} contains the indexed word j^{th} . $\mathcal{D}(i, j) = 0$ states otherwise. The search operation for each query becomes a simple logic operation between column of 0 and 1. Table 2.4 shows the Boolean indexed matrix of the report on movie casts. The “M. Caine AND K. Watanabe AND NOT(E. Page)” query for a term-document incidence matrix in Table 2.4 has the Boolean query representation of “1110 AND 1100 AND 0111”. Hence, the search result in Table 2.4 is 0100.

For text mining, result evaluations vary from one task to another. For example, in the document retrieval task, we can count how many documents among the top-5 and top-10 rankings are related to the given query [62]. In question and answering systems, predefined question and answer pairs were tested on the system to see how many pairs were answered correctly [89]. Lin and Hovy proposed an automated model to evaluate the summarization system [67]. An automate N-gram based

Table 2.4: A term-document incidence matrix, where matrix element in row i and column j is 1 if an actor, or actress, in row i appears in the movie in column j .

	Inception	Batman Begins	The Prestige	Memento
G. Pearce	0	0	0	1
M. Caine	1	1	1	0
K. Watanabe	1	1	0	0
C. Bale	0	1	1	0
E. Page	1	0	0	0

system to measure the quality of machine translation was suggested by Doddington in [27]. Faloutsos and Oard [32], Raghavan [88] and Crestani et al. [24] provided the overviews and the brief discussions of the other evaluation methods for text mining.

Text retrieval applications range from the text retrieval systems [81, 96], extracting information for question answering system [89], to the recent introduction of trends tracking in Blog community [35]. Text retrieval has been a highly active research area especially after the creation of the Internet. One conference devoted for text retrieval is the Text Retrieval conference (TREC) [106]. Since 1992, TREC has provided various dataset, evaluation techniques and test collections, and workshops series to increase collaboration in text retrieval researches. In recent years, there were numerous interests of text retrieval in electronic media in research labs as well as those in the industrial. In the industrial world, there are many information retrieval commercial applications. One of which is the online search engine. From a simple idea of providing an online search engine using simple indexing method, to PageRank algorithm, nowadays, search engine such as Google and Yahoo have become multi-billion dollars businesses [44]. Besides the search engines, applications such as Google trends [38], Yahoo Buzz [116], BlogPluse [35], etc. are emerging text retrieval technologies that help users to search for documents whose relevant to a query are not only the matches between documents and the query, but also the time in which the documents are published and how many other people appear to be interested in the same set of documents. Selected current areas of focus, as shown in list of new TREC Tracks, are the followings [106].

1. Blog Track: Blog track focuses on analyzing the behavior of the bloggers

especially in how they find their interested information on Blogosphere.

2. Web Track: With the fast expansion rate of Internet, web track aims to improve efficiency and quality in web-specific retrieval tasks from large data collections.
3. Entity Track: This track is interested in the entity-related search on online documents, such as the study on the required information for finding a given entity on the URLs.

2.1.3 Text Mining Applications

So far, we have discussed various existing text mining applications in online media, sentiment analysis, and information retrieval in operating system. There are numerous applications which use text mining as their foundation. Here, we present a selected few of text mining applications which have gained a lot of interests recently.

1. Web: Besides the ongoing research on online search engine, Blog and news media had attracted a lot of online text mining research projects. Some of the applications are BlogPulse and Trend Tool. Both BlogPulse and Trend Tool detect the current top news and/or topic of discussions, also referred to as buzz or trend, in the blog. Other interesting web application is the automatic text summarization. In [95], knowledge of internal text structure and extracted semantic hypertext links, are used to create the summary. In [75], the text summarization is performed with aid from the paragraph extraction methods.
2. Computer Security: Data mining were applied for computer security purposes such as intrusion detection, anomaly detection and credit card fraud [25, 53, 103]. Text mining is simply a subset of such work. For example, in author identification forensic work by Vel et al., text mining was applied to email content to discriminate between authors [107]. José et al. used text categorization to detect the intrusion in form of misuses of web applications [53]. Although the work was performed offline, the results showed that they could detect intrusion with over 90% success rate [53].

3. Biomedical: Biomedical is the research area where there are numerous publications. The number of published articles in a given topic that come out each month can be so large that one cannot hope to engage all of them. In [21], Cohen and Hersh gave a survey of the current works to find efficient and effective ways to retrieve the information from large digital biomedical text collections. Some well-known applications in Biomedical are related to extracting information from PubMed (see [87]) such as GoPubMed – a document retrieval tools from almost 19 million analyzed documents using gene ontology and subject heading [28].

2.2 Sequential Pattern Mining

A common problem in data mining is to mine patterns from the dataset. One type of the dataset can be considered as a sequential dataset – a set of ordered list of items. Agrawl and Srikant, also Mannila et al., introduced sequential pattern mining problem as a data mining task that extracted informative patterns, such as association rules, from such sequential data sets [2, 68]. In the other words, sequential pattern mining is a data mining technique that extracts the significant pattern in the sequential data. A sequence pattern refers to a set of features that occur in the training dataset, and convey some hidden information. The existing algorithms and applications automatically decide significant scores based on the provided data such as search query, or presetting scoring function such as “frequency” [69]. One of the sequential pattern mining methods is called Apriori algorithm.

Apriori algorithm extracts an association rule from a dataset by first finding an itemset which satisfies the user specified frequency threshold, also called minimum frequency or simply “minsup” [3]. If the items appear together more than the minsup, the set of these items is called a frequent itemset. Then, the algorithm generates all possible the association rules from the frequent itemset as follow. For example, if $S = \{s_1, s_2, \dots, s_k\}$ is a frequent item set, then the Apriori algorithm deducts all possible two subset combinations $\{s_2, s_3, \dots, s_k\}$ implies s_1 , $\{s_1, s_3, \dots, s_k\}$ implies s_2 , etc. The outline of Apriori algorithm from [3] is shown in algorithm 1. Note that, if there is a hierarchical structure in a set of item, a hidden associate rule can also

Algorithm 1 Apiori Algorithm: Below is the overview of the Apiori algorithm

Input: Database \mathcal{D} and minimum frequency \mathcal{MINSUP}

Output: Frequent sets \mathcal{F}

```

1:  $\mathcal{F}_1 =$  item set of size 1 that occurs in  $\mathcal{D}$  at least  $\mathcal{MINSUP}$  times
2:  $i = 2$ 
3: while  $\mathcal{F}_{i-1} \neq \emptyset$ 
4:  $\mathcal{C}_i =$  All candidate itemsets generated by joining  $\mathcal{F}_{i-1}$  with itself
5: for candidate  $c \in \mathcal{C}_i$ 
6: if  $c$  occurs at least  $\mathcal{MINSUP}$  times in  $\mathcal{D}$  then
7:  $\mathcal{F}_i = \mathcal{F}_i \cup c$ 
8:  $\mathcal{F} = \mathcal{F} \cup \mathcal{F}_i$ 
9:  $i++$ 
10: end while

```

be extracted. For example, if the current data is the records of checked out items at a super market, such as {Pepsi, Lay potato chips}, {Coke, Lay potato chips}. If we consider, Pepsi and Coke as types of soda, then one possible pattern is the hidden association rule of checked out items of {soda} implies {Lay potato chips}.

Text document can also be viewed as a sequence of items. If we consider a document of length n , $d = w_1, w_2, \dots, w_n$, where w_i is the word at position i^{th} of d , then d is simple an ordered list of words – a sequence. The significance of a sequential pattern depends on users and applications. The concept of sequential pattern in text mining has been around for sometimes. One of the well-known language models using sequential pattern is N-gram model. As mentioned earlier in this chapter, N-gram refers to the ordered words of length N . The N-gram model uses the previous $N - 1$ words to predict the next word. According to [54], the N-gram model has two important behaviors: the N-gram model relies on its training corpus, and the accuracy of N-gram model increases as the value of N increases. Thus, the n-gram model is simply a fixed length pattern, whose significant depends on the underlying probabilistic model corresponding to each training corpus. N-gram is a simple yet powerful language model as shown in topic discovery in text retrieval related task [109]. In [83, 27], N-gram was introduced as a language model and used to assist in an automatic evaluation model using BLEU/NIST scoring process.

In [89], a sequential pattern composed of mixed of words and abstracted notion

are used to extract information from the documents. The pattern, referred to as surface text pattern, can be a particular order of word such as $\langle X \rangle$ “was born on” $\langle D1 \rangle$, and $\langle X \rangle$ “died on” $\langle D2 \rangle$ where $\{\langle X \rangle, \langle D1 \rangle\}$ and $\{\langle X \rangle, \langle D2 \rangle\}$ are the hidden birth and death dates information we would like to extract from the document. The significant of a surface text pattern in question and answering system are defined by the similarity of the input query to that of the existing pattern. For example, if we want to search for the birth date of Albert Einstein, the pattern $\{\langle X \rangle, \langle D1 \rangle\}$ from $\langle X \rangle$ “was born on” $\langle D1 \rangle$ is more significant to us than $\langle X \rangle$ “died on” $\langle D2 \rangle$. In [63], a sequential pattern of arbitrary length with approximated match was traced across documents in news to documents in blog posts to study the information flow between news media and bloggers.

As we shown the common key benefits of a sequential pattern based text mining approach in Chapter 1, we now show the disadvantages of the model which uses sequential pattern:

- The feature space of sequential pattern is large. For example, if a word feature space of one dataset is of size N , then a bigram feature space of the same dataset can be as large as N^2 .
- The complexity of a sequential pattern feature based model is much larger than that of a single-word feature based model.
- The computation cost of a sequential pattern feature based model is often high such that online application is not always possible.

Thus, we must be careful when use sequential pattern mining for text mining applications. If the performance of the model using sequential pattern features improves only a little over that of word features, but the computational cost is more than triple, then sequential pattern mining is not likely the answer.

2.3 Concepts of Burstiness in Text Mining and Related Works

The usages of words and phrases change over time. Certain words or phrases may be used to explain or refer to different information over time. For example, the phrases containing “president” and “obama” before and after 2008 are likely to have different meanings. The pattern “president, \perp , obama” is likely to be found in the documents created in the period after 2008 rather than in those from before 2008. In previous chapter, we considered the documents as a single set. In electronic documents, one of the meta-data of the electronic document is its time of creation. Thus, one possible view of the set of documents is in sequence of the creation time of the documents. In such text stream, bursty interval can be defined as a period when a given term appears more frequently than its global frequency [62]. In turn, the detection of bursty interval is referred to as burst detection.

In information retrieval, on-line information and text mining areas, the same ideas can be seen in the studies of temporal dynamics of text stream [60]. One aspect of such study discussed in [60] was the concept of the burst of term or query in sequential data sets. In [108], Vlachos et al. used information gained from compressed representations of periodic data to detect peaks of online search queries. The S2, a similarity tool, allows user to detect burst detection and find the best-k periods of similarity queries from query database [108]. In [110], Wang et al. proposed the coordinated mixture model to detect the “correlated bursty topic patterns” from document sequences. Their mixture model can detect correlation between two patterns from two completely different languages such as English vs. Chinese. In [62], Lappas et al. introduced a “burstiness-aware search framework” as integrate the burstiness concepts with the document ranking and indexing of documents. In which, burstiness of a term in text streams are broadly defined as the measurement of how a term’s occurrence in a given time frame varies from its expectation value. In language modeling context [98, 20], “burstiness” was used to explain the functionality of the event where a word appeared more than once in a document.

In [60], the bursty region of a given term was detected using a two-state finite

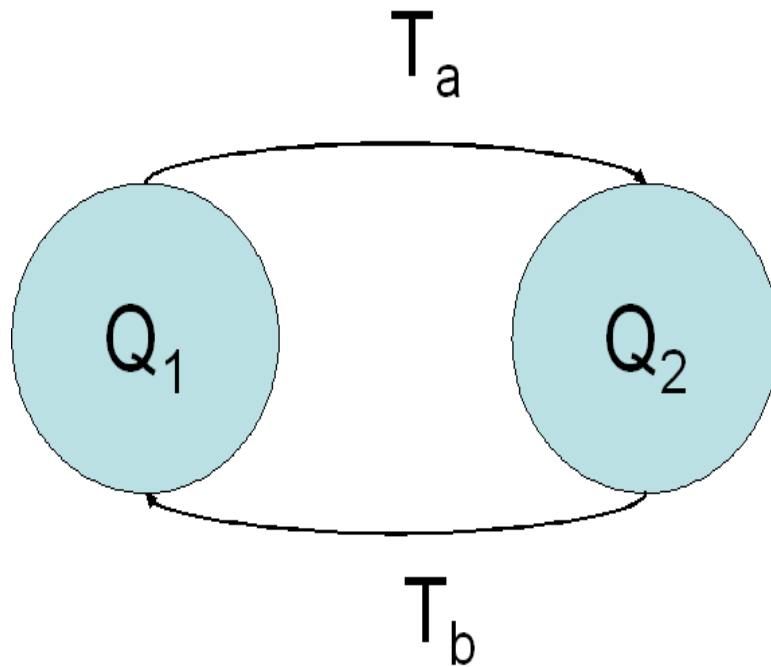


Figure 2.3: Two-state finite machine to detect the bursty period of a given term in the data collection.

machine as shown in Figure 2.3. The first state of the finite machine, Q_1 is the state of low occurrence of a given term. The other state, Q_2 is the state of high occurrence of the term. T_a and T_b are the transition functions, such that if the occurrence rises and falls to certain thresholds the state transition is made. Finally, the period of time in state Q_2 is considered bursty period.

In [62], the numerical discrepancy was used to described such value as seen in the following equation.

$$\mathcal{B}(t, [l : r]) = \left(\frac{|d_i : t \in d_i, \forall i \in [l : r]|}{|d_j : t \in d_j, \forall j|} - \frac{\text{len}([l : r])}{m} \right) \quad (2.3)$$

where t is the interested term, or word, $[l : r]$ is the time period, d_i is the document containing term t in $[l : r]$, d_j is the document containing term t in the dataset, $\text{len}([l : r])$ is the length of $[l : r]$, and m is the total time steps. We can deduce from Equation 2.3 that the burstiness' range is $[-1, 1]$. Using this range, Lappas et al. found the burstiness region of a given query using the maximization method called “max burst” [62].

The study of the bursty of a term often focuses in the global view. However, in [63], a statistical model was introduced to study the burstiness of topics and short phrases in online news and blog datasets. The model focused on the level of bursty around the peak region of a given phrase. The peak of a given phrase often had a fast rising and a slightly slower fall. The phrase also exhibited a lasting impression much longer after the peak than the time of the initial report in the news to its peak of discussion.

The concepts of burstiness have been studied in various areas. In signal processing, a burst in data is referred to as a spike in signal, where spike detection methods detect the sudden change of signals. In [79], Nenadic and Burdick proposed a novel method for spike detection using the continuous wavelet transform. In [12], the wavelet transformation based methods were used for automated spike detection of renal sympathetic nerve activity in mouse. Previous study of using waveforms for spike detection in neural activity can also be seen in [64]. In data mining, [7] provided a detailed summary of the abrupt changes detection in data. In [25], Curry et al. introduced the change detection using cubes of models that detects changes in large data sets and discussed various issues in developing a change detection system.

CHAPTER 3

Recursive Data Mining

3.1 Introduction

The general assumption of Naïve Bayes is that each feature is independent of each other. The empirical results showed that Naïve Bayes worked well in many data sets even if they did not satisfy this assumption [74]. In natural language, two words in the same paragraph are less likely to be dependent on each other than two words in the same sentence. Thus, the naïve assumption on word features is not always true. Moreover, given a certain structure of the document and a given selected sequence of words in the beginning, human can make a reasonable guess of which type of words would follow. For example, if the document start off with “to whom it may”, we can guess that it would follow by “concern”. The language model based using naïve assumption on word features is unlikely to make the correct guess. The language model based on sequential patterns such as N-gram can probably give the word “concern” as one of the most likely word to follow. The empirical results showed that N-gram language model also worked well in various problem settings [84, 58]. However, language model with limited length of patterns such as N-gram language model is likely to fail to guess that “sincerely” and “best regards” are likely to appear toward the end of the document. In this chapter, we present “Recursive Data Mining” (RDM) framework for discovering statistically significant sequence patterns from a stream of data using its hierarchical structure.

RDM extracts “significant” patterns from the sequence of inputs in three main steps: (i) it abstractly represents each phrase with a unique id, (ii) it outputs a rewritten sequence and remove any word of a sequence deemed “noisy”, (iii) it recursively works on the rewritten sequence again until no more pattern can be found. Detail descriptions of these steps are explained in section 3.3. RDM framework incorporates the ideas of the naïve assumption when we go from one level to the next as

Portions of this chapter previously appeared as: V. Chaoji, A. Hoonlor, and B.K. Szymanski, “Recursive data mining for role identification in electronic communications,” *International Journal of Hybrid Information Systems*, vol. 7, pp. 89–100, Apr. 2010.

follow. If a token occurs frequently together with another, they both will become a part of the higher level token. Hence, if there are dependencies between occurrences of tokens at one level, they would disappear at the higher level. Consequently, at the highest level of RDM processing, occurrences of tokens are independent of each other by definition will work very well at that level. Without the naïve assumption on word, classifiers based on RDM framework are expected to perform better than the Naïve Bayes classifier.

The hierarchical nature of RDM enables us to capture patterns at various levels of abstractions. Moreover, it allows us to remove noisy symbols from the stream as we move from a lower level to a higher level in the hierarchy. This ultimately leads to discovery of “long range patterns” that are separated by long noisy intermediate segments, such as “to whom it may concern ... sincerely”. RDM framework allows arbitrary size patterns and approximate (similar) patterns to be discovered. It uses statistical techniques, such as log likelihood ratio tests, information gain and mutual information methods, for identifying significant patterns. Techniques such as part of speech tagging, word clustering and synonym dictionaries can augment RDM. RDM is developed to find significant sequential patterns in a stream of text such as email, blog or chat-room session. However, RDM can be applied to any data possessing a sequential order (time series data, genome data, etc.).

Our work shares with [80] and [101] an assumption that the underlying structure in languages can be learned in a hierarchical manner. In [80], the authors extracted a hierarchical nested structure by substituting grammar for repeated occurrences of segments of tokens. Similarly, in [101], the authors presented a data independent hierarchical method for inferring significant rules present in natural languages and in gene products. Our efforts differ in that we provide certain flexibility in the patterns found by allowing gaps. This enables us to work with much smaller datasets as compared to [101]. In recent works [18] and [122], the frequent mining was modified to obtain useful patterns which are used for classification in various domains.

To illustrate RDM framework, we applied RDM on a special form of text categorization task called *role identification* – an extension to author identification

problem. Within realistic human interactions, typically the nature of the interaction is governed by the relationship between the entities involved in the interaction. For instance, the manner in which a person communicates with her superior is usually different from the way in which she communicated with her subordinate. This implies that each entity can assume multiple roles over a range of interactions. As a result, the task of identifying the role an entity plays in an interaction is more complex as compared with authorship assessment. To effectively deal with the role identification problem, we used RDM as a generic feature extraction. RDM obtained *sequential patterns* and used them as features for role identification task. We focused our task to patterns extraction from electronic mail. Specifically, we experimented on the Enron dataset which was introduced in [61] as a benchmark for email classification. The extracted patterns were subsequently used to build a classifier. The features obtained from the Enron dataset were used to identify the organizational role (e.g., manager, president, secretary, etc.) of the sender of an email. Potential applications of our approach include analysis of groups on the Internet for which the network structure is not clear or not explicitly defined and, in some cases intentionally obstructed or hidden by the group members. Identifying the leaders and followers in such informal groups is of great value for social sciences, network science and security.

The rest of the chapter is organized as follows. We establish a set of symbols used in this chapter in the next section. Section 3.3 contains a detailed description of our methodology. We discuss how RDM framework can be applied for text categorization task in Section 3.4. The experimental results are presented in Section 3.5, while Section 3.6 offers conclusions.

3.2 Preliminaries

Consider a set of sequences, denoted as \mathcal{SEQ} . Each sequence consists of a series of *tokens* from a set \mathcal{T} . Thus, a sequence $S \in \mathcal{SEQ}$ of length n can be represented as t_1, t_2, \dots, t_n , where $t_i \in \mathcal{T}$. Depending on the application, a token may represent a different entity. For instance, in the domain of text documents, a token can either represent a character or a word and a sequence S would then correspond to the whole

document. For stock market data, each token could represent a numeric value (price and volume) while the sequence would represent the entire time series of purchases (or sales) of a certain stock. A special token, called the *gap token*, corresponds to a blank entry and is represented by the symbol \perp . The gap token mimics the '.' character in regular expressions - it can be matched with any other token. A *sequence pattern* \mathcal{P} is an ordered sequence of tokens from $\mathcal{T} \cup \{\perp\}$. Formally, \mathcal{P} can be denoted as $\{s_i : s_1, s_{l(\mathcal{P})} \in \mathcal{T} \wedge s_j \in \mathcal{T} \cup \{\perp\}, j = 2 \dots l(\mathcal{P}) - 1\}$, where i is the index of a token in the sequence and $l(\mathcal{P})$ is the *length* of the pattern \mathcal{P} . It should be noted that the first and last tokens are never the gap token. This restriction is useful for combining contiguous patterns.

Two patterns are said to have an *exact match* if they consist of the same sequence of tokens. Given a *similarity function*, $sim(\mathcal{P}_1, \mathcal{P}_2)$ a similarity score between 0 and 1 is assigned to each pair of patterns. Exact matching restricts the similarity score to binary values - $sim(\mathcal{P}_1, \mathcal{P}_2) = 1$ if $\mathcal{P}_1 = \mathcal{P}_2$, 0 otherwise. The presence of a gap token in a sequence pattern relaxes the exact match constraint, allowing it to match a wider set of patterns with $sim(\mathcal{P}_1, \mathcal{P}_2) \in [0, 1]$. A match with similarity score greater than $\alpha \in (0, 1)$ is called a *valid match*. The set $\mathcal{M}_{\mathcal{P}}$ is the set of valid matches for a pattern \mathcal{P} . A pattern \mathcal{P} of length l and g gaps is termed as a (l, g) - *pattern*. If \mathcal{P} has a match at index i in sequence S , then it belongs to the set of patterns $S_i(l, g)$ -*patterns*. The set of patterns $S_i(l)$, given by the expression $\cup_{g=0}^{max_gap} S_i(l, g)$ represents all patterns of length l starting at index i in S . *max_gap*, as the name indicates, is the maximum number of gaps allowed in a pattern. In the rest of the chapter, the term pattern would always imply a sequence pattern and terms pattern and feature would be used interchangeably, unless stated otherwise.

3.3 Recursive Data Mining

Recursive Data Mining (RDM) is an approach for discovering features from sequences of tokens. Given a set of sequences as input, the algorithm processes the input sequences using some predefined methods, such as stop words, stemming and/or some user-defined conditions such as ignoring terms that start with 0, or considering portion of sequence *ATCT* and *TACT* as a gap. The processed sequence is

used as the initial sequence for the first iteration in the iterative step of RDM. In the first iteration, the algorithm captures statistically significant patterns from the initial sequences. The obtained patterns are assigned new tokens. The initial sequences are re-written by collapsing each sequence pattern to its newly assigned token, and removing the “noisy” tokens from the sequence. Next, the algorithm operates on the re-written sequences continues to iterate through the pattern generation and sequence re-writing steps until either the sequences cannot be re-written further or a predefined number of iterations is reached. Each generation of sequences in the above process is termed a *level*, with the initial set of sequences called *level(0) sequences*, and denoted as \mathcal{SEQ}_0 . The patterns obtained at each level form a set of features. The term “recursive” in the name refers to this iterative step that obtains the next level by operating on the current level. In the RDM process, we claim that the recursive (hierarchical) processing of the data captures distinctive features at varying levels of abstraction. Intuitively, at lower levels the patterns obtained are more specific, resulting in a smaller set of valid matches (\mathcal{M}). At higher levels, the patterns are more general, resulting in a larger \mathcal{M} set. On the other hand, with increasing levels, the number of patterns found decreases monotonically.

3.3.1 General Framework

In this section we present the details framework of an RDM based classifier. RDM starts with *pre-processing*, *pattern generation*, and follows by pattern selection through the *pattern significance assessment* step. Out of the significant patterns, the *dominant patterns* form the feature set for a level. The overall RDM process is outlined in Algorithm 2. The input is a set of sequences \mathcal{SEQ}_{input} . The preprocessing step applies the user’s predefined preprocessing algorithm to \mathcal{SEQ}_{input} and generates \mathcal{SEQ}_0 for the iterative procedure. The set of sequences for level $(i + l)$ is generated from the sequences in level i and the set of dominant patterns \mathcal{D} . \mathcal{P}_{ALL} and \mathcal{P}_{SIG} represent the sets of all patterns and significant patterns respectively. Dominant patterns (denoted by \mathcal{D}) for a level are obtained from the *get_domi_patterns* method. The union of dominant patterns at each level is collected in \mathcal{L} . Below, we describe each process in detail.

Algorithm 2 Recursive Data Mining

Input: Set of sequences \mathcal{SEQ}_{input}
Output: Sets of patterns (features) \mathcal{L} , one for each level

```

1:  $\mathcal{SEQ}_0 = preprocessing(\mathcal{SEQ}_{input})$ 
2:  $\mathcal{L} = \{\}, i = 0$ 
3: repeat
4:   if  $i > 0$  then
5:      $\mathcal{SEQ}_i = make\_next\_level(\mathcal{SEQ}_{i-1}, \mathcal{D}) // \text{Level}(i)$ 
6:   end
7:    $\mathcal{P}_{ALL} = pattern\_generation(\mathcal{SEQ}_i)$ 
8:    $\mathcal{P}_{SIG} = sig\_patterns(\mathcal{SEQ}_i, \mathcal{P}_{ALL})$ 
9:    $\mathcal{D} = get\_domi\_patterns(\mathcal{SEQ}_i, \mathcal{P}_{SIG})$ 
10:   $\mathcal{L} = \mathcal{L} \cup \mathcal{D}$ 
11:   $i++$ 
12: until  $\mathcal{D} == \emptyset \vee i == max\_level$ 
13: return  $\mathcal{L}$ 

```

3.3.2 Preprocessing

The preprocessing step of RDM prepares the input sequence, \mathcal{SEQ}_{input} for the first iteration step in the iterative process of RDM. There are two parts in the preprocessing step: (i) the sequence manipulating step, and (ii) the sequence encoding step. Neither of the two parts is required to perform on the input sequence prior to running RDM iterative steps. However, both parts are beneficial in their own ways.

During the sequence manipulating process, the user may manipulate an input sequence by using such well-known methods as stop words and stemming to remove the irrelevant terms from the input sequence. In addition, the user can incorporate the domain knowledge by providing regular expressions and their corresponding identification number. All tokens that match the regular expressions will be presented as one single entity. For example, in the sequences extracted from text documents, the user may consider that different dates do not provide any information toward the user's intended task. Hence, the user can provide a regular expression for all date strings and a corresponding identification number to represent all dates. Furthermore, the user can consider the sequence at a finer or coarser level. For example, in a text, the user may decide to use the sequence of characters instead of words as the input sequence.

Word clustering, a method that clusters similar words together into the same group, can also be applied in this step. By using cluster of words instead of a single word as feature, we can reduce the feature space size in text categorization. WordNet, a lexical database, clusters English words with the same synonym into the same group [33]. Other proposals of word clustering methodologies use various similarity measurements between two words. In [6], two words were similar if their class distributions were similar. Various other similarity measurements were used in word clustering [65]. For example, words are put into the same predefined group of documents if they appear together in the same document in at least 200 documents. Both [6] and [65] illustrated that features based on word clustering could reduce the sparseness of text categorization tasks, while the learned classifiers retained the same level of performance as those of word features. An example of manipulating input sequence in RDM framework can be seen in section 3.5.3.

The sequence encoding process simply encodes \mathcal{SEQ}_{input} into numerical format. First, the sequence encoding process assigns a unique identification number to each unique term in the input sequence. Next, it replaces all terms in all input sequences with their corresponding identification numbers. The replaced sequence is the initial sequence, \mathcal{SEQ}_0 shown in algorithm 2. Even without the user input in the sequence manipulating process, the sequence encoding process is useful for RDM. Since the sequence of number also take up lesser space to store than the word sequence, or the protein sequence, RDM can handle the numerical sequence more efficiently than other type of sequences.

3.3.3 Pattern Generation

A sliding window of length l_w moves over \mathcal{SEQ}_v ($v = 0$ initially). At each position p of the window, all possible (l_w, max_gap) -sequence patterns are generated. The number of generated patterns equals the number of combinations of tokens covered by the window along with the gap token. A bounded hash keeps count of the number of occurrences of each pattern at level v , as the sliding window moves over \mathcal{SEQ}_v . This forms the first pass over sequence \mathcal{SEQ}_v . Figure 3.1 shows the patterns generated at position 1 and 2 of the sequence.

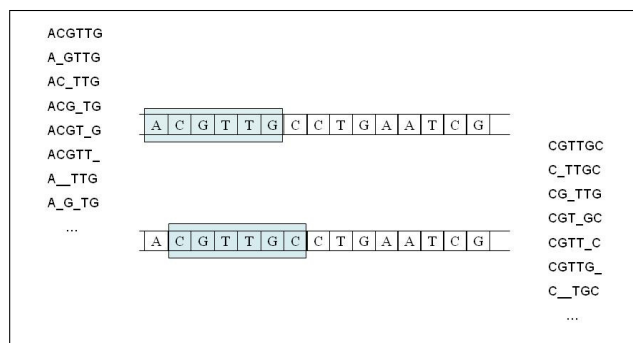


Figure 3.1: Pattern Generation Step. ($l_w = 6, max_gap = 2$)

3.3.4 Pattern Significance

The number of (l_w, max_gap) -patterns uncovered in the sequences is generally large. Many of those patterns are either very specific to a certain sequence or insignificant because they contain commonly occurring tokens. In either case, they are ineffective in capturing any stylistic attributes while adding to the computation cost of the algorithm. The “usefulness” of a pattern is computed using a statistical significance test. Applying the significance test to the set of patterns \mathcal{P}_{ALL} gives us a smaller set of *significant patterns*, \mathcal{P}_{SIG} . Patterns that are deemed insignificant are eliminated from further consideration. RDM framework does not require the user to use any specific significance test to operate. The user can specify which significance test determines the pattern usefulness to its intended task.

In the area of Pattern Mining, various significance tests for sequence patterns have been proposed. Permutation test [36] provides a simple approach for comparing the observed occurrences of a pattern with the number of likely occurrences over a random sequence. The practical application of this method requires generating a large number of random permutations of the input sequence and computing the statistics on the random permutations. If the input sequence is long this operation can be computationally very expensive. Karlin et al. [55, 56] proposed many significance tests for identifying relevant regions in protein sequences. Their approach relied on assigning scores to the tokens such that the sum of the expected scores for all the tokens was negative. Such conditions are easier to find for biological sequences as compared to text documents. In this section, we introduce five simple statistical significance tests: Chi-square test, Document Frequency test, Information

Gain test, Mutual Information test, and Unigram test.

3.3.4.1 Chi-square Test

The chi-square statistic, χ^2 , of two random variables indicates whether their distributions are different or not. The χ^2 of (A, B) , or $Chi(A, B)$ has a natural value of zero if random variables A and B are independent. Following [118], we can define the chi-square score between class c_i and \mathcal{P} , $Chi(\mathcal{P}, c_i)$, using the two-way contingency table as

$$Chi(\mathcal{P}, c_i) \equiv \frac{N(WZ - YX)^2}{(W + Y)(X + Z)(W + X)(Y + Z)} \quad (3.1)$$

where W is the number of $\mathcal{SEQ} \in c_i$ that contains \mathcal{P} , X is the number of $\mathcal{SEQ} \in \bar{c}_i$ that contains \mathcal{P} , Y is the number of $\mathcal{SEQ} \in c_i$ that does not contain \mathcal{P} , Z is the number of $\mathcal{SEQ} \in \bar{c}_i$ that does not contain \mathcal{P} , and N is the total number of sequence.

In feature selection, as suggested by [118], we can measure the usefulness of a term by averaging the scores of a term across all predefined categories. We adopt that idea and define the Chi-square score of \mathcal{P} as

$$Chi(\mathcal{P}) \equiv \sum_{i=1}^m p(c_i) Chi(\mathcal{P}, c_i) \quad (3.2)$$

From Equation 3.2, we define Chi-square test as follow. For a pattern \mathcal{P} , \mathcal{P} is significant if $Chi(\mathcal{P}) \geq K$, where K is a user-defined constant.

3.3.4.2 Document Frequency Test

In text categorization, the document frequency of a word, $DF(w)$, is the number of documents in the training corpus that contains w . Let us consider a training corpus of 10,000 documents. If there is only one document that contain a word, w , then the chance of seeing w in a future document is 0.0001. Following the above example, document frequency is applied as a feature selection methodology which considers words that less likely to appear in the future document as insignificant. Feature selection based on document frequency considers a feature, F , as relevant feature if $DF(F) \geq x$ where x is a user-define threshold. We adopt the idea for the

significance test as follow. For a pattern \mathcal{P} , \mathcal{P} is significant if $DF(\mathcal{P}) \geq K$, where K is a user-defined constant.

3.3.4.3 Information Gain Test

The machine learning community widely uses information gain for feature selection. Given that we have information regarding the value of a feature, F , Information gain of F , $IG(F)$, is the changes in information entropy from applying the known value of F to the current classification model [74]. In term of a pattern, \mathcal{P} , $IG(\mathcal{P})$ is defined as

$$IG(\mathcal{P}) = \sum_{i=1}^m -p(c_i)\log_2 p(c_i) + p(\mathcal{P} = 1) \sum_{i=1}^m p(c_i|\mathcal{P} = 1)\log_2 p(c_i|\mathcal{P} = 1) \quad (3.3)$$

$$+ p(\mathcal{P} = 0) \sum_{i=1}^m p(c_i|\mathcal{P} = 0)\log_2 p(c_i|\mathcal{P} = 0)$$

where $p(c_i)$ is the probability that \mathcal{SEQ} belongs to class c_i , $\mathcal{P} = 1$ refers to an event that \mathcal{P} appears in \mathcal{SEQ} , $\mathcal{P} = 0$ implies otherwise. Similar to Document frequency test, we define Information gain test as follow. For a pattern \mathcal{P} , \mathcal{P} is significant if $IG(\mathcal{P}) \geq K$, where K is a user-defined constant.

3.3.4.4 Mutual Information Test

Given two random variables, their Mutual information refers to the level of mutual dependence between them. In [74], the mutual information, $MI(A, B)$, where A and B are two random variables, is defined as

$$MI(A, B) = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{p(a, b)}{p(a)p(b)} \quad (3.4)$$

If A and B are independent, then $MI(A, B)$ has a natural value of zero.

Similar to Chi-square score, we redefine the Chi-square score of a pattern \mathcal{P} as

$$MI(\mathcal{P}) = \sum_{i=1}^m p(c_i)MI(\mathcal{P}, c_i) \quad (3.5)$$

From Equation 3.5, we define Mutual information test as follow. For a pattern \mathcal{P} ,

\mathcal{P} is significant if $MI(\mathcal{P}) \geq K$, where K is a user-defined constant.

3.3.4.5 Unigram Test

The assumption of unigram model is that if a pattern has a chance to appear more than a random sequence of tokens of the same length, then the pattern is significant. Recall that the set of unique tokens appearing in a set of sequences \mathcal{SEQ} is denoted by \mathcal{T} . The frequency of a token t_i appearing in \mathcal{SEQ} will be denoted by f_{t_i} . So the probability of token t_i over \mathcal{SEQ} is $P(t_i)$, where

$$P(t_i) = \frac{f_{t_i}}{\sum_{j=1}^{|\mathcal{T}|} f_{t_j}} \quad (3.6)$$

For a pattern \mathcal{P} of length l_w , the probabilities of tokens appearing in the pattern can be represented as a vector $(p_{t_1}, p_{t_2}, \dots, p_{t_{l_w}})$. Recall that a gap is represented by a special token \perp . The probability of pattern \mathcal{P} is thus given by the expression

$$\begin{aligned} \mathbf{P}(\mathcal{P}) &= P(RV_1 = t_1, RV_2 = t_2, \dots, RV_{l_w} = t_{l_w}) \\ &= p(t_1)p(t_2 | t_1) \cdots p(t_{l_w} | t_1, \dots, t_{l_w-1}) \end{aligned} \quad (3.7)$$

where RV_i is a random variable for token t_i . Assuming that the words appear independent of each other (this assumption is just for the purpose of measuring pattern significance, because if they are not, frequently co-appearing words will eventually be merged into a single token at the higher level of RDM abstraction), just the marginal probabilities for the words need to be computed, resulting in

$$\mathbf{P}(\mathcal{P}) = \prod_{i=1}^{l_w} p_{t_i} \quad (3.8)$$

The probability of a gap token, denoted as ϵ , is a user defined constant (see 3.3.5 for details). The probability of occurrence of \mathcal{P} under the independent appearance assumption (random model) is given by

$$\mathbf{P}_R(\mathcal{P}) = P(RV_1 = t_1, RV_2 = t_2, \dots, RV_{l_w} = t_{l_w}) \quad (3.9)$$

Since under the random model each token is equally likely to appear, the above expression simplifies to

$$\mathbf{P}_R(\mathcal{P}) = \left(\frac{1}{|\mathcal{T}|} \right)^{l_w}. \quad (3.10)$$

The ratio $\frac{\mathbf{P}_R(\mathcal{P})}{\mathbf{P}(\mathcal{P})}$ is used to determine significance of the pattern. If the above ratio is smaller than 1, then the pattern is considered significant, otherwise it is considered insignificant. The ratio indicates the likelihood of pattern occurrence under the random model as compared to its occurrence under the unknown observed distribution. This is similar in essence to the *log-likelihood ratio test*, with null hypothesis (H_0), that the observed distribution is similar to the random distribution. The alternate hypothesis H_1 states otherwise. The log-likelihood ratio is given by the expression

$$\mathbf{LRT} = -2 \log_e \left(\frac{\mathcal{L}_R(\theta)}{\mathcal{L}_O(\theta)} \right) \quad (3.11)$$

where $\mathcal{L}_R(\theta)$ is the likelihood function under the random model and $\mathcal{L}_O(\theta)$ is the likelihood for the observed distribution. H_0 is a special case of H_1 , since it has fewer parameters (captured by θ) as compared to the more general alternate hypothesis. In practice, computational cost of the pattern generation step can be reduced by checking whether a sequence of tokens in the current window have the ratio of $\frac{\mathbf{P}_R(\mathcal{P})}{\mathbf{P}(\mathcal{P})}$ smaller than 1 or not. If not, then we can conclude that no pattern generated from this window is significant.

3.3.5 Dominant Patterns

After the significant patterns at level v are determined, a second pass is made over the sequence of tokens S_v . At each position in the sequence, the tokens in the significant patterns are matched against the tokens in the sequence. The matching score is defined as the conditional probability of a match given two symbols, i.e., if $\mathcal{P}[i]$ and $S_v[j]$ are the same then the conditional probability of a match is 1. On the other hand, if $\mathcal{P}[i] = \perp$ then the conditional probability is ϵ . The matching score

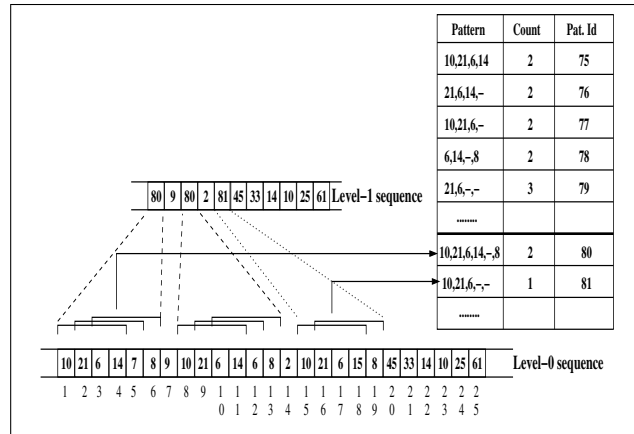


Figure 3.2: Sequence Re-writing Step

can be computed as follows:

$$score(\mathcal{P}[i], S_v[j]) = \begin{cases} 1 & \text{if } \mathcal{P}[i] = S_v[j] \\ \epsilon & \text{if } \mathcal{P}[i] = \perp, \epsilon < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

where $\mathcal{P}[i]$ is the i^{th} token of the pattern and j is the corresponding index over sequence S . ϵ is intended to capture the notion that a \perp symbol is not as good as an exact match but much better than a mismatch. The value of ϵ is user defined, which is set to 0.95 in our experiments to favor a match with the gap token. The total score for a pattern, starting at index j in S , is given by

$$score(\mathcal{P}, S_v[j]) = \sum_{i=1}^{|\mathcal{P}|} score(\mathcal{P}[i], S_v[j+i]). \quad (3.13)$$

The pattern that has the highest score starting at location j in the input sequence is termed as the **dominant pattern** starting at position j . In other words, this is a pattern x defined by the expression $\operatorname{argmax}_{x \in S_v} score(x, S_v[j])$. The term dominant pattern reflects the fact that this pattern dominates over all other significant patterns for this position in the sequence. Two dominant patterns that are placed in tandem can be merged to form longer dominant patterns. The merging process is continued till no further dominant patterns can be merged. An example of the merging process is shown in Figure 3.2. A new token is assigned to each dominant pattern. During

this second pass of the sequence at level v , the sequence for level $v + 1$ is generated. The sequence corresponding to a dominant pattern is replaced by the new token for this dominant pattern. When a dominant pattern is not found at position j , the original token is copied from sequence S_v to the new sequence S_{v+1} . Figure 3.2 illustrates this step.

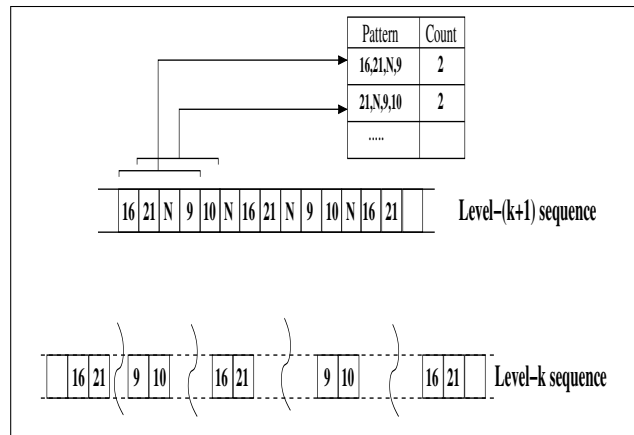


Figure 3.3: Removing Noisy Tokens for Long Range Patterns

As the RDM algorithm generates subsequent levels, certain tokens get carried over from lower levels without participating in any dominant patterns at higher levels. Such tokens are termed “noisy” for the following reasons. First, they do not contribute to any patterns at these levels. Second, they obstruct the discovery of patterns that are separated by a long sequence of noisy tokens. Patterns separated by noisy tokens are called *long range patterns*. These long range patterns can be captured only if the noisy tokens lying in between them can be collapsed. As a result, at each level, we collapse contiguous sequence of tokens that have not resulted in new dominant patterns for the last k levels, into a special noise token. k is selected using the tuning dataset (see Section 3.5). Figure 3.3 illustrates the process of collapsing noise tokens into a single special token N . Once the noise tokens are collapsed, distant tokens can now fall within the same window, leading to more patterns being discovered at higher levels. The set of dominant patterns D_v for level v form the features for this level. This iterative process of deriving level $v + 1$ sequence from level v sequence is carried on till no further dominant patterns are found or $v + 1$ has reached a user predefined maximum value. The sets of features extracted are

utilized by an ensemble of classifiers.

3.3.6 Settings parameters for RDM

We have designed RDM as the general model for all sequential pattern mining. The expressive power of RDM is defined by its parameter settings. The following list summarizes the RDM parameters.

1. **Window size** is the size of the sliding window, which defines the maximum length of generated pattern.
2. **Number of gaps** specifies the degree of approximated match between a pattern and an input sequence.
3. **Significance test** is used to identify significant patterns. As mentioned in section 3.3.4, different tests yield different sets of significance patterns.
4. **Dominant pattern matching matrix** defines matching function and its scoring matrix. Different functions result in different dominant patterns.
5. **Feature item** must be defined to distinguish which tokens are to be considered features. If all generated patterns are denoted as features, then the classifier at each level of RDM is equivalent to a specific N-gram model, depending on the size of the window. If the window size is one, then all generated patterns are simply the tokens in the training data set. Hence, the classifier based on such feature implements a unigram model. If one chooses significant patterns as features, then the classifier at each level of RDM is simply a specific N-gram model with feature selection method depending on the size of the window and the significant tests.
6. A **learning method** must be provided so that a classifier can be created at each level of iteration.
7. Other parameters are **termination conditions (maximum number of iteration)**, **noise token settings**, and **confidence values of classifiers**.

In section 3.5, we illustrate how many of these choices affect the performance of RDM.

3.4 Recursive Data Mining for Text Categorization

In supervised learning setting, we can consider it as a two-phase problem: the training phase and the testing phase. In the training phase, we train a model based on the labeled training dataset. In the testing phase, we apply the learned model on the testing dataset for evaluation.

3.4.1 Training Phase

For RDM framework, the training phase involves using dominant patterns generated at each level to construct an ensemble of classifiers ($\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_{max_level}$), one for each level. By choosing the dominant patterns as features, we select the most relevant patterns, as well as filter out insignificant patterns, as features. The classifiers can be created using any machine learning method, such as Naïve Bayes or Support Vector Machine. Given a set of text documents \mathcal{SEQ}_{tr} , along with the labels r_1, r_2, \dots, r_v of all possible classes, dominant patterns are generated for each document starting at level 0 up to level max_level . The union of all tokens in \mathcal{T} and dominant patterns at a level v across all documents in \mathcal{SEQ}_{tr} forms the set of feature for classifier \mathcal{C}_v . For the ensemble of classifiers, the final prediction value is the weighted sum of the class prediction of individual classifier. Each classifier is assigned a weight that reflects the confidence of the classifier. In order to determine this confidence value, the set \mathcal{SEQ}_{tr} is further split into a training set \mathcal{SEQ}_{new} and a tuning set. Each classifier in the ensemble trains its model based on \mathcal{SEQ}_{new} . The accuracy of the classifier on the tuning set determines the confidence of classifier \mathcal{C}_i as

$$conf(\mathcal{C}_i) = \frac{accuracy(\mathcal{C}_i)}{\sum_{j=1}^{max_levels} accuracy(\mathcal{C}_j)}. \quad (3.14)$$

3.4.2 Testing Phase

After the training phase discovers features from the training data, the testing phase finds occurrences of those features in the test data. The testing phase as such follows the training phase in terms of preprocessing and level by level operating strategy. If a dominant pattern X was discovered at $level(Y)$ during the training phase, then it can be only applied to $level(Y)$ in the testing phase. Initially, the

frequencies of tokens and $level(0)$ dominant patterns are counted over the $level(0)$ test sequence. This vector of frequencies forms the feature vector at $level(0)$. Then, the next level sequence is generated by substituting the token of the best matching pattern at every position in the $level(0)$ test sequence. It should be noted that if the best match has a score below the user defined threshold then the token at $level(0)$ is carried over to $level(1)$. Now the occurrences of the dominant patterns at $level(1)$ are counted over $level(1)$ test sequence. This process continues till all levels of dominant patterns are exhausted. Each classifier in the ensemble classifies the test data and the final prediction value is assigned based on the following weighing scheme:

$$\mathbf{P}(C | x) = \sum_{i=1}^{max_levels} conf(\mathcal{C}_i) \times \mathbf{P}_{\mathcal{C}_i}(C | x) \quad (3.15)$$

where x is a test sequence and $\mathbf{P}_{\mathcal{C}_i}(C | x)$ is the prediction value assigned by classifier \mathcal{C}_i .

3.5 Experiments and Results

There are four sets of experiments presented in section 3.5.2, section 3.5.3, section 3.5.4, and section 3.5.5. For the first set of experiments, we studied the effects of various significance tests on RDM. We experimented with unigram test, document frequency test, information gain test, mutual information test, and chi-square test (described in section 3.3.4). The second set of experiments was designed to illustrate that semantic tool such as part-of-speech tagging and word clustering could improve the performance of RDM on role identification tasks. Within the same task, we also showed that RDM could take various abstraction levels of sequence input, such as sequence of characters from text document. We used JTextPro for part-of-speech tagging [85]. We adopted distributional clustering method in [6] for word clustering. In addition, we provided a word clustering based on frequency as a baseline comparison. For the first and section set of experiments, we used RDM with NB and RDM with SVM on the role identification task.

In the third set of experiments, we used RDM to extract the pattern of ordered words for the role identification task. We showed that classifiers based on

RDM performed better than comparable classifiers such as Naïve Bayes (NB), Support Vector Machines (SVM) and Predictive Association Rule based (CPAR [120], which, authors claimed, combines the advantages of associative and traditional rule-based classifiers). Support Vector Machines based classifiers were shown by [51] to perform well for text classification tasks. We used SVMLight as the SVM implementation [50], and IlliMine package for CPAR [120]. RDM did not use any semantic tools (part-of-speech tagging or synonym groups) in order to extract patterns that later served as features for the classifiers. As a result, we compared RDM with other techniques that did not utilize domain or semantic knowledge either. In the last set of experiments, we studied the effects of the training set sizes and the influence of the sliding window size on the performance of RDM on role identification tasks. We focused our attention to RDM with NB and used NB as a base line comparison. A brief introduction to the Enron dataset used for running the experiments is provided before the discussion on the experimental setup.

3.5.1 Data Preparation and Experimental Setup

Experiments were performed on the March 2, 2004 version of Enron dataset, distributed by William Cohen [22]. The dataset was cleaned to eliminate attachments, quoted text and tables from the body of the email messages and header fields from the email. No effort was made to correct spelling errors or to expand abbreviations in an attempt to reduce the noise in the data. During the preprocessing steps for the experiments in section 3.5.4, we applied Porter stemming from the Snowball Project [86], on the text documents in the input because it improves the overall performance of all classifiers on the tuning dataset.

For our purpose of identifying roles, employees were partitioned into groups based on their organizational role in Enron, as suggested in [1]. Only the roles *CEO*, *Manager*, *Trader* and *Vice-president* were used in our experiments because a large number of employees were designated with these roles. Since we were concerned with identifying roles based on messages sent by employees, we only dealt with the messages in the *Sent* folder of each participant. For each of the roles, the total number of emails is summarized in the first column of Table 3.1 which contains

all the messages in the *Sent* folder. The second column Table 3.1 indicates the number of emails whose words have been tagged with their part-of-speech. All the experiments were performed on the first set of data in Table 3.1, except for the experiment in 3.5.3 that required part-of-speech tags. Finally, each word in an email was considered a token, and each email represented one sequence.

Table 3.1: Dataset for Role Identification task

Role	First set	Second set
CEO	1260	1092
Manager	1752	1751
Trader	816	816
VP	1643	1494
Total	5417	5153

The RDM algorithm required a few parameters to be set for the classification model. They included 1) the size of the window, 2) the maximum number of gaps allowed in the window, 3) the weights assigned to the classifier at each level, 4) the parameter k used to eliminate noisy tokens. A greedy search over the parameter space was conducted to determine the best set of parameter values. To compute the parameter values, the training set was further split into two parts. A classifier was trained on the larger part, and tuned on the smaller part (called the tuning set). We classified a token as a noise token if such a token had not been part of any dominant pattern for the past two iterations. The termination conditions of RDM were either when no dominant pattern was found, or when the maximum *level* – 20 iteration step was completed. Dominant patterns were used as features. *F-measure*, also called F-score, was used to evaluate performance of the classifiers. For our experiments, F-measure was defined as $\frac{2*precision*recall}{precision+recall}$.

For role identification task, we defined the task under two classification settings: binary and multi-class. In the binary classification setting, given a test message m , the task is to answer the question “*is message m sent by a person with role r*”?, where $r \in \mathcal{R} = \{\text{CEO, Manager, Trader, Vice-president}\}$. The training set was divided in such a way that all messages belonging to role r formed the positive class and all messages belonging to $\mathcal{R} \setminus r$ formed the negative class, where $\mathcal{R} \setminus r$ denotes the set difference operation $\mathcal{R} - r$ (\mathcal{R} minus r). The second set of results compared

the performance under the multi-class classification setting, wherein the task was to answer the question “*which is the most likely role, out of roles R_1, \dots, R_n , for sender of message m ?*” For NB and RDM with NB, the training data was split into four groups and probabilities computed for each of the roles. For SVM and RDM with SVM, four sets of datasets were generated, one each for role $(r, \mathcal{R} \setminus r)$ pairs.

3.5.2 Comparative Study of Significance Tests

We performed a comparative study on the effects of unigram test (Uni), document frequency test (DF), information gain test (IG), mutual information test (MI), and chi-square test (Chi) on RDM. For this study, we used RDM with both Naïve Bayes, and SVM as the ensemble classifiers. For DF, IG, MI and Chi, we picked the significance test threshold such that the patterns with the highest 10% score of each respective tests were the significant patterns. The results are shown in Table 3.2. The results showed that unigram and chi-square tests perform best. However, the

Table 3.2: Comparison experiment of five significance tests: Unigram test (Uni), Document Frequency test (DF), Information Gain test (IG), Mutual Information test (MI), Chi-Square test (Chi). The F-measure value in bold font is the highest F-measure of the corresponding role.

Role	Classifier	Type of significance test				
		Uni	DF	IG	MI	Chi
CEO	RDM-NB	0.83	0.83	0.82	0.82	0.83
	RDM-SVM	0.82	0.82	0.82	0.82	0.82
Manager	RDM-NB	0.91	0.91	0.91	0.91	0.91
	RDM-SVM	0.94	0.94	0.94	0.94	0.94
Trader	RDM-NB	0.75	0.75	0.73	0.73	0.76
	RDM-SVM	0.68	0.68	0.68	0.68	0.68
Vice-President	RDM-NB	0.88	0.87	0.87	0.87	0.88
	RDM-SVM	0.81	0.81	0.81	0.81	0.81

results showed that all five tests performed equally well, because RDM did not take the level of significant into account. The significant patterns ranking based on all the tests were different. But, the set of patterns within the top 10% ranking were rather similar. Thus, they had the same set of significant patterns. Hence, the resulted dominant patterns were fairly similar across all significance tests.

3.5.3 Sequence Manipulating Step

In this section, we describe experiments performed to illustrate the power of sequence manipulation in the preprocessing step of RDM. We focused the experiments on input in the form of the text documents to show how the user can manipulate input sequences of text for text categorization task. For each sequence manipulation, we used RDM-NB and RDM-SVM to construct learning models for binary classification task explained in 3.5.1. To control the sequence manipulation on the mentioned topics, we neither performed any stemming nor removed any stop word in the documents. We used F-measure to evaluate all classifiers. For the significance test, we used unigram test on all experiments due to its low computational cost and its good performance on tuning dataset.

There are two experiments described in this section. In the first experiment, we illustrated how three different levels of abstraction of input sequence affected the performance of RDM. In the second experiment, we showed the effects of word clustering methods on the RDM performance. We performed three-fold cross-validation on both set of experiments. However, some email messages were not successfully tagged by the part-of-speech tagger. The actual dataset used for this second experiment is shown in the second column in Table 3.1.

For the first experiment, we explain below each sequence abstraction that was applied to the initial sequence in the first iteration level of RDM.

1. **Sequence of Characters:** A sequence of words is also the sequence of characters. A word is a pattern of characters. In a language that an ordered set of characters forms a word, many sequences of characters can appear in different words. For example, the pattern “U,S,E” can appear in the words “use”, “useful”, “fuse”, “refuse”, etc. The idea of considering the sentence at the level of characters is that RDM should be able to pick up patterns of words from the sequences of characters and use them for the role identification task. For the sequence of characters, we considered only letters, a space, numerical and punctuation marks, and ignore all other characters.
2. **Sequence of Words:** As discussed earlier, the usage of the patterns of words on text categorization task is not a new idea. We used sequence of words as the

initial sequence in the previous experiments, where RDM captured patterns of words and used those significant patterns as features. Hence, we included the sequence of words in this experiment as a baseline for comparison.

3. **Sequence of clustered words:** In many languages, different words may have the same meaning if used in the same context. We can cluster these words together using various similarity measures such as frequency, class distribution, or use the semantic knowledge such as part of speech. If we assign each cluster with a unique identification number, and replace a word in the string with its cluster identification number, the resulting sequences will consist of symbols representing clustered words. The number of patterns generated from such a sequence will be smaller than the total number of patterns generated from sequence of words. Also, the frequency of such patterns will increase compared to frequency of patterns of words. In short, we will have fewer but more frequent patterns. For this experiment, we used the class distribution of words as a mean for similarity comparison for word clustering. We reduced the number of distinct tokens in input sequence from 12400 words on average down to 1000 clusters of words.

The experiment result in Table 3.3 shows the effects of running RDM on different levels of input sequences in the binary classification task. On the roles of CEO, Manager and Vice-President, the input of sequence of words performed best. On the role of Trader, the input of sequence of clustered words performed best. A feature base on the frequency of a pattern of characters had no relevant information and did not help with the text categorization task as shown in Table 3.3. On the input of sequence of characters, the F-measures of RDM-SVM are 0.0 on all roles because it classifies every document as one class. Even though, RDM-NB did not classify every document as one class, the accuracies on test datasets were lower than that of RDM-SVM. For example, in the role of CEO, the accuracy of RDM-NB model is 0.75 while the accuracy of RDM-SVM model is 0.77. As for the sequence of clustered words, the F-measures of RDM-NB and RDM-SVM on the sequence of clustered words were as good as those on the sequence of words. As for the best result on role of Trader, some examples of the clustered words according to this role were

{taken, taking}, {yessssssss, laugh, excited}, {ticket, return}, and {due, before}. Clusters such as {taken, taking} were helpful in increasing the pattern frequency similar to stemming process. Clusters, such as {ticket, return} and {due, before} helped detecting patterns such as “return ticket” and “due before”. The downside of distributional word clustering was that not all clusters displays meaningful grouping, and some words such as “eat”, “ate” and “eaten” were in different clusters.

Table 3.3: Comparison experiment of three abstraction level of input sequence: sequence of characters, sequence of words, and sequence of clustered words. The F-measure value in bold font is the highest F-measure of the corresponding role.

Sequence of	Classifier	Binary Classification of Role			
		CEO	Manager	Trader	Vice-President
Characters	RDM-NB	0.29	0.29	0.35	0.46
	RDM-SVM	0.0	0.0	0.0	0.0
Words	RDM-NB	0.83	0.91	0.75	0.88
	RDM-SVM	0.82	0.94	0.68	0.81
Clustered words	RDM-NB	0.83	0.88	0.75	0.82
	RDM-SVM	0.79	0.89	0.52	0.80

The resulted clusters in general depend on the similarity measurement as well as the number of clusters specified by the user. The different numbers of clusters of words directly influence the significant patterns found by RDM. In the second set of experiments, we ran RDM on the binary classification tasks on the following similarity measurements with various numbers of clusters.

1. **Frequency:** Intuitively, words that appear either frequently or infrequently throughout the training corpus are unlikely to contribute to identifying the role of a document’s author. However, ignoring them completely can also remove certain patterns where these words appear. Specifically, given two constants F_1 and F_2 where $F_1 \geq F_2$, every term that occurs more than F_1 times is clustered in high frequency group, while every term that occurs at most F_2 times is clustered in low frequency group. Every other term with occurrence frequency in between those two constants belongs to a cluster of its own. By introducing the clusters of words with high/low frequency, RDM can capture similar sequences which use these words while limit the irrelevant information

shown in patterns to minimum. We used this frequency based word clustering method as a baseline for comparison in this experiment.

2. **Part-of-Speech Tagging:** In English language, human has an idea of what a sentence tries to convey even though some words are left out. Part-of-Speech tagging clusters words with the same part of speech together. For example, in a sentence “Let us meet at Four Season’s Hotel in New York City on 01/02/03”, Part-of-Speech tagging version of this sentence is “Let us meet at Noun-Phrase on DATE”. We can understand that both versions try to set up a meeting. We adopted this idea and applied Part-of-Speech tagging to the low frequency words. Specifically, given a constant F , every word that occurs at most F times was clustered according to its part-of-speech tag. Every other word (that is a word with occurrence frequency above this constant) belonged to a cluster of its own. Part-of-Speech reduces the sparseness of the generated patterns while retaining the semantic and syntactic information in each pattern.
3. **Word distribution:** In [6], distributional clustering clustered two words together if their distributions on predefined classes were similar. Similar to RDM, this method does not require external knowledge and is language independent. The clustered words approach reduces the number of all generated patterns and increases their occurrence counts. A sample output of distributional clustering can be seen in the result discussion of the previous experiment.

The experiment results shown in Figure 3.4 demonstrate the effects of using the different similarity measurement for word clustering and the impact of different size of clusters on the quality of patterns found by RDM. The x-axis defines the number of distinct tokens in \mathcal{SEQ}_0 . It is also the number of word clusters. The marking on each curve indicates the collected data point. POS stands for part-of-speech tagging. FR stands for frequency clustering. WC stands for distribution clustering. RN and RS stand for RDM with NB and RDM with SVM, respectively. Note that only distribution word clustering allows the user to specifically choose the numbers of results clustered. POS and Frequency models rely on occurrence count of each word. Thus, data collection points for all three methods are different. On all

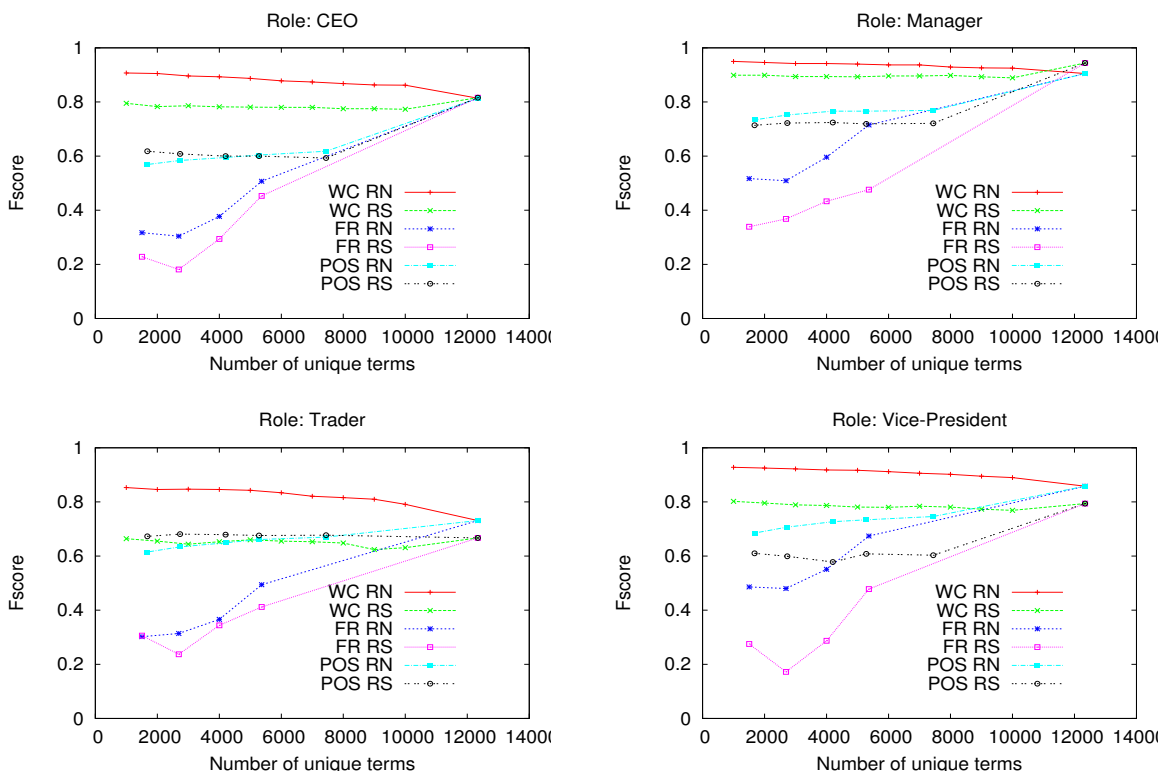


Figure 3.4: Comparison of three word clustering methods on various size of clusters for RDM applications on role identifications. WC stands for distribution clustering, POS stands for part-of-speech tagging and FR stands for frequency clustering. RN and RS stand for RDM with NB and RDM with SVM respectively.

roles, distributional clustering improved the performances of RDM with NB as the number of word clusters decreased. For RDM with SVM, distributional clustering only improved the performance of RDM on the role of Vice-President. However, on all other roles the F-measure of RDM with SVM only dropped by at most 0.035. POS performed well only on the role of trader on RDM with SVM.

3.5.4 Performances of RDM

The extracted patterns from RDM are varied in length - depending on the dominant patterns. The RDM framework is decided to extend the pattern length from the given window size if the extended patterns are created from combining two or more significant patterns. RDM does not extend the pattern if it is statistically better than a fixed size pattern. However, it has not been empirically proven. We

Table 3.4: The performances of RDM, Bigram and Trigram models on binary classification of the role identification tasks. The f-scores are used as the evaluation means

Role	Naïve Bayes Models based on Patterns from		
	Bigram	Trigram	RDM
CEO	0.78	0.6	0.83
Man	0.85	0.71	0.91
Tra	0.58	0.51	0.75
Vic	0.72	0.63	0.88

showed that traditional sequential pattern does not perform well on noisy Enron dataset. The following experiment compared the RDM-NB against the Naïve Bayes (NB) models generated based on the bigrams and trigrams. We ran the 4-gram models. However, the 4-gram patterns recovered were too infrequent to be significant. Specifically, there are less than 10 4-gram patterns which has the minimum support of at least 5%. Hence, we did not create the NB model based on 4-gram patterns. We used unigram test on RDM-NB and low frequency cut off were applied as feature selection on Naïve Bayes models. The 3-fold cross validation experiments were performed on the binary classification of the role identification tasks of CEO, Manager, Traders and Vice-president. The average f-score for each model is shown in Table 3.4. The results from Table 3.4 showed that the Naïve Bayes model based on the patterns extracted from RDM framework performed better than those based on bigram and trigram. Moreover, as the length grew, the classifiers performed worse. Since the Enron dataset is noisy, only small number of significant sequential patterns was recovered. Since RDM recovered approximated patterns, it can discard noisy tokens. As evidenced by the results, approximated patterns extracted by RDM performed better than sequential patterns in such a noisy dataset.

To show that the pattern extracted from RDM can help improve the performance in role identification task, we also performed the *paired t-test* for statistical significance under multi-class classification setting. The paired t-test provides a hypothesis test of the difference between population means for a pair of random samples whose differences are approximately normally distributed. Note that a pair of samples, each of which may not be from a normal distribution, often yields

differences that are normally distributed. The null hypothesis H_0 states that the difference in performance between RDM and the other methods is not significant. In other words, H_0 states that the two methods perform equally well. The alternate hypothesis, states otherwise. Two 20-fold-cross-validation experiments were performed on the data. In the first 20-fold cross validation experiment, the accuracy results obtained therein were used for the t-test, where SVM and CPAR were compared against RDM with SVM (denoted as RDM-SVM and RS), and NB was compared against RDM with NB (denoted as RDM-NB and RN). For the significance test, we used unigram test on all experiments due to its low computational cost, and it performs well on tuning dataset. The results of paired t-test are shown in Table 3.5. Based on the p -value in Table 3.5 we reject the null hypothesis, indicating a definite improvement provided by RDM. The confidence interval for the mean difference shows that the improvement lies between 1.8% and 3% for RDM-NB compared to NB alone, whereas RDM-SVM when compared to SVM (and CPAR) provides the improvement between 8% and 10%.

Table 3.5: Results of paired t-test. RN stands for RDM with NB and RS stands for RDM with SVM

	Classifier Pair		
	RN vs NB	RS vs SVM	RS vs CPAR
Mean difference	0.02393	0.08927	0.09329
Std. Dev. of (\bar{d})	0.002525	0.00434	0.00535
t-statistic (df=19)	9.48	20.55	17.45
p-value	1.23E-08	1.94E-14	3.74E-13
95% confidence value	(0.019 - 0.029)	(0.082 - 0.098)	(0.082 - 0.105)

In the second 20-fold cross validation experiment, we compared RDM-NB against NB models which used approximated sequential patterns of length 3 and 4 as features. We limited the number of approximated sequential patterns recovered using the minimum support, i.e. we ignored all patterns who had less than $x\%$ support. $x\%$ support was determined using the same tuning set prepared for RDM. The experiment results, shown in Table 3.6, indicated that RDM performed over 30% than approximated sequential patterns.

To further analyze the performance of RDM, we computed the Root Mean

Table 3.6: Results of paired t-test. RN stands for RDM with NB, NB3 stands for NB model using approximated sequential pattern of length 3, and NB4 stands for NB model using approximated sequential pattern of length 4

	Classifier Pair	
	RN vs NB3	RN vs NB4
Mean difference	0.36	0.36
Std. Dev. of (\bar{d})	0.03	0.029
t-statistic (df=19)	53.9	54.58
p-value	1.51e-09	1.50e-09
95% confidence value	(0.3484 - 0.3766)	(0.3496 - 0.3774)

Square Error (RMSE) for NB and RDM with NB. The RMSE is computed using the expression

$$RMSE(\mathcal{T}_{test}) = \sqrt{\frac{\sum_{i=1}^{|\mathcal{T}_{test}|} (1 - P(r | \mathcal{T}_{test}^i))^2}{|\mathcal{T}_{test}|}} \quad (3.16)$$

where \mathcal{T}_{test}^i is the i^{th} document in the test set and $r = \operatorname{argmax}_c P(c | \mathcal{T}_{test}^i)$. The lower the RMSE value, the more confident the classifier is in its prediction. Since the decision function value from SVMLight could not be converted to an error term, the plot in Figure 3.5 does not show comparison with SVM. Similarly, CPAR does not provide any comparable measure. Figure 3.5 shows that RDM with NB is more confident in its predictions even when the F-measure's for RDM with NB and NB might be very close for a certain role.

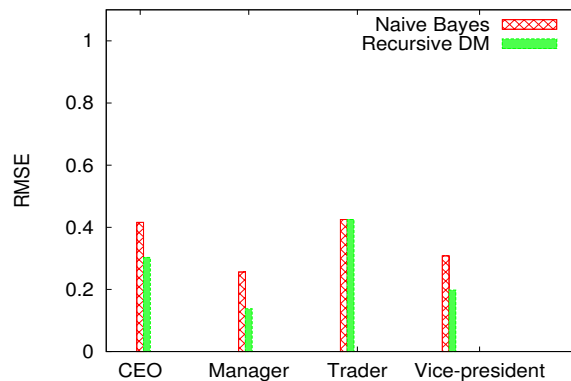


Figure 3.5: Binary Classification – RMSE Comparison

For the final test we divided each role into two parts based on the users. For

instance, the folders of *Jeff Skillings*, *David Delainey* and *John Lavorato* form the CEO group. It should be noted that a CEO of Enron subsidiaries was also considered as an Enron CEO for our experiments. The first part, namely training set, contains messages from *John Lavorato*, *David Delainey* while messages from *Jeff Skillings* form the second part (test set). An RDM based classifier is trained using messages in the first part and tested on messages in the second part. In this experiment we analyze the performance of the classifier for a member whose messages are not in the training set. The results for different roles are shown in Figure 3.6. The test set size is gradually increased and the accuracy is noted. Notice that for the roles Manager, Trader and Vice-president the accuracy increases with larger number of message. The opposite effect is observed for the role of CEO. On examining the messages for the CEO, we observed that most of the messages were written by secretaries. This explains the poor performance of classifiers for this role. From the results, even after size of the super-message reach 20, the accuracies of the RDM framework are still less than 50% in most roles except for that of Trader. However, it was evidenced that the classifier based on word features alone showed no sign of learning.

3.5.5 Effect of Parameter Changes

In this section, we take a quick look at the effects of varying certain parameters within RDM on the role identification tasks. For this section, we used accuracy for evaluation purposes. Figure 3.5.5 shows the variation in accuracy of RDM with NB on the increasing training set size in the binary setting of role identification task. The training set for each of the roles is increased in steps of 10% of the total training set size. From these results we observed that RDM with NB consistently performs as good as or better than NB. Moreover, they showed that both classifiers were quite robust and attained a fairly high accuracy even for smaller training set sizes.

Figure 3.5.5 captures the effect of varying window size on overall accuracy of RDM with NB in the multi-class setting of role identification task. The maximum number of gaps is set to 1. Figure 3.5.5 shows that the accuracy was best for a window size of 3 and reduced as the window size was increased. This result was intuitive as larger significant patterns were captured by merging smaller significant

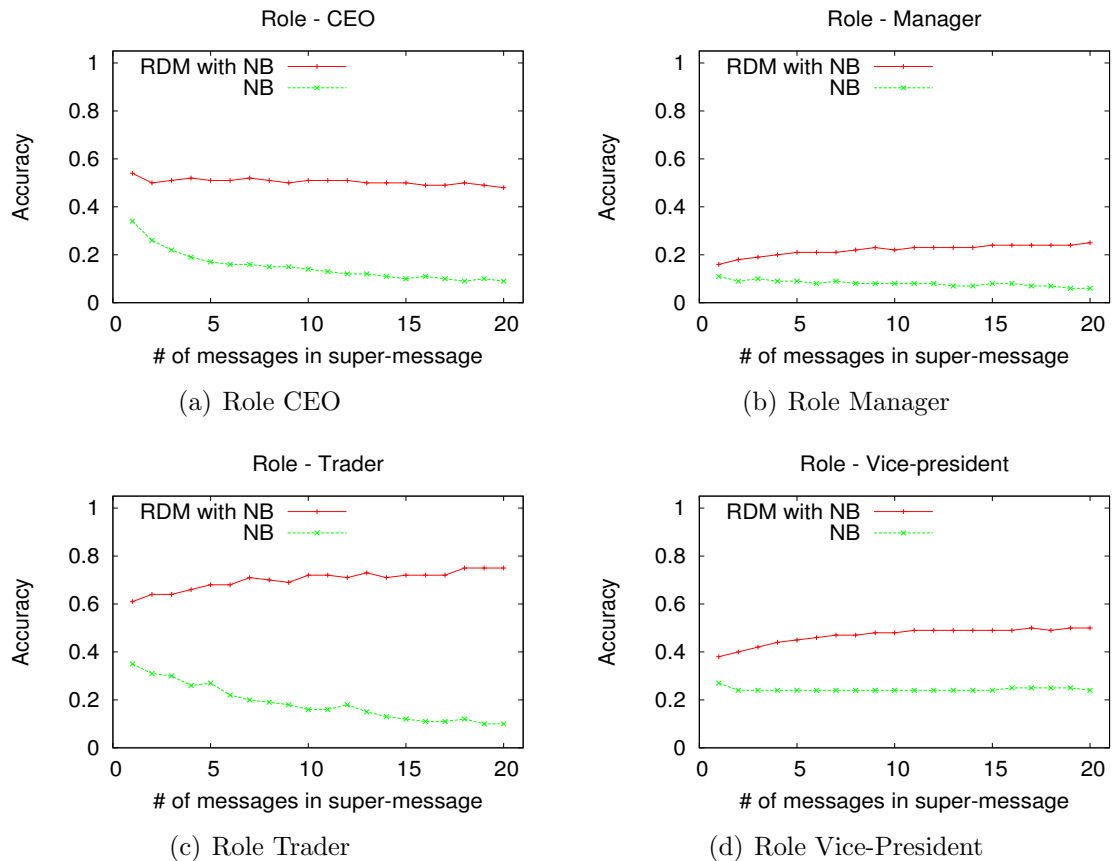
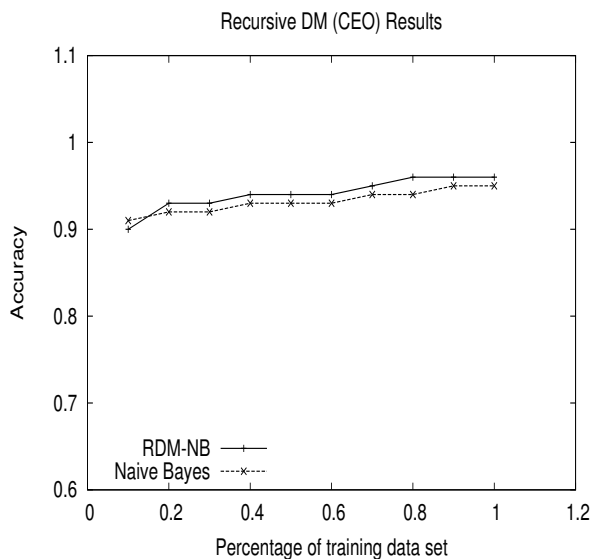


Figure 3.6: Classification Probability over Unseen Message Folder

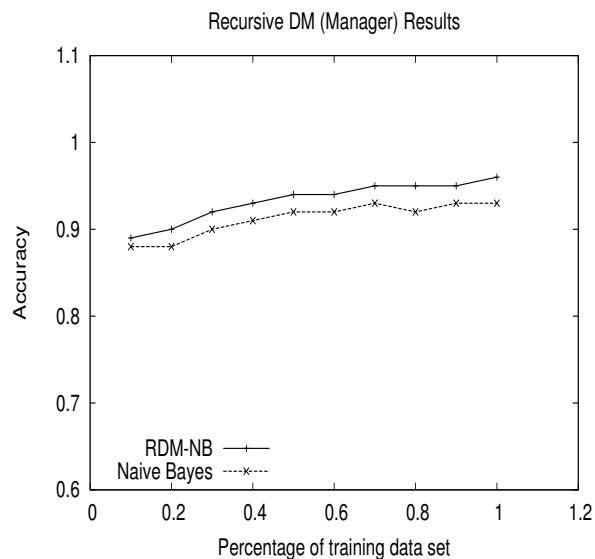
patterns, whereas on the other hand smaller patterns could not be captured using a large window size.

3.6 Conclusion

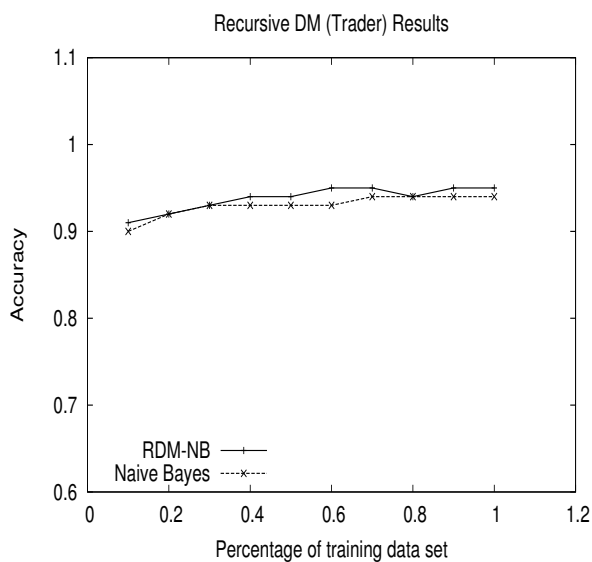
We propose a general framework for feature extraction from a sequence of tokens. The framework is based on the idea of capturing statistically significant sequence patterns at increasing levels of generalization. These patterns act as features for an ensemble of classifiers, one at each level. The proposed method is simple and flexible, hence, it can be applied to a range of applications. We applied it to capturing stylistic patterns in the Enron dataset and used those patterns for identifying the organizational roles of authors. The method, in its current state, is devoid of any semantic knowledge, which can be easily incorporated to identify semantically related patterns. We also show that RDM, using the greedy search to extract the



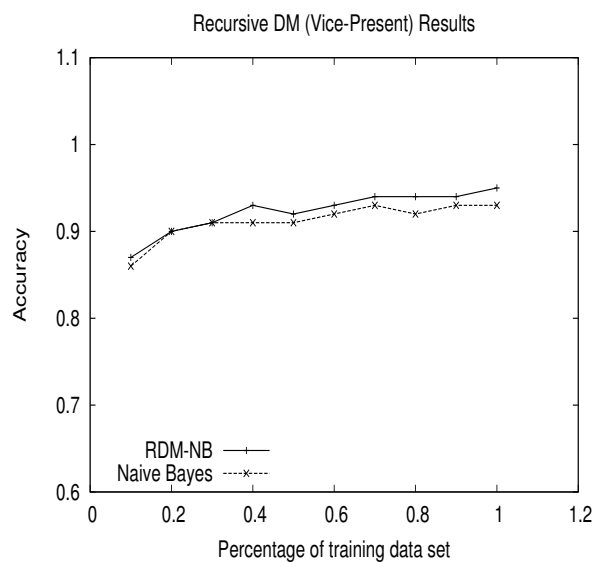
(a) Role - CEO



(b) Role - Manager



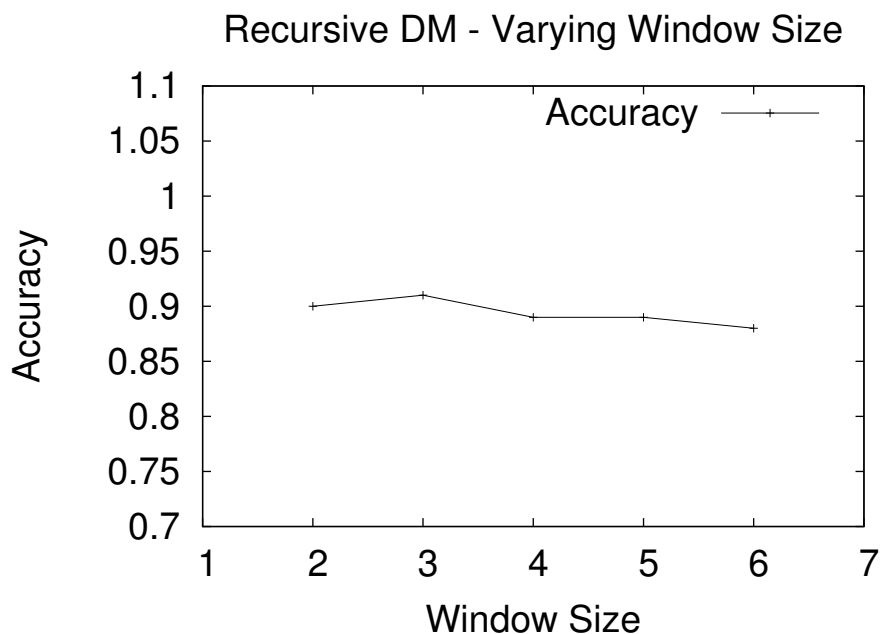
(c) Role - Trader



(d) Role - Vice-president

sequential pattern at each level, can extract a much longer pattern than the traditional sequential pattern such as N-grams. Overall, patterns recovered from RDM were shown to be more relevant to the role identification task than fixed-length sequential patterns and fixed-length approximated sequential patterns.

We illustrated that techniques such as part of speech tagging and word clustering can augment our approach. The improved performance resulting from applying word clustering method to RDM indicates that using synonym dictionaries is likely



to improve the performance of RDM on text categorization task. It should be noted that words are the most logical basic unit of many languages in which a native speaker can understand their meanings and what they convey when put in a sentence. Each word has a meaning in itself. Still, in many languages, word's meaning depends also on the context in which it is used in a sentence. Thus, any two documents with different content are likely to use some words in different meanings. Assigning different identifiers to words in different context indiscriminately would hopelessly increase the number of patterns for RDM. However, a contextual tagging combined with clustering may lead to large improvement in quality of RDM classification. In the experiment on pattern significance, we consider all significant patterns as equal so the resultant dominant patterns are similar. However, if we take the significance score of each pattern into account while creating dominant patterns, the performance of RDM is likely to improve. Based on the success of the method on a noisy dataset, we believe that the method can perform better on cleaner datasets and on other application areas such as grouping gene products by their families.

CHAPTER 4

Burstiness in Text Mining

4.1 Introduction

As discussed in Chapters 1 and 2, burst detection and burstiness concepts have been studied in various areas. For Text Mining, we use these concepts to detect the burst interval of a term in sequence of documents as well as how irregularity the patterns are in a sequential data set. Commercialized products such as Google Trends and BlogPulse produced the trend curves. However, the displayed results focus on the visualization of the data, showing the trends on the timeline. Burst detection allows further analysis on the data. In this chapter, we present two frameworks that use the definition of burstiness for feature analysis. We apply the frameworks to extract information and analyze the streams of documents on ACM Dataset. Finally, we illustrate how we can use the burstiness to improve the document clustering performances.

This chapter is organized as follows. Section 4.2 provides additional notations and definitions used in this chapter. Our definitions of bursty period, term’s burstiness and its integration into text mining applications are given in section 4.3. We introduce a framework to create a *bursty distance measurement* that improves utilization of bursty period and bursty score in section 4.5. The framework is based up on our burst detection method that focuses on local metrics of burstiness using kernel density estimation.

4.2 Preliminaries

For our work, we consider a text corpus, denoted \mathcal{S} , as a sequence of documents which appear in n consecutive time steps. Formally, \mathcal{S} is defined as $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$, where s_i refers to the set of documents that appear in the i^{th} time step. t_d is the time step when document, d , appears in \mathcal{S} . In turn, we define

A. Hoonlor, B. K. Szymanski, M. J. Zaki, and V. Chaoji, “Document Clustering with Bursty Information,” *Computing and Informatics*, Bratislava: Slovak University Press, (In Press).

a set of documents at time step i^{th} as $s_i = \{d_1^i, d_2^i, \dots, d_m^i\}$, where d_j^i is the j^{th} document in s_i , and $m = |s_i|$ is the number of documents in s_i . Let \mathcal{W} be set of all words found in \mathcal{S} . We define the appearance function of a word, w , in a document, d , denoted $app(w, d)$, as a binary function. $app(w, d) = 1$ if d contains w . Otherwise, $app(w, d) = 0$. $p_{[b,e]}$ represents the period from time step b to time step e . $app(w, d) \in p_{[b,e]}$ is the number of document containing w in $p_{[b,e]}$. We denote $occ(w, d)$ as the number of time w appears in document, d . Finally, we denote $tfidf(t, d)$, as the term frequency-inverse document frequency (TFIDF) value of a word w in document d . For each document d , its VSM with TFIDF value is the tuples $\langle w_1, tfidf(w_1, d) \rangle, \dots, \langle w_{|d|}, tfidf(w_{|d|}, d) \rangle$, where $w_i \in \mathcal{W}$. $tfidf(w_i, d) = tf(w_i, d)idf(w_i, d)$, where $tf(w_i, d)$ (TF) equals $\frac{occ(w_i, d)}{\sum_{j=1}^{|d|} occ(w_j, d)}$ and $idf(w_i, d)$ (IDF) equals $\log(\frac{|\mathcal{S}|}{\sum_{d_j \in \mathcal{S}} app(w_i, d_j)})$ VSM with binary value vector representation of d , is the tuples $\langle w_1, app(w_1, d) \rangle, \dots, \langle w_{|d|}, app(w_{|d|}, d) \rangle$.

4.3 Burst Detection

Burst detection is useful in recovering two pieces of important information: the level of burstiness of any given period, and the period in which term t is bursty. While burstiness is a concept defined in Text mining, the measurement of burstiness can be applied to almost all datasets. Similar to significant test, there is no general definition of burstiness. The only limitation of the burstiness is that it can only be applied to a sequence of feature vectors. To sort feature vectors into a sequence, there must exist a feature whose values have the total order relation. The set of feature values, V has a total order relation if V has the following three relations.

1. **Totality Relation:** $\forall a, b \in V \{a \leq b, \text{ or } a \geq b\}$
2. **Antisymmetry:** $\forall a, b \in V \{ \text{if } a \leq b \text{ and } a \geq b, \text{ then } a = b \}$
3. **Transitivity:** $\forall a, b, c \in V \{ \text{if } a \leq b \text{ and } b \leq c, \text{ then } a \leq c \}$

For a dataset \mathcal{S} that has an arbitrary attribute \mathcal{A} , whose values has a total order relation, we give the general definition of burstiness, $B(q, \mathcal{S}, [r, l, A])$, as the measurement of how bursty the data retrieved from \mathcal{S} using q is in period of $[r, l] \in \mathcal{A}$,

where query q , and range $[r, l]$ of \mathcal{A} are given. If we want to explain the burstiness in terms of Database, we can use simple table, and query operations as follow. If the table \mathcal{S} is a specific view on a table in a relational database, given that q is the expression of relational algebra, then the burstiness is the analysis on the temporal table resulted from the given q .

In summary, the burstiness is a measurement of how likely the positive results of q on \mathcal{S} will be found between period $[r, l]$. In [110], the weight of the transition function of a finite state machine is used as the burstiness score. In [62], numerical matrix is used. In Figures 4.1 and 4.2 the burstiness score is defined as the probability of finding the data points with satisfy query in a given period. In both figures, the data used to generate these plots is the automobile data set, taken from the UCI Machine Learning Repository. The horsepower attribute is discretized into 10 periods and used to construct the sequence of data points. From Figure 4.1, the burstiness plots show that most engines have less than 200 horsepower. Only the naturally aspirated engines are made for high horsepower. In Figure 4.2, the burstiness plots show that the most cars with diesel engine have low horsepower. Most diesel engine cars, such as trucks, tend to have more torque at lower engine speeds than petrol engines, because of their commercial usages such as towing.

After the burstiness score is defined, a threshold must be set to indicate the bursty period. In the other words, a query q is considered bursty at step i , if the occurrence of q at the i^{th} is greater than a given threshold. The bursty period is then defined as the period where the query q is bursty over consecutive steps. In previous works, the bursty threshold is set to the mean average over all steps. Hence, the bursty period of a given term is defined as the period where its occurrence in this period is higher than that of an average.

4.4 Temporal Correlated Term Detection

In general, some patterns are only meaningful in certain time period. For example, although terms “Hillary Clinton” and “Secretary of State” have appeared in documents before 2008, only after 2008 that “Secretary of State” has been the job title of “Hillary Clinton”. In September 2009, we used Google Trends to collect

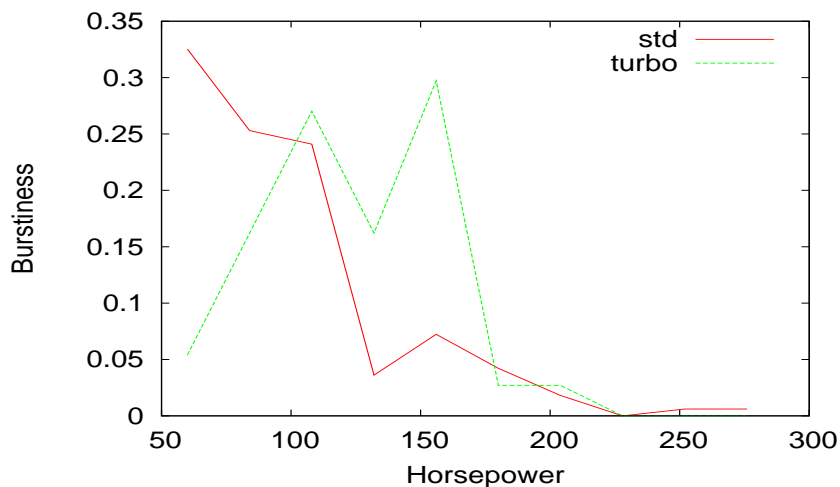


Figure 4.1: The burstiness values of “Aspiration == std” and “Aspiration == turbo” queries on the automobile data set using horse power to arrange, on the x-axis. For y-axis, it is the burstiness, which we defined using probabilities.

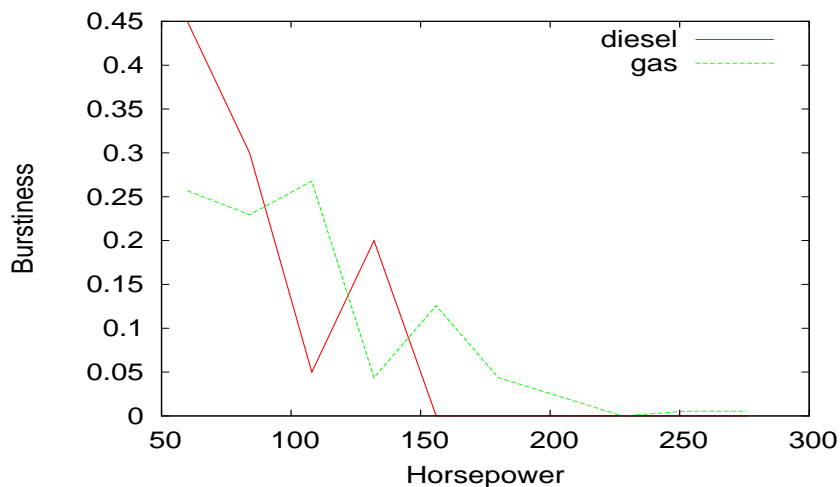


Figure 4.2: The burstiness values of “fuel type == diesel” and “fuel type == gas” queries on automobile data set using horse power to arrange, on the x-axis. For y-axis, it is the burstiness, which we defined using probabilities.

the number of queries submitted by users for each year, from 2000 to 2009. The query that we collected the data for are ‘Hillary Clinton’, “Hillary Clinton” and “Secretary of State” (HC SS), “Hillary Clinton” and “Senator” (HC Senator), and “Hillary Clinton” and “Candidate” (HC Candidate). The collected numbers are shown in Figure 4.3.

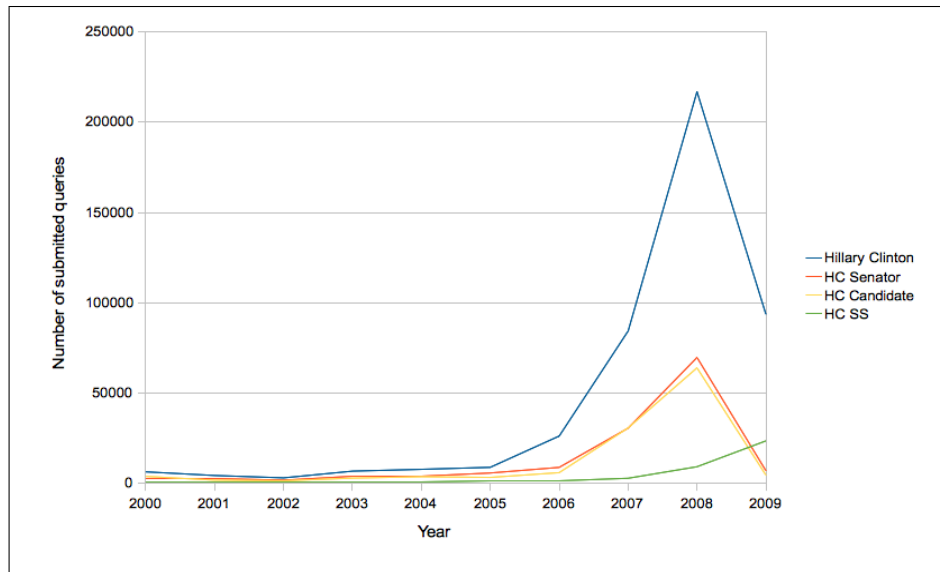


Figure 4.3: Number of search queries input to Google: ‘Hillary Clinton’, “Hillary Clinton” and “Secretary of State” (HC SS), “Hillary Clinton” and “Senator” (HC Senator), and “Hillary Clinton” and “Candidate” (HC Candidate)

In statistics, two terms, q_a and q_b , are independent if the probability of seeing both q_a and q_b in the same document equals the probability of seeing q_a in a document multiply by that of q_b . On the other hand, if they are not independent, we consider them to be correlated. The correlation between two terms is the indicator of how far away from them being independent. There are various coefficient measures used for the degree of correlation between two terms. Example includes Pearson product-moment correlation coefficient [78], and Kendall tau rank correlation coefficient [57]. In English language, words correlations depend on the contexts of documents. For example, “bat” and “cat” are likely to be independent if the document is listed under “baseball”. On the other hand, “bat” and “baseball” should have high correlation coefficient in the same topic. Words correlations also

depend on the time period in which they appear. As seen in Figure 4.3, the query “Hillary Clinton” and “Candidate” are more correlated than “Hillary Clinton” and “Secretary of State” in 2008. However, in 2009, the reverse is true. Using this latter information, we can detect the event surrounding a give query or entity using bursty information.

There is more than one way to extract the temporal correlated terms or patterns using bursty information. We will suggest two ways of extracting such correlated terms. The first natural way is to extract the bursty patterns within the bursty region of a given term. The other is to extract the correlated terms at each time period according to the bursty correlation score defined using bursty information. The latter is difference from the first in a sense that they contain words which are correlated even though they may not be bursty in the same time. We suggested the temporal bursty correlation score in Equation 4.1.

$$Burst_{corr}(q_1, q_2, t) = Burst\left(\frac{Occ(q_1 + q_2, t)}{Occ(q_2, t)}, t\right) \quad (4.1)$$

where q_1 and q_2 are the input queries, t is the interested time period and $Burst$ is the burstiness function. Note that $Burst_{corr}(q_1, q_2, t)$ does not necessary equal $Burst_{corr}(q_2, q_1, t)$. Nevertheless, the temporal bursty correlation score is a good information extraction if the user wants to know what terms are temporally related to a known q_2 .

To illustrate the usefulness of the temporal bursty correlation score, we applied it on the collected data on queries related to “Hillary Clinton” mentioned at the beginning of this section. First, we used Equation 2.3 in Chapter 2 to find burstiness scores of the collected information shown in Figure 4.3. The burstiness scores of these queries are shown in Figure 4.4. Figure 4.3 shows an increase in number of online documents containing the query “Hillary Clinton” since year 2006, corresponding to her reelection campaign of United States Senate election in New York, 2006. The highest peak of the query is in 2008 where Hillary ran her presidential campaign. Figure 4.4 shows the following: (i) “Hillary Clinton” is bursty after 2006, (ii) HC Senator and HC Candidate are bursty between 2007 and 2008, and (iii) HC SS is bursty after 2007. While the burstiness of HC Senator, HC Candidate, and HC SS

are detected, it was only after Clinton’s reelection campaign. Equation 2.3 failed to detect Hillary Campaign and the subsequent victory for the United States Senator election in New York in 2000. The failure to detect such events is caused by the global threshold employed in Equation 2.3. The relatively high number of published online documents and news articles related to H. Clinton – when she enters the race for presidential election 2009 – caused the average number of documents containing each query to be higher than the actual number collected before 2007.

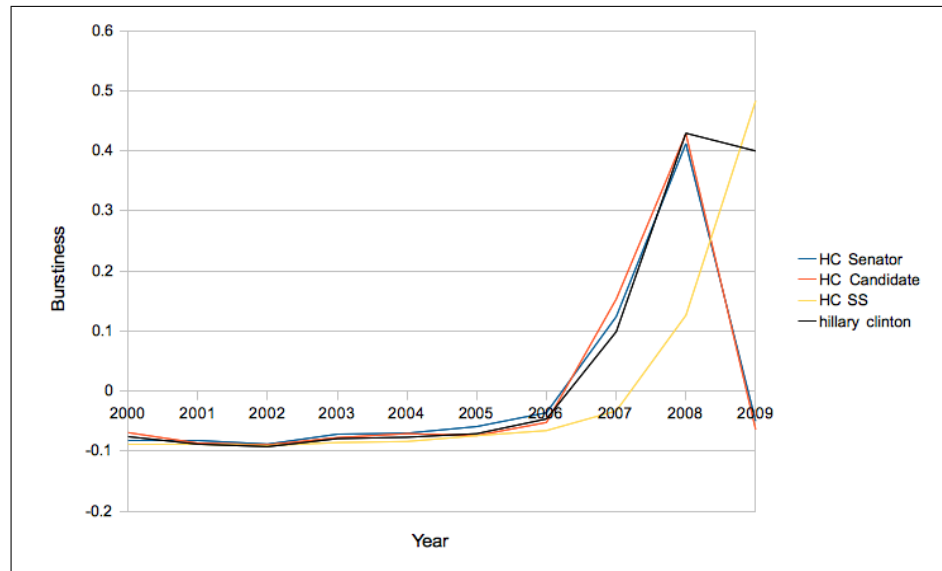


Figure 4.4: Burstiness score, using Equation 2.3, of the queries: ‘Hillary Clinton’, “Hillary Clinton” and “Secretary of State” (HC SS), “Hillary Clinton” and “Senator” (HC Senator), and “Hillary Clinton” and “Candidate” (HC Candidate)

Next, we applied the temporal bursty correlation score to the “Hillary Clinton” example. The results are shown in Figure 4.5. For these results, we used Equation 2.3 as the burstiness function in Equation 4.1. Using the temporal bursty correlation score, we can extract different set of information than those considered bursty in Figure 4.4. We can now match the time in which the correlated queries became bursty with the events that were likely the causes. From Figure 4.5, HC Candidate is bursty in 2000 corresponding to the event of H. Clinton entering United State Senator Election in NY. The burst of HC Senator in 2001 corresponds to the first time that H. Clinton was elected as United State Senator in NY. The burst of HC

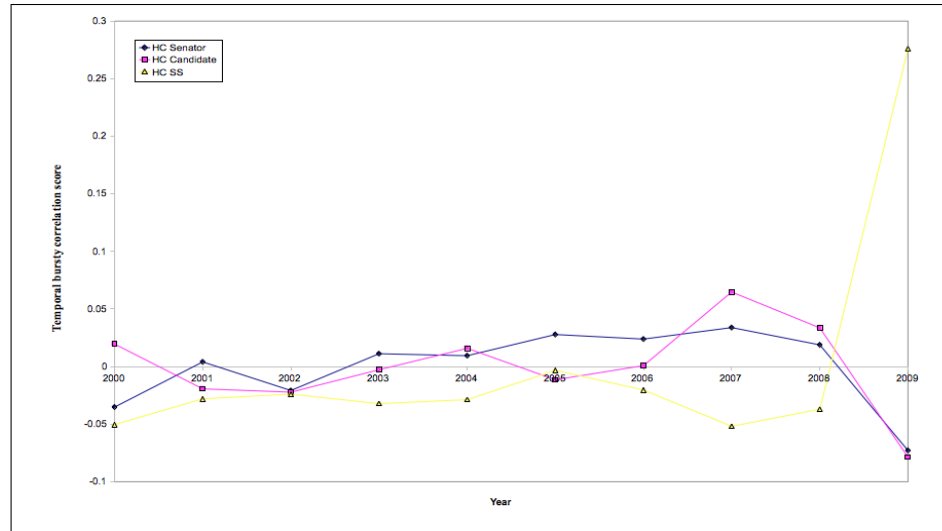


Figure 4.5: Temporal bursty correlate scores of various search query following Equation 4.1

Candidate in 2004 and the increase in burstiness of HC Senator in the following few years correspond to the events of H. Clinton seeking the second US senator term, and the time where she won the second term election respectively. The burst of HC Candidate in 2007 corresponds to the preparing for presidential campaign of H. Clinton. Finally, the burst of HC SS in 2009 corresponds to the event of H. Clinton officially joining the United States Cabinet as Secretary of State. In section 4.4.1, we apply the temporal bursty correlated score to extract interesting correlated terms in ACM dataset.

4.4.1 Temporal Correlated Term on ACM Dataset

In this section, we illustrate how to extract interesting data using burstiness on a real life dataset, ACM Dataset. Specifically, we use burstiness to extract two types of temporal correlated term (keywords in this dataset): (i) the bursty word in the bursty period, and (ii) terms with high temporal bursty correlation score.

We collected the ACM dataset from ACM Digital Library. The ACM Digital Library contains the record of articles published with ACM and its affiliated organizations. We extracted the title, year of publication, publication venue, and author-defined keywords from each article published between year 1990 and 2010. We ignored all articles with incomplete records. The total of 116003 articles were

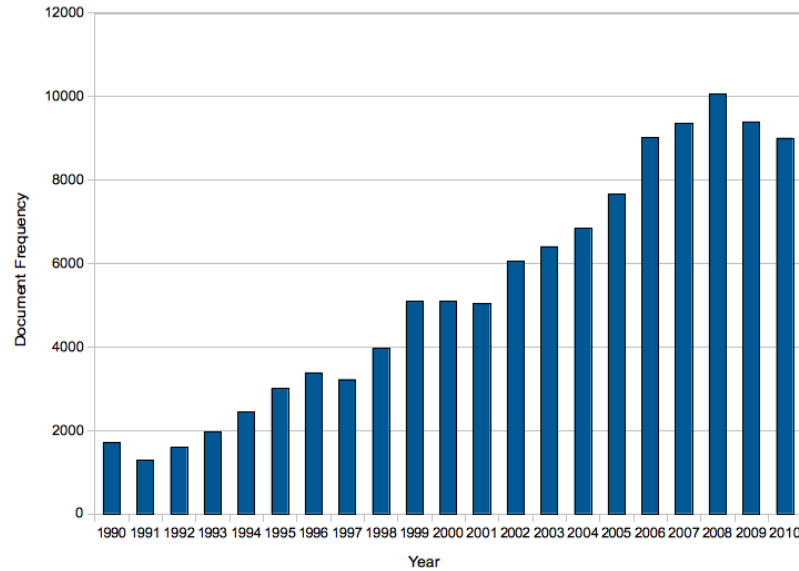


Figure 4.6: The number of records found each year from year 1990 to 2010 on ACM datasets.

retrieved. We created the vector representation of each article using bag-of-words with binary representation. We used the author-defined keywords and the year of publication as the features. Figure 4.4.1 shows the number of records found in ACM datasets in each year.

4.4.1.1 The Bursty Words in the Bursty Period

We performed an analysis of the burstiness, i.e. “given the highly frequent word A , what is its bursty period, and what keywords associated with it are bursty in such period?”. Essentially, the patterns that we want to extract are the correlated terms (A, B) where B is the bursty words in the bursty periods of A . To do that, we need to find the bursty periods of A first. Then, for each bursty period, we find the bursty correlated words to A .

The bursty period was defined as the maximum sum segment – the period whose total burstiness score is greater than zero [62]. We used the burstiness score defined in Equation 2.3 in Chapter 2 to find the bursty score of each word at each time step. Finally, the maximal segments of burstiness scores in the sequence of documents were recovered using the linear-time maximum sum algorithm by Ruzzo and Tompa [92]. We applied the framework on ten research topics with the highest

number of publications, listed in left column of Table 4.1. For each research topic, we displayed its corresponding bursty research topics in the right column of the same table. In the other word, Table 4.1 displays the highly active research topics related to the top research topics at their peaks.

In Table 4.1, “wireless sensor networks” (WSN) is temporally related to “simulation”, “security” and “clustering” in order of bursty period. The order of bursty periods in these three areas corresponded to how WSN research was developed, beginning from simulation of WSN, to the security issues, and finally how clustering algorithms were applied within WSN. The table also reflects the fact that Data Mining research has been broader than that of IR. While Data Mining was used in Computational Science, Web Mining, Time series mining and Security, the bursty topics related to IR are Web related topics. “Text mining” was temporally related to both Information Retrieval and Data Mining.

While we showed only the burstiest period, there could be multiple bursty periods for each keyword. These bursty periods also contain interesting temporal correlated terms. In ACM dataset, there are three bursty periods for “scheduling”: (1900, 1991), (1999, 1999), and (2001, 2006). In 1999, the bursty correlated words, listed in order of the burstiness score, are genetic algorithms, parallel processing, performance evaluation, embedded systems, approximation algorithm, multimedia, quality of service, optimization, and heuristics. From 2001 to 2006, the top 10 bursty correlated words, listed in order of the bursty score, are approximation algorithms, multimedia, online algorithms, real-time, embedded systems, fairness, multiprocessor, quality of service, genetic algorithms, partitioning, load balancing, grid computing, high-level synthesis and wireless networks.

These lists of bursty keywords tell us that in the beginning “real-time systems” and “parallel processing” were research topics related to scheduling. Later on, while the “scheduling” research communities were still interested in “real-time systems” and “parallel processing”, they also showed strong interests in “genetic algorithms”, “embedded systems”, “approximation algorithms”, “performance evaluation”, etc. Before the interests in “scheduling” faded, the communities were interested in “approximation algorithms”, “multimedia”, and “online algorithms”. It shows a com-

elling argument that “genetic algorithms” and “parallel processing” are persistent areas related to “scheduling”. Also, in the last several years of its peaks, the emerging hot research topics related to “scheduling” were multimedia, online algorithm, fairness, multiprocessor, partitioning, load balancing, grid computing, and high-level synthesis wireless networks.

Table 4.1: Top 10 bursty correlated words, listed in order of the busy ranking, in the busiest period of the 10 most frequent words.

Keywords	BP	Top 10 Bursty keywords
security	2000 - 2010	wireless sensor networks, routing, sensor networks, web services, usability, grid computing, wireless networks, peer-to-peer, static analysis, rfid
simulation	1996 - 2010	scheduling, optimization, visualization, wireless sensor networks, sensor networks, qos, wireless networks, ad hoc networks, analysis, validation
data mining	2000 - 2010	genetic algorithms, privacy, bioinformatics, feature selection, time series, web mining, clustering, security, pattern recognition, text mining
scheduling	1990 - 1991	real-time systems, parallel processing
optimization	1992 - 1999	neural networks
neural networks	1992 - 2001	learning, pattern recognition, optimization, fuzzy logic, stability
clustering	2002 - 2010	wireless sensor networks, visualization, data mining, classification, ad hoc networks, genetic algorithms, text mining, neural networks, IR
IR	1999 - 2010	xml, semantic web, ontology, peer-to-peer text mining, information extraction, web search, query expansion, evaluation, search engine
stability	1991 - 1998	robust control, adaptive control, nonlinear systems, robustness, bifurcation
genetic algorithms	1995 - 2009	scheduling, fuzzy logic, heuristics, clustering multi-objective optimization, simulated annealing, neural networks, optimization, data mining

4.4.1.2 Terms with high Temporal Bursty Correlation Score

As discussed previously, temporal bursty correlation score is not the same quantification as document frequency and burstiness. We used it to extract the first time when two bursty terms become correlated. Table 4.2 shows the pair of temporal correlated terms of five most frequent words, as they became bursty for the first time. We presented the results only the correlated terms, with in the first four-bursty years, of five most frequent keywords.

The results from previous section showed that the bursty periods of “scheduling” did not include 1992 and 1993. During which periods, in Table 4.2, research topics such as multimedia, dynamic programming, resource allocation, heuristic, on-line algorithms and multiprocessor had been temporally correlated to scheduling for the first time. The results in Table 4.2 also indicated that the “security” research community were interested in topics such as access control, encryption, integrity, routing, modeling, verification, and authentication in early 1900s. Later on, the communities were also interested in wireless sensor networks, sensors networks, and web services.

As we can see from the results of this experiment, the second framework provided us with a complementary view to the previous framework, i.e. while the previous framework focuses on the bursty regions, this framework looks at the boarder view of the text streams. For a concrete example, we can look at research in security. We learned from the previous experiment that security has been a bursty topic in the last decade and that its research during this time focused on networks. In this experiment, we also learned that research in security began from information integrity.

4.5 Burst Detection for Document Clustering

In context of text mining, if we consider a news report, when an event occurs, the news regarding the event would be written over a period of time following the event. The length of the active period of the news and the number of articles in

Table 4.2: The temporal correlated term in the first four-bursty years of five most frequent keywords.

Keyword	Year	Temporal Correlated Word
security	1991	access control, encryption, integrity, routing, modeling, verification
	1992	trust, management, safety, policy
	1994	distributed systems, authentication,
	1995	reliability, mobility, intrusion detection, anonymity, authorization
simulation	1990	performance evaluation, performance
	1991	routing, modeling, verification, performance analysis
	1992	design
	1995	education, modeling
data mining	1995	visualization
	1996	neural networks, classification, association rules, knowledge discovery, decision tree, simulation
	1997	information retrieval, machine learning, data warehouse
	1998	pattern recognition, text mining, cluster analysis
scheduling	1990	performance evaluation, parallel processing, real-time
	1991	computational complexity, load balancing, partitioning
	1992	multimedia, dynamic programming, resource allocation, heuristic
	1993	online algorithms, multiprocessor
optimization	1991	routing
	1992	neural networks, simulated annealing, compiler
	1994	genetic algorithms
	1995	performance, heuristics

the media related to the event depends on the popularity of the story. If we plot the graph of the number of news articles related to the event on a timeline, it is likely to increase in the beginning of the active period of the event and to reduce afterward. News articles are often grouped into major categories such as, business, entertainment, and politics. Several websites, such as CNN and Yahoo News, also provide related keywords for each article. The query composed of such keywords is likely to be bursty in this period. We used this concept to create our “bursty distance measurement” framework for document clustering.

A word can be bursty in two different time periods. Even though the bursty score between two time periods are similar, they can correspond to two entirely separate events. For example, in Entertainment news in 2009, the word “death” is very bursty in March, June and July. In March, the burstiness was attributed to the high number of news articles regarding the death of actress Natasha Richardson on March 16th, 2009. In June and July, the burstiness of the term was attributed largely to the high number of news articles regarding the death of Michael Jackson on June 25, 2009. While both sets of news articles were related to the death of celebrities, they corresponded to two separate events. The current bursty feature representation cannot distinguish between the two events.

All the mentioned methods identify the bursty periods using a hard global threshold, such as the mean frequency or the overall emission rate of the document. Specifically, a word w is considered bursty at time step i , where $i = 1 \dots n$, if the occurrence of w at time i^{th} is greater than its average occurrence. Since the burst period is defined as the period where w is bursty over consecutive time steps, the bursty period represents the period when its occurrence in this period is higher than the average. Our framework considers that the information from the nearby time period as more important than those from other time periods further away. Our framework also emphasizes the important of bursty period information more than the previous work.

In general, there are several ways to group document together. Online news articles are often grouped into major categories such as, business, entertainment, and politics. They can also be assigned the predetermined keywords related to each article. In the latter case, when an event occurs, the news regarding the event would be given the same tag. Such news articles are written over a period of time following the event. The length of the active period of the news and the number of articles in the media related to the event depends on the popularity of the story. If we plot the graph of the number of news articles related to an event on a timeline, it is likely to increase in the beginning of the active period of the event and to decay afterward.

The active period of the event often corresponds to the bursty period of the keywords related to the event. For example, the occurrence plot of documents

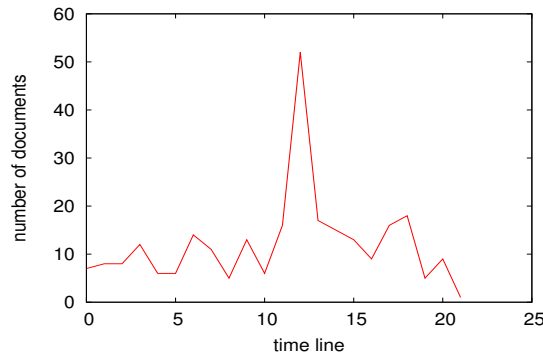


Figure 4.7: Numbers of articles in San Francisco Call newspaper containing words “victoria” and “death” per month from Jan. 1900 to Dec. 1901.

containing keywords “victoria” and “death” in the San Francisco Call newspaper between January, 1900 and December, 1901 is shown in Figure 4.7. The highest peak between January 1901 and February 1901 – correspond to the death of Queen Victoria in late January of 1901. As discussed in Chapter 2, the documents are clustered according to the similarity scores between the documents. From this example, if we increase the similarity score of the document containing these two words in the highest peak, the documents related to this event would likely be clustered to the same group. In the other words, if two documents contain the bursty terms in their bursty period, then they are likely to belong in the same cluster. Hence, we can use the burst detection task to help clustering by finding the period that a word w is bursty, and its level of burstiness in that period.

4.5.1 Bursty Features for Document Clustering

In [42], He et al. used bursty information to create the bursty features for document clustering. The bursty feature representation for a document $d = w_1, w_2, \dots, w_n$ at time t_d is defined as follow.

$$w_i = \begin{cases} FP_i + \beta b_i & \text{if } w_i \text{ is bursty at time } t_d \\ FP_i & \text{otherwise} \end{cases} \quad (4.2)$$

where FP_i is the static feature weight such as binary weighting, b_i is the bursty score and $\beta > 0$ is the burst coefficient. There are two existing burst detection methods

that can be applied to Equation 4.2: Kleinberg’s algorithm introduced in [60], and Discrepancy model of Burstiness [62]. Both of them defined the bursty period of a word w , as the period during which the documents containing w appears more frequently than on average.

4.5.1.1 Kleinberg’s Algorithm:

Kleinberg proposed to identify the bursty period of t using finite-state automaton [60]. The Kleinberg’s two-state finite automaton, \mathcal{A} , has q_0 and q_1 states. They represent the normal period and the bursty period, respectively. \mathcal{A} is in q_1 state at time a , if the emission rate is $\frac{s \cdot |D_w|}{n}$, where $|D_w|$ is the number of documents containing w , n is the total number of time steps in S , and $s > 1$. \mathcal{A} is in q_0 state otherwise. a is a bursty period if \mathcal{A} is in q_1 state at time a . The cost of being in state q_i of a word w at time i is defined in Equation 4.3.

$$\sigma(i, app(w, s_i), |s_i|) = -\ln \left[\binom{|s_i|}{app(w, s_i)} p_i^{|s_i|} (1 - p_i)^{|s_i| - app(w, s_i)} \right] \quad (4.3)$$

where $app(w, s_i)$ is the number of documents in s_i that contain w at time i , $p_0 = \frac{app(w, s_i)}{|S|}$ and $p_1 = s \cdot p_0$. The bursty score of t in period $p_{[b,e]}$ is the cost increment if \mathcal{A} is in q_1 rather than q_0 as defined in Equation 4.4. He et al. adopted this two-state finite automaton and used it to create the bursty feature representation described in Section 4.5.1 [42].

$$Burst(w, p_{[b,e]}) = \sum_{i=w_1}^{t_2} (\sigma(0, app(w, s_i), |s_i|) \sigma(1, app(w, s_i), |s_i|)) \quad (4.4)$$

4.5.1.2 Discrepancy Model of Burstiness (Max-1):

In [62], the bursty period of word w is detected using the discrepancy model and the maximal segment problem. First, the discrepancy model identifies the time step in which the document appears more frequently than on average. Specifically, for a given word w , the burstiness score of each time step is defined by the following Equation 4.5.

$$Burst_{Max1}(w, i) = \frac{app(w, s_i)}{app(w, S)} - \frac{1}{n} \quad (4.5)$$

where $app(w, S)$ is the number of documents in the whole sequence that contain w . If $Burst(w, i) > 0$, then w is bursty at time i . Next, the maximal segments of burstiness scores in the sequence of documents are recovered using the linear-time maximum sum algorithm by Ruzzo and Tompa [92]. The bursty period is defined as the recovered maximum sum period that is strictly positive. Finally, the burstiness score of each maximum sum period is defined as

$$Burst_{Max1}(w, p_{[b,e]}) = \frac{app(w, p_{[b,e]})}{app(w, S)} - \frac{|p_{[b,e]}|}{n} \quad (4.6)$$

4.5.2 Bursty Distance Measurement

Although, the bursty feature representation is robust and shows improvement when applied to classical VSM, there are two distinct weaknesses in the bursty feature representation. First, according to [42], the same improvement cannot be seen as clear on VSM with TFIDF value. The cause of marginal different on VSM with the TFIDF value was the fact that TFIDF value of a rare word has high TF and IDF scores. Based on how often words appear in the corpus, words are partitioned into three separate categories: common, average, and rare words. The common words are words that appear commonly throughout the corpus such as “the” and “do”. These common words are often considered irrelevant and dropped from the vector space. The rare words are words that appear in a small group of document such as “vampire” and “spaceship”. Other words are considered average words. Rare words are bursty simply because they appear in a small time span in the corpus. Noisy words are also part of rare words. By adding the bursty weight to the TFIDF value of the rare word, we also amplify the noise. On the other hand, average words appear in larger group of documents. So on average they will have lower IDF score. In order to put more emphasize on the bursty information, their TFIDF values needed to be boosted more. Since some rare features with low DF are already emphasized with high IDF value, the bursty feature effect on rare features on VSM with TFIDF value would not be as strong as that of VSM with binary value.

Second, the bursty models identify a period of word w as bursty if either there is an increase in volume of documents containing w or the burstiness score of w rises

above the global average threshold. These definitions of bursty period will not work well in event detection because a small event may disappear in as noise. For example, using SIGHT software, Baumes et al. recovered that prior to the declaration of independence of Kosovo in 2008, there was a small group of bloggers discussing the issue [8]. Although, the number of blogs was small, the discussion group was persistent up until the actual week of declaration of independence. If we used the mean as the threshold value to find the burstiness of “independence Kosovo”, such a discussion would never be considered due to high volumes of discussion of the event in the following weeks.

To address these issues, we introduce a Bursty Distance measurement (BDM). For the first issue, given a word w , it is likely that the sharp increment in the number of occurrences of w at time b indicates the beginning of the event related to w . Also, the decay of the number of occurrences of w at time e indicates the ending of the event related to w . We make an assumption following these observations that the bursty interval of the word indicates an active life span of the news of an event. BDM considers documents d_1 and d_2 , on two different bursty intervals of a word w as time independent documents on word w . For time independent document on word w , BDM does not add the bursty weight of w to VSM value when distance between d_1 and d_2 is measured. For the second issue, we defined that a word w is bursty at t if the actual occurrence is higher than its probability density estimation. By looking at the probability density estimation, we allow detection of the small event such as those described above. Such event would give a higher occurrence than the smooth estimated by the kernel density estimation. BDM assigns the bursty score of a word w at time i as follow

$$Burst_{kde}(w, i) = \frac{app(w, s_i)}{app(t, S)} - f_h(i) \quad (4.7)$$

where $f_h(i)$ is the kernel density estimated at time i . We used the Gaussian function as the kernel function for this work. h can be found using the fast bandwidth estimation in [90]. After finding the bursty score, we defined the burst period as the maximal segment of the burstiness similar to that of Max-1. The burstiness of word w on the bursty period $p_{[b,e]}$ is the average sum of $Burst_{kde}(w, i)$ for $i \in p_{[b,e]}$

as defined by Equation 4.8.

$$BBDM(w, p_{[b,e]}) = \frac{\sum_{j=b}^e Burst_{kde}(w, j)}{e - b + 1} \quad (4.8)$$

We define $B(t_1, t_2, w)$ as a bursty weight function where $B(t_1, t_2, w) = BBDM(w, p_{[b,e]})$ if times t_1 and t_2 are in the same maximal segment $p_{[b,e]}$. Otherwise, $B(t_1, t_2, w) = 0$.

The complete BDM algorithm is described below. We define d_1 and d_2 to be two documents at time t_{d_1} and t_{d_2} respectively. The distance between d_1 and d_2 is defined using algorithm in the following algorithm, where $DIST$ is the provided distance measurement between two vectors, ρ is the upper bound frequency for rare words. BDM uses this bound to control the noise amplification of rare words.

Bursty Distance Measurement

Require: d_1, d_2, β, ρ , and $DIST()$

Ensure: D_{d_1, d_2}

for $i = 1$ **to** m

if $B(t_{d_1}, t_{d_2}, w) > 0$

$$\theta = \begin{cases} \beta^{-1} & \text{if } app(w_i) < \rho \\ 1 & \text{otherwise} \end{cases}$$

$$v_i^1 = tfidf(w_i^1) + \theta \cdot B(t_{d_1}, t_{d_2}, w_i)$$

$$v_i^2 = tfidf(w_i^2) + \theta \cdot B(t_{d_1}, t_{d_2}, w_i)$$

endif

endfor

$$D_{d_1, d_2} = DIST(\{v_1^1, \dots, v_n^1\}, \{v_1^2, \dots, v_n^2\})$$

4.5.3 Experiments

We evaluated our model on three sets of experiments based on three groups of datasets: synthetic dataset experiment, news article dataset experiment and additional real-life datasets. We performed experiments on synthetic and new article datasets to illustrate that if there was a time-dependent storyline, the clusters recovered using the bursty distance measure tended to cluster documents from the same

storyline together. On additional real-life datasets, where time-dependent storyline were not apparent, our experiments showed that our framework yielded similar qualities as the other methods.

For the experiments, we tested all three bursty period and bursty score analysis methods discussed in Section 4.3. We used cosine distance as the provided distance function in the framework. In addition to the bursty distance measurement (BDM), we applied the cosine distance to the bursty feature representation for document clustering model discussed in section 4.5.1. We used two bursty models for the bursty feature: the discrepancy model (Max1), and Kleinberg’s algorithm (KL). For a baseline comparison, we also clustered documents using the normal VSM with TFIDF representation (TFIDF). We used three clustering methods, K-Medoids, hierarchical (UPGMA) and lower bound on similarity clustering (Star). In K-Medoids, the interpretation of its results was not sound as a center did not necessary reflect the time period in the cluster that it represented. Hence, we only used the K-Medoids in the synthetic dataset experiment to show that BDM was robust enough to be applied in K-Medoids. We implemented the K-Medoids algorithm. For UPGMA, we used the built-in code in Matlab. We chose Star algorithm for our experiment to evaluate BDM performance on clustering methods when true number of clusters was not known. The code for Star algorithm was acquired from clustering with lower bound algorithm package in [41].

Since the input of Star algorithm required the similarity score, we calculate the similarity score by subtracting the cosine distance score from one. Clustering using lower bound on similarity does not always provide the target number of clusters. We applied the following steps to find the clustering with the right number of clusters, k : (i) find the similarity threshold that gives the smallest upper bound k_{τ} to the target number of clusters. (For these experiments, the precision of the threshold is in the order of 10^{-3} .) (ii) if $k_{\tau} = k$, terminate. Otherwise, find the smallest cluster and merge it with its closest cluster using average similarity. (ii) Repeat step 2–3 until the number of remaining clusters is reduced to k .

4.5.3.1 Synthetic Dataset

In the first experiment, four synthetic datasets were generated using the generative probabilistic model described in [102]. The first two synthetic datasets, Syn-1 and Syn-2, were generated using the generative models where each document contained two time dependent topics: the major topic and the minor topic. By time dependent topics, we imply that if a document contains the major topic, then it can only contain a unique set of minor topics. Each major topic has five unique minor topics. Each topic has its active time span such that the topic can only appear if the document was generated in its active time span. Each topic is represented by 5 distinct keywords. There are a total of 20 major topics and 100 minor topics. Each document is 40-word long. For Syn-1, 20 words were drawn from the topic keywords (10 words each for the major and minor topic keywords). For Syn-2, 6 words were drawn from the topic keywords (3 words per each topic). The remaining words were randomly drawn from the vocabulary of 40000 words. Syn-3 was generated using the same model as Syn-1, but each topic was time independent. Syn-4 is the combination of Syn-3 and Syn-2, where each document contains two time dependent topics and two time independent topics. Each document in Syn-4 is 40 words long with 3 words drawn from each of the topics in the document; the rest of the words were randomly drawn from the vocabulary of 2000 words. Each data set contains 10000 documents. The precision, recall and F-score were used to evaluate the clusters.

The results are shown in four consecutive tables from Table 4.3 to Table 4.6. For UPGMA, the BDM and Max1 performed consistently better than TFIDF. BDM performed better than Max1 and KL in most cases. For K-Medoids, the methods performed equally well in general. K-Medoids selected one document to represent each cluster. The selected document's bursty features may not represent the whole cluster, reducing the effectiveness of bursty information. Star method produced poor clustering across all methods. The results from Star method indicate that documents from two clusters are closer together than their corresponding centers. However, BDM was the only burstiness framework that showed improvement over TFIDF on all three clustering methods. Such results indicate that BDM is more robust than

Max-1 and KL. The collective results show that bursty information can help with clustering if documents in the same group are time dependent. Using a bursty period to indicate distinct storyline, BDM is able to cluster time dependent documents better than those of classical VSM (TFIDF) and bursty feature representations (Max1 and KL).

Table 4.3: Synthetic Dataset Syn1

Algorithm	Vec. Rep	Major topic			Minor topic		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	100	100	100	71	79.7	64
	Max1	100	100	100	71.1	79.8	64
	KL	99.97	99.97	99.97	75.39	88.57	65.63
	BDM	100	100	100	77.36	86.26	70.12
KMedoid	TFIDF	100	100	100	73.49	77.35	70
	Max1	100	100	100	73.36	77.05	70
	KL	100	100	100	73.24	77.64	69.31
	BDM	100	100	100	73.49	77.35	70
Star	TFIDF	8.46	90	4.44	1.97	92.8	0.99
	Max1	6.36	75.52	3.32	4.19	85.9	2.15
	KL	8.88	95	4.66	2.7	100	1.37
	BDM	8.88	95	4.66	2.16	97.86	1.09

4.5.3.2 News Article Data sets

In this set of experiments, our framework was applied to two datasets: Entertainment and Politics datasets. Both datasets are news headlines collected by [8] from the following RSS feeds: www.cnn.com/services/rss/ in 2009, news.yahoo.com between 2008 – 2009, and today.reuters.com between 2007 – 2008. The data was collected by simply checking feed every hour for new messages. For entertainment news, we extracted the total of 9087 news headlines from 03/01/2009 to 08/31/2009.

Table 4.4: Synthetic Dataset Syn2

Algorithm	Vec. Rep	Major topic			Minor topic		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	10.41	5.49	99.42	6.98	3.62	95.41
	Max1	10.42	5.5	99.44	10.67	5.66	93.46
	KL	10.62	5.61	99.18	5.86	3.02	97.16
	BDM	10.46	5.52	99.4	8.7	4.56	94.44
KMedoid	TFIDF	18.22	40.82	11.73	18.74	30.61	13.5
	Max1	24.69	45.31	16.97	22.35	33.64	16.73
	KL	16.05	41.8	9.93	14.99	33.45	9.66
	BDM	24.79	40.83	17.8	26.5	37	20.64
Star	TFIDF	9.72	100	5.11	2.23	98.84	1.13
	Max1	7.12	82.93	3.72	2.24	92.66	1.13
	KL	8.52	92.1	4.47	2.27	99.99	1.15
	BDM	9.93	100	5.22	2.68	99.09	1.36

Table 4.5: Synthetic Dataset Syn3

Algorithm	Vec. Rep	Major topic			Minor topic		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	96.41	96.41	96.41	34.15	91.35	21
	Max1	99.96	99.96	99.96	32.63	68.66	21.4
	KL	86.44	86.11	87.78	32.33	79.35	20.3
	BDM	99.96	99.96	99.96	39.24	85.87	25.43
KMedoid	TFIDF	48.93	49.62	48.26	42.5	44.44	40.73
	Max1	58.94	60.99	57.02	48.33	51.13	45.82
	KL	33.5	33.14	33.87	21.4	22.43	20.46
	BDM	60.72	61.97	59.52	48.94	52.03	46.2
Star	TFIDF	8.46	93.01	4.43	2.20	97.39	1.11
	Max1	7.81	89.27	4.08	1.98	93.39	1
	KL	7.8	88.74	4.08	2.38	98.48	1.21
	BDM	7.88	88.36	4.12	2.51	100	1.27

For politics news, we extracted the total of 15585 headlines from 10/01/2008 to 12/31/2008. Although the sources of the news were different, we considered them as one single sequence of documents.

We tagged part of the documents related to four particular events "Obama won 2008 presidential election" (Obama), "Clinton, H. is named the secretary of state" (Clinton), "the death of Natasha Richardson" (NR), and "the death of Michael Jackson" (MJ). We used the Star algorithm to cluster documents into roughly 1700

Table 4.6: Synthetic Dataset Syn4

Algorithm	Vec. Rep	Major topic			Minor topic		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	68.75	69.47	68.04	18.57	20.27	17.14
	Max1	79.27	79.88	78.68	21	23.73	18.83
	KL	71.47	71.33	71.60	27.93	57.67	18.43
	BDM	79.61	80.49	78.75	26.54	52.75	17.73
KMedoid	TFIDF	57.15	58.9	55.51	35.67	39.16	32.75
	Max1	46.07	46.89	45.28	35.86	37.19	34.62
	KL	46.86	47.27	46.48	33.51	35.5	31.73
	BDM	57.41	58.99	55.91	35.54	36.6	34.54
Star	TFIDF	10.17	100	5.36	2.83	95.13	1.44
	Max1	9.98	100	5.25	2.18	94.5	1.1
	KL	10	100	5.26	2.53	94.46	1.28
	BDM	7.09	81.12	3.71	3.6	99.84	1.83

clusters for entertainment news and roughly 4000 clusters for political news. Our goal was to cluster the data so that each cluster had about 5 to 6 news articles on average. A good clustering should be able to cluster the similar news together, while filtering the non-related news out. Then, the pair-tagged-document counts for each event were collected as follow. Given that document A and B are tagged as Obama, the pair will produce the count of one, if A and B appear in the same cluster. Otherwise, no count will be recorded. The recall was calculated from these counts as the percentage of recovered pair-tagged-document. We collected the number of clusters containing the documents, and used it to compute the precision as the number of tagged pairing over the number of total pairing recovered from the clusters containing tagged documents. Note that, if there exists one large cluster with every document, this setting will get the highest recall score. However, the precision for such a clustering will be low. For a good clustering method, we expect the recall and precision to be high, while the number of clusters containing tagged documents to be low.

The results of the experiment are shown in Table 4.7 in which “Num. Doc.” refers to the number of tagged documents. “Cluster” refers to the number of clusters containing tagged document. For all four events, BDM achieved the highest precision score and the lowest number of clusters. With the exception of Obama

event, BDM had the highest recall score. TFIDF performed better than Max1 and KL on all events except Obama. In Obama event, Max1 had the highest recall score. However, this is due to a single cluster containing many documents with the words “obama” and “barack”. As such, Max1 had a low precision for that event. For other events, Max1 and KL cluster related reports together instead of focusing on specific events. For example, the news articles on the wake for both Natasha Richardson and Michael Jackson are clustered together by Max1. Since bursty feature representations (Max1 and KL) consider all bursty periods as related to certain degrees, they cannot distinct the given example as well as BDM. Such results indicate that not only BDM was able to filter out documents, which did not belong to the same event, from the clusters better than other methods, but it could also cluster documents from the same event together in a tighter group of clusters.

Table 4.7: RSS feed dataset (Entertainment news 03/01/2009 - 08/31/2009 and Political news 10/01/2008 - 12/31/2008)

Events	Num Doc.	Vec. Rep.	cluster	recall	precision
NR	44	TFIDF	9	20.71	30
		Max1	8	17.44	17
		KL	7	16.43	29.28
		BDM	8	28	31
MJ	48	TFIDF	6	62.94	2.5
		Max1	13	13.92	2.1
		KL	13	61.63	2.4
		BDM	6	69.68	3.1
Obama	61	TFIDF	21	12.24	0.02
		Max1	15	27.54	0.55
		KL	35	4	0.04
		BDM	12	23.22	1.3
Clinton	40	TFIDF	4	51.67	0.05
		Max1	10	15.64	0.05
		KL	22	6.5	0.001
		BDM	3	61.67	2.9

4.5.3.3 Additional Real-life Datasets

We used four other real-life datasets: 20 newsgroup, Reuters-21578, Nature and ACM datasets in the following experiments. The purpose of these experiments is to show that BDM is a robust framework that can improve clustering results on the same data even though it may have different feature space. For each dataset, we used the rainbow package [72] to prepare two indexes for the experiments. We generated two sets of indexing based on two feature selection methodologies: information gain and frequency. For information gain, we used the rainbow package to find 100 highest words in terms of information gain statistics. For frequency, we used the rainbow package to create the index of all words except the top and bottom 10%. We ignored any document that does not contain any of the features. The precision, recall and F-score are used to evaluate the clusters.

20 newsgroup dataset: The dataset is available on the UCI Machine Learning Repository (see [34]). The dataset is a popular test bed for text clustering experiments. We automatically removed the headers and the reply/forward portions from each document. For multiple label documents, we selected the first listed label as the actual label. The results of the experiment on 20 newsgroup dataset are shown in Table 4.8.

Reuters-21578 dataset: The Reuters-21578 dataset is made available on the UCI Machine Learning Repository (see [34]). We extracted the body from each document and labeled each document with the first listed topic. We dropped any documents that did not contain a topic. We also dropped any topic that contained less than 5 documents.

Nature dataset: Nature dataset is the collection of Nature articles, published weekly and organized into five distinct classes. In this collection, there are 7,964 journal articles published between 01/01/2005 and 12/21/2006. The final dataset contains 7437 journal articles. The results of the experiments with cosine distance are shown in Table 4.10.

ACM dataset: ACM dataset contains 24897 Computer Science articles from the ACM digital library divided into 11 classes, published yearly from 1980 to 2001. The results are shown in Table 4.11.

The results from Tables 4.8 to 4.11 indicated that BDM performed much better than KL, Max1 and TFIDF. From the total of 16 experiments, BDM had the best F-score on 9 experiments. TFIDF has the best F-score on 5 experiments. KL and Max1 has the best F-score on 1 experiment. Overall, BDM was better than Max1 in 13 experiments on F-score. BDM was better than KL in 12 experiments on F-score. On average, BDM showed 2% improvement over TFIDF on F-score. Again, BDM showed remarkable robustness as it was able to improve clustering on two types of features. Moreover, on 20 Newsgroup and Reuters-21578 datasets, where the text streams were likely to contain time dependent topics, frameworks that used bursty information for document clustering out performed TFIDF in general. Similar to Syn-3 dataset, KL and Max1 performed worse than BDM and TFIDF on Nature and ACM datasets, since these two datasets were not likely to have as much time dependent topics as the other two datasets.

Table 4.8: 20 Newsgroup Dataset

Algorithm	Vec. Rep	Info. Gain			Frequency		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	44.57	52.14	38.92	11.95	89.46	6.4
	Max1	48.26	54.93	43.03	11.27	92.99	6
	KL	49.91	62.02	41.75	11.44	89.25	6.11
	BDM	50.46	69.96	39.46	10.31	98.56	5.44
Star	TFIDF	31.85	29.28	34.91	5.03	4.61	5.54
	Max1	37.55	31.61	46.25	5.77	5.33	6.3
	KL	32.24	28.6	36.94	7.95	4.39	41.75
	BDM	36.34	30.15	45.73	10.9	5.88	74.88

Table 4.9: Reuters Dataset

Algorithm	Vec. Rep	Info. Gain			Frequency		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	52.34	63.3	44.61	47.24	64.27	37.35
	Max1	52.59	62.79	45.24	49.32	63.02	40.51
	KL	58.22	68.67	50.53	46.07	64.87	35.71
	BDM	54.62	64	47.63	49.99	61.04	42.33
Star	TFIDF	29.23	63.67	18.97	35.56	62.74	24.81
	Max1	29.66	58.91	19.82	29.71	61	19.64
	KL	26.6	53.47	17.7	35.08	64.14	24.14
	BDM	34.47	63.15	23.71	34.6	57.49	24.75

Table 4.10: Nature Dataset

Algorithm	Vec. Rep	Info. Gain			Frequency		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	86.11	99.78	75.74	86.95	99.51	77.21
	Max1	81.91	89.26	75.69	86.31	97.92	77.16
	KL	85.61	98.5	75.7	85.82	96.66	77.16
	BDM	84.34	95.21	75.7	87.03	99.80	77.16
Star	TFIDF	64.02	69.54	59.3	84.54	93.48	77.16
	Max1	61.17	66.29	56.79	83.39	90.71	77.16
	KL	56.65	68.16	48.46	74.83	72.64	77.16
	BDM	61.01	69.71	54.24	85.77	96.55	77.16

4.6 Conclusion

In this chapter, we illustrated how we can use bursty information to extract interesting information from the data. Also, we defined the temporal bursty correlation score that can assign high correlation score to two patterns even though one or both of the patterns are not bursty. This feature allows for the emerging correlated pattern to be extracted. We used it to recover early focus of research topics in Computer Science research from ACM datasets. The concepts of burstiness are not limited to only such information extractions. They can be used to improve the performances on other text mining tasks. We introduced the bursty distance measurement framework that used both the bursty period and burstiness score of each feature as the weight for distance calculation between two documents. We showed that our framework excelled in clustering documents related to the same event both on Synthetic dataset and RSS feed dataset. It performed better than

Table 4.11: ACM Dataset

Algorithm	Vec. Rep	Info. Gain			Frequency		
		F-score	Precision	Recall	F-score	Precision	Recall
UPGMA	TFIDF	43.61	30	79.79	41.29	27.28	84.87
	Max1	42.52	28.47	83.97	41.1	27.23	83.73
	KL	42.38	27.16	96.4	41.46	27.51	84.09
	BDM	48.47	36.24	73.16	42.37	27.17	96.14
Star	TFIDF	20.1	24.44	17.07	2.17	5.07	1.38
	Max1	18.67	25.55	14.7	1.65	7.19	0.93
	KL	16.8	18.25	15.57	0.52	3	0.29
	BDM	19.75	24.19	16.68	8.52	9.4	7.79

simply using VSM with existing bursty feature representation in such tasks. Our framework’s performance did not suffer when applied to other real world datasets. We used the local burstiness score that focuses on the local word occurrences in our framework. The local burstiness score leads to better event-related clustering on RSS feed dataset. These results suggest that our framework is likely to work well on event-related clustering on streaming text such as RSS feed, and Blogs. Also, we found that K-Medoids clustering algorithm is not compatible with clustering with bursty information in general.

CHAPTER 5

Conclusions and Future work

The current growth rate of online information is at an all-time high, and it shows no sign of slowing down. Pattern extraction is an interesting and important concept that has been used to extract information from electronic documents. In this thesis, we look at the problems that arise when classical word feature and sequential pattern feature such as N-gram are applied to mining online text documents. The word feature cannot capture the information embedded in the order of words. The traditional sequential patterns extracting methods are not scalable enough to extract long approximate patterns. They have difficulties extracting information from online text document, which can be a long and noisy sequence of words. The combinatorial search space for such data is simply too large to extract all sequential patterns. In Chapter 3, we show that we need sequential patterns mining in order to solve hard text mining tasks such as role identification. We also show that the selected space search is necessary for locating sequential patterns in such a large combinatorial search space problem as text mining.

The other problem for traditional word feature and sequential pattern feature base methods have is that they cannot capture the temporal information embedded in the online text streams. In Chapter 4, we propose two frameworks that use the bursty information for extracting patterns in the text streams. We show that, if we include the bursty information into the feature values, we can extract interesting temporal information from the data – leading to a better learning model and more informative patterns. There are two interesting directions of future works: burstiness for Recursive Data Mining and Surface Text Patterns. We discuss them next in section 5.1 and 5.2 respectively.

Portions of this chapter previously appeared as: A. Hoonlor, and X. Zhu. (2006, Apr.). Unsupervised learning for surface text recovery in QA system, Computer Science Department, University of Wisconsin - Madison, WI. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.4787>, (Date Last Accessed on Jun. 2, 2011)

5.1 Burstiness for Recursive Data Mining

The first idea we want to explore is to add the temporal sense into the sequential pattern. If we look at today newspaper, the news article in the newspaper often mention just current event. The current events include periodical events such as presidential election, or an on-going event such as Iraq war, or commonly occurring events like a death of celebrity. Thus, some phrases used in the current week or month of newspaper to describe the current events can be found in other times. It would also be interesting to see if the structures of the document change over time.

Burstiness is the concept that describes such property. If we can define a burstiness score for sequential pattern, we can apply it as the significant test for RDM. We will also need to improve RDM framework by making it more efficient. While the current system can handle long and noisy sequence, it still has limitations on the level of approximation allowed in the patterns at each level of abstraction. Also, by adding the temporal information into the test, RDM has to keep track of the dimension of time in the dataset – immensely increasing the size of the search space.

5.2 Surface Text Patterns

One of our goals is to incorporate more information into sequential patterns. The surface text pattern is an intuitive way to include the knowledge base into the pattern extraction algorithm allowing it to reduce the size of search space. In Chapter 1, we give a brief introduction to the surface text pattern and its uses in the question answering (Q&A) system. The quality of an answer to the Q&A system depends on the information extracted from the data. The surface text pattern can filtered out most of the irrelevant data and extract the answers to questions such as “When X was born?”, “When was Y invented?” and “What is Z?” with fairly high accuracy. The problems of the current surface text pattern systems are the following: (i) the surface text pattern is extracted based on patterns extraction system in semi-supervising manners, and (ii) its application is limited to Q&A.

In an ongoing work to make the learning of surface text pattern less dependent on supervision, we proposed an unsupervised learning framework to learn and

construct the surface text patterns. There are four main steps in the framework.

1. **Query generator:** For a given question, the query generator creates a set of possible queries for document search using the words from the question.
2. **Web Search:** The documents are retrieved using the search engine based on the generated query. From these documents, the sentences that contain all the terms in the query are returned.
3. **Finding Answer:** The sequential pattern that is a possible answer to the given question is extracted from the returned sentences in previous step. For example, if the question is asking for a date, then any date pattern found in the sentences is extracted. The sequential patterns are ranked based on the frequency, and returned as possible answers.
4. **Pattern Learning:** Pattern learning is a four-step process: (i) create a question and answer pair using the given question and the highest ranking answer, (ii) generate the surface text patterns from the answer, (iii) if there is a new surface text pattern, use it to find new question and answer pairs from the web – terminate otherwise, and (iv) select the highest new question and answer pair, and input them to step (ii).

The connections between these units are shown in flow diagram in Figure 5.1. As an example of the Pattern Learning step, we applied our framework to the question “When was Gandhi born?”. The generated query was “Gandhi” + “born”. The highest-ranking answer found by the system was 1869. From this, the new question and answer pair, the most common surface text pattern generated was “< *NAME* > wasborn < *ANSWER* >”. This surface text pattern lead to a new answer pair “Obama” - “1961”, and a new surface pattern < *NAME* > (< *ANSWER* >).

In the future, we aim at improving the above framework and using it to create an application for information extraction using the surface text patterns. This information extraction application will create the surface text patterns based on the user defined sequential patterns. For example, if a user performs a search of “Albert Einstein”, and highlight documents discussing his Nobel Prize winning work, the application will automatically find an entity with matching features using surface text

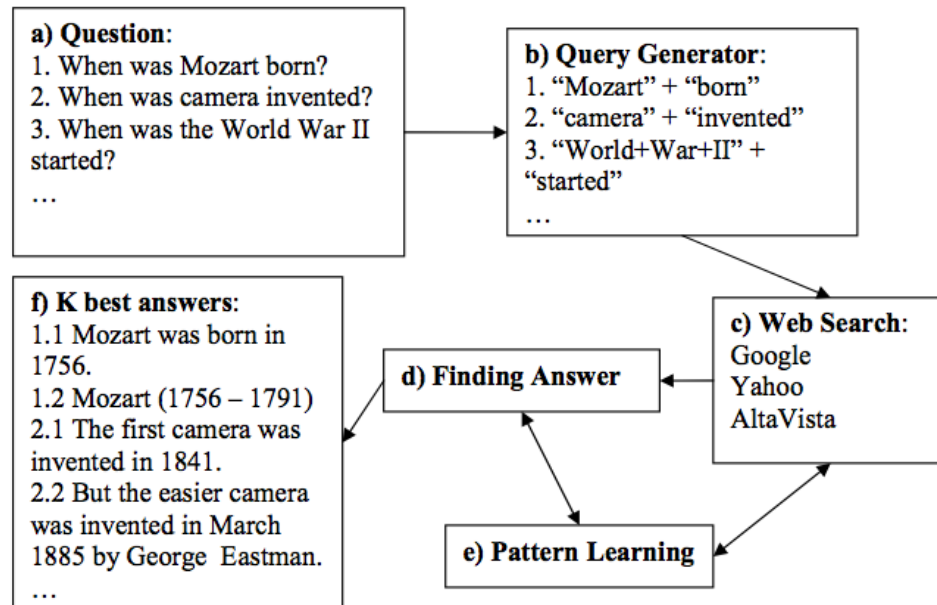


Figure 5.1: The Q&A framework that automatically generated surface text patterns from the provided questions.

patterns, “< *NAME* > received Nobel Prize Physics”. The extracted information will be used to generate search for results of similar events to the given query.

BIBLIOGRAPHY

- [1] J. I. Adibi, “Enron employee status record,” Internet: [http://isi.edu/~adibi/Enron/Enron Employee Status.xls](http://isi.edu/~adibi/Enron/Enron_Employee_Status.xls), (Date Last Accessed on Jul 10, 2007).
- [2] R. Agrawal and R. Srikant, “Mining sequential patterns,” In *Proceedings of the Eleventh International Conference on Data Engineering*, 1995, pp. 3–14.
- [3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, “Fast Discovery of Association Rules,” in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA: AAAI Press, 1996, pp. 307-328.
- [4] William P. Alston, “Philosophical Analysis and Structural Linguistics,” *The Journal of Philosophy*, vol. 59, pp. 709–720, Nov. 1962.
- [5] “Ask”, Internet: www.ask.com/, (Date Last Accessed on Oct. 25, 2009).
- [6] L. D. Baker and A. K. McCallum, “Distributional clustering of words for text classification,” in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998, pp. 96–103.
- [7] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1993, pp. 1–194.
- [8] J. Baumes, M. Goldberg, M. Hayvanovych, S. Kelley, M. Magdon-Ismael, K. Mertsalov, and W. Wallace, “Sights: A software system for finding coalitions and leaders in a social network,” in *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, 2007, pp. 193–199.
- [9] M. W. Berry, *Survey of Text Mining: Clustering, Classification, and Retrieval*, New York: Springer, 2003, pp. 1–122.
- [10] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, Jan. 2003.
- [11] T. Briscoe and J. Carroll, “Robust accurate statistical annotation of general text,” in *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, 2002, pp. 1499–1504.

- [12] R. J. Brychta, S. Tuntrakool, M. Appalsamy, N. R. Keller, D. Robertson, R. G. Shiavi, and A. Diedrich, "Wavelet methods for spike detection in mouse renal sympathetic nerve activity," *IEEE transactions on bio-medical engineering*, vol. 54, pp. 82–93, Jan. 2007.
- [13] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *Proceedings of SDAIR-94, the third Annual Symposium on Document Analysis and Information Retrieval*, 1994, pp. 161–175.
- [14] Yu-Chuan Chang, Shyi-Ming Chen, and Churn-Jung Liao, "Multilabel text categorization based on a new linear classifier learning method and a category-sensitive refinement method," *Expert Systems with Applications*, vol. 34, pp. 1948–1953, Apr. 2008.
- [15] V. Chaoji, A. Hoonlor, and B.K. Szymanski, "Recursive data mining for role Identification," in *Proceedings of IEEE/ACM Fifth International Conference on Soft Computing as Transdisciplinary Science and Technology*, 2008, pp. 218–225.
- [16] V. Chaoji, A. Hoonlor, and B.K. Szymanski, "Recursive data mining for role identification in electronic communications," *International Journal of Hybrid Information Systems*, vol. 7, pp. 89–100, Apr. 2010.
- [17] E. Charniak, "Statistical parsing with a context-free grammar and word statistics," in *Proceedings of the National Conference on Artificial Intelligence*, 1997, pp. 598–603.
- [18] H. Cheng, X. Yan, J. Han, and C. Hsu, "Discriminative frequent pattern analysis for effective classification," in *Proceedings of the 7th IEEE International Conference on Data Mining*, 2007, pp. 716–725.
- [19] K. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *Proceedings of the Second Conference on Applied Natural Language Processing*, 1988, pp. 136–143.
- [20] K. W. Church and W. A. Gale, "Poisson mixtures," *Natural Language Engineering*, vol. 1, pp. 163–190, Jun. 1995.
- [21] Aaron M. Cohen and William R. Hersh, "A survey of current work in biomedical text mining," *Briefings in Bioinformatics*, vol. 6, pp. 57–71, Mar. 2005.
- [22] W. W. Cohen, "Enron email dataset", Internet: www.cs.cmu.edu/~enron/, (Date Last Accessed on May 25, 2008).
- [23] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery, "Learning to extract symbolic knowledge from the World

- Wide Web,” in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, 1998, pp. 509–516.
- [24] F. Crestani, M. Lalmas, C.J. Van Rijsbergen, and Iain Campbell, “Is this document relevant? - probably: a survey of probabilistic models in information retrieval,” *ACM Computing Surveys*, vol. 30, pp. 528–552, Dec. 1998.
- [25] C. Curry, R. Grossman, D. Locke, S. Vejcik and J. Bugajski, “Detecting changes in large data sets of payment card data: A case study,” in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge discovery and data mining*, 2007, pp. 1018-1022.
- [26] A. Ioana Deac, J. C. A. van der Lubbe and E. Backer, “Feature selection for paintings classification by optimal tree pruning,” in *Proceedings of Multimedia Content Representation, Classification and Security, International Workshop*, 2006, pp. 354–361.
- [27] G. Doddington, “Automatic evaluation of machine translation quality using N-gram co-occurrence statistics,” in *Proceedings of the Second International Conference on Human Language Technology Research*, 2002, pp. 138–145.
- [28] A. Doms and M. Schroeder, “GoPubMed: exploring PubMed with the gene ontology,” *Nucleic Acids Research*, [On-line]. vol. 33(Web Server issue), pp. W783–W786, Available: www.ncbi.nlm.nih.gov/pmc/articles/PMC1160231/ (Date Last Access on May 14, 2011).
- [29] E. Dransfield, G. Morrot, J. F. Martin, and T. M. Ngapo, “The application of a text clustering statistical analysis to aid the interpretation of focus group interviews,” *Food Quality and Preference*, vol. 15, pp. 477–488, Jul. 2004.
- [30] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng, “Web question answering: Is more always better?,” in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002, pp. 291–298.
- [31] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, “Taming the curse of dimensionality in kernels and novelty detection,” in *Applied Soft Computing Technologies: The Challenge of Complexity*, A. Abraham, B. Baets, M. Koppen, and B. Nickolay, Eds. Berlin: Springer Verlag, 2006, pp. 431–444.
- [32] C. Faloutsos and D. W Oard, “A survey of information retrieval and filtering methods,” University of Maryland at College Park, College Park, MD, Tech. Rep. CS-TR-3514, Aug. 1995.
- [33] C. Fellbaum, *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press, 1998, pp. 1–304.

- [34] A. Frank and A. Asuncion, “UCI repository,” Internet: <http://archive.ics.uci.edu/ml>, (Date Last Accessed, Dec 14, 2010).
- [35] N. Glance, M. Hurst and T. Tomokiyo, “BlogPulse: Automated trend discovery for weblogs,” in *Proceedings of WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2004.
- [36] Phillip Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*, New York: Springer, 2000, pp. 33–40.
- [37] “Google Sets”, Internet: <http://labs.google.com/sets>, (Date Last Accessed on Oct. 22, 2009).
- [38] “Google Trends”, Internet: www.google.com/trends, (Date Last Accessed on Jul. 7, 2009).
- [39] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003.
- [40] H. Halteren, J. Zavrel, and W. Daelemans, “Improving accuracy in NLP through combination of machine learning systems”, *Computational Linguistics*, vol. 27, pp. 199–229, Jun. 2001.
- [41] M. A. Hasan, S. Salem, B. Pupakdi, and M. J. Zaki, “Clustering with Lower Bound on Similarity,” in *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2009, pp. 122–133.
- [42] Q. He, K. Chang, E. P. Lim, and J. Zhang, “Bursty feature representation for clustering text streams,” in *Proceedings of the 2007 SIAM Conference on Data Mining*, 2007, pp. 491–496.
- [43] M. A. Hearst, “Untangling text data mining,” In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999, pp. 3–10.
- [44] Miguel Helft, “Google beats forecast even as its profit tapers,” *The News York Times* (Jan. 23, 2009), sec. B p. 3.
- [45] J. R. Hobbs, M. E. Stickel, D. E. Appelt, and P. Martin, “Interpretation as abduction,” *Artificial Intelligence*, vol. 63, pp. 69–142, Oct. 1993.
- [46] A. Hoonlor, and X. Zhu. (2006, Apr.). Unsupervised learning for surface text recovery in QA system, Computer Science Department, University of Wisconsin - Madison, WI. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.4787>, (Date Last Accessed on Jun. 2, 2011).

- [47] A. Hoonlor, B. K. Szymanski, M. J. Zaki, and V. Chaoji, “Document Clustering with Bursty Information,” *Computing and Informatics*, Bratislava: Slovak University Press, (In Press).
- [48] A. Hotho, A. Nurnberger, and G. PaaB , “A brief survey of text mining,” *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, vol. 20, pp. 19–62, May 2005.
- [49] P. S. Jacobs, “Joining statistics with NLP for text categorization,” in *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992, pp. 178–185.
- [50] T. Joachims, “A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 143–151.
- [51] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *Proceedings of the 10th European Conference on Machine Learning*, 1998, pp. 137–142.
- [52] T. Joachims, *Learning to Classify Text using Support Vector Machines Methods, Theory and Algorithms*, Boston, MA: Kluwer Academic Publishers, 2002, pp. 12–16.
- [53] J. José, G. Adeva, J. Manuel, and P. Atxa, “Intrusion detection in web applications using text mining,” *Engineering Applications of Artificial Intelligence*, vol. 20, pp. 555–566, Jun. 2007.
- [54] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing*, 1st ed, Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2000, pp. 1–322.
- [55] S. Karlin and V. Brendel, “Chance and statistical significance in protein and DNA sequence analysis,” *Science*, vol. 257, pp. 39–49, Jul. 1992.
- [56] S. Karlin, “Statistical significance of sequence patterns in proteins,” *Current Opinion Struct Biology*, vol. 5, pp. 360–371, Jun. 1995.
- [57] M. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, pp. 81–89, Jun. 1938.
- [58] V. Keselj, F. Peng, N. Cercone, and C. Thomas, “N-gram-based author profiles for authorship attribution,” in *Proceedings of the Conference Pacific Association for Computational Linguistics*, 2003, pp. 255–264.
- [59] Simon King, Todd Stephenson, Stephen Isard, Paul Taylor, and Alex Strachan, “Speech Recognition using Phonetically Featured Syllables,” in *Proceedings ICSLP*, 1998, pp. 1031–1034.

- [60] J. Kleinberg, “Temporal dynamics of on-line information streams,” in *Data Stream Management: Processing High-Speed Data Streams*, M. Garofalakis, J. Gehrke, and R. RastogiSpringer, Eds. New York: Springer, 2005.
- [61] B. Klimt and Y. Yang, “The Enron corpus: A new dataset for email classification research,” in *Proceedings of the 15th European Conference on Machine Learning*, 2004, pp. 217–226.
- [62] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos, “On burstiness-aware search for document sequences,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 477–486.
- [63] J. Leskovec, L. Backstrom, and J. Kleinberg, “Meme-tracking and the dynamics of the news cycle,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 497–506.
- [64] M. S. Lewicki, “Bayesian modeling and classification of neural signals,” *Neural Computation*, vol. 6, pp. 1005–1030, Apr. 1994.
- [65] D. D. Lewis, “An evaluation of phrasal and clustered representations on a text categorization task,” in *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992, pp. 37–50.
- [66] Shoushan Li and Chengqing Zong, “Multi-domain sentiment classification,” in *Proceedings of the 46th Annual Meetings of the Association for Computational Linguistics: Human Language Technologies*, 2008, pp. 257–260.
- [67] C. Y. Lin and E. Hovy, “Automatic evaluation of summaries using n-gram co-occurrence statistics,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 2003, pp. 71–78.
- [68] H. Mannila, H. Toivonen, A. I. Verkamo, “Discovering frequent episodes in sequences extended abstract,” in *Proceedings the first Conference on Knowledge Discovery and Data Mining*, 1995, pp. 210–215.
- [69] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*, Cambridge: Cambridge University Press, 2008, pp. 1–252.
- [70] A. McCallum and K. Nigam, “A comparison of event models for Naïve Bayes text classification,” in *Proceedings of AAAI 98 Workshop on Learning for Text Categorization*, 1998, pp. 41–48.
- [71] A. McCallum, “Multi-label text classification with a mixture model trained by EM,” in *Proceedings of AAAI 99 Workshop on Text Learning*, 1999.

- [72] A. McCallum, “Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering,” Internet: www.cs.cmu.edu/~mccallum/bow, (Date Last Accessed on Dec. 14, 2010).
- [73] A. McCallum, “SRAA UseNet data”, Internet: www.cs.umass.edu/~mccallum/code-data.html, (Date Last Accessed on Jan. 17, 2011).
- [74] T. Mitchell, *Machine Learning*, Madison, Wisconsin: WCB McGraw-Hill, 1997, pp. 21–258.
- [75] M. Mitra, A. Singhal, and C. Buckley, “Automatic text summarization by paragraph extraction,” in *Proceedings of ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, 1997, pp. 31–36.
- [76] D. Mladenić, “Feature subset selection in text learning,” in *Proceedings of the 10th European Conference on Machine Learning*, 1998, pp. 95–100.
- [77] F. Mosteller and D. L. Wallace, *Inference and Disputed Authorship: The Federalist*, Reading, MA: Addison-Wesley, 1964, pp. 1–214.
- [78] J. L. Myers and A. D. Well, *Research Design and Statistical Analysis*, 2nd ed, Mahwah, NJ: Lawrence Erlbaum, 2003, pp. 47–53.
- [79] Z. Nenadic and J. W. Burdick, “Spike detection using the continuous wavelet transform,” *IEEE Transactions on Biomedical Engineering*, vol. 52, pp. 74–87, Jan. 2005.
- [80] C. G. Nevill-Manning and I. H. Witten, “Identifying hierarchical structure in sequences,” *Journal of Artificial Intelligence Research*, vol. 7, pp. 67–82, Sept. 1997.
- [81] L. Page, S. Brin, R. Motwani, and T. Winograd, “The Pagerank citation ranking: Bringing order to the web,” Stanford InfoLab, Stanford University, Palo Alto, CA, Tech. Rep. SIDL-WP-1999-0120, Jan. 29, 1998.
- [82] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 79–86.
- [83] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002, pp. 311–318.
- [84] F. Peng and D. Schuurmans, “Combining Naive Bayes and N-gram language models for text classification,” in *Proceedings of the 25th European Conference on Information Retrieval Research*, 2003, pp. 335–350.

- [85] X. Phan, “JTextPro: A Java-based text processing toolkit,” Internet: <http://jtextpro.sourceforge.net/>, (Date Last Accessed on Jan. 14, 2011).
- [86] M. F. Porter, “Snowball: A language for stemming algorithms,” Internet: <http://snowball.tartarus.org/texts/introduction.html>, (Date Last Accessed on Oct. 22, 2009).
- [87] “PubMed,” Internet: www.ncbi.nlm.nih.gov/pubmed/, (Date Last Accessed on Oct. 24, 2009).
- [88] P. Raghavan, “Information retrieval algorithms: a survey,” in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997, pp. 11–18.
- [89] D. Ravichandran and E. Hovy, “Learning surface text patterns for a question answering system,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2001, pp. 41–47.
- [90] V. C. Raykar and R. Duraiswami, “Fast optimal bandwidth selection for kernel density estimation,” in *Proceeding of the Sixth SIAM International Conference on Data Mining*, 2006, pp. 524–528.
- [91] C. Rey and J. Dugelay, “Blind detection of malicious alterations on still images using robust watermarks,” in *Proceedings of IEEE Seminar on Secure Images and Image Authentication*, 2000, pp. 7/1–7/6.
- [92] W. L. Ruzzo and M. Tompa, “A linear time algorithm for finding all maximal scoring subsequences,” in *Proceedings for the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 234–241.
- [93] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A Bayesian approach to filtering junk e-mail,” in *Proceedings AAAI’98 Workshop on Learning for Text Categorization*, 1998.
- [94] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, pp. 513–523, Aug. 1988.
- [95] G. Salton, A. Singhal, M. Mitra and C. Buckley, “Automatic Text Structuring and Summarization,” *Information Processing and Management*, vol. 33, pp. 193–207, Mar. 1997.
- [96] G. Salton and C. Buckley, “Improving retrieval performance by relevance feedback,” in *Readings in Information Retrieval*, K. S. Jones and P. Willett, Eds. San Francisco, CA: Morgan Kaufmann Publishers, 1997, pp. 355–364.
- [97] F. Sebastiani, “Machine learning in automated text categorization”, *ACM Computing Surveys*, vol. 34, pp. 1–47, Mar. 2002.

- [98] M. Serrano, A. Flammini, and F. Menczer, “Modeling statistical properties of written text,” *PLoS ONE*, [Online] 4(4): e5372, Available: www.plosone.org/article/info:doi/10.1371/journal.pone.0005372, (Date Last Accessed on Jul. 22, 2011).
- [99] Stephen Shankland and Tom Krazit, “Widespread Google outages rattle users,” Internet: <http://news.cnet.com/widespread-google-outages-rattle-users/>, (Date Last Accessed on Jul. 22, 2011).
- [100] A. Singhal, “Modern information retrieval: A brief overview,” *IEEE Data Engineering Bulletin*, vol. 24, pp. 35–42, Mar. 2001.
- [101] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, “Unsupervised learning of natural languages,” in *Proceedings of the National Academy of Sciences of the United States of America*, 2005, pp. 11629–11634.
- [102] M. Steyvers and T. Griffiths, “Probabilistic topic models,” in *Latent Semantic Analysis: A Road to Meaning*, T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, Eds. Mahwah, NJ: Lawrence Erlbaum Associates, 2007, pp. 427–448.
- [103] B. Szymanski and Y. Zhang, “Recursive data mining for masquerade detection and author identification,” in *Proceedings of the Fifth IEEE Information Assurance Workshop*, 2004, pp. 424–431.
- [104] “Text Fixer,” Internet: www.textfixer.com/resources/common-english-words.txt, (Date Last Accessed on Jul. 22, 2011).
- [105] S. Tomlinson, “Lexical and Algorithmic stemming compared for 9 European languages with hummingbird SearchServer™,” in *Working Notes for the CLEF 2003 Workshop*, Carol Peters, ed. New York: Springer, 2004, pp. 286–300.
- [106] “TREC,” Internet: <http://trec.nist.gov/>, (Date Last Accessed on Oct. 24, 2009).
- [107] O. de Vel, A. Anderson, M. Corney, and G. Mohay, “Mining e-mail content for author identification forensics,” *SIGMOD Record*, vol. 30, pp. 55–64, Dec. 2001.
- [108] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos, “Identifying similarities, periodicities and bursts for online search queries,” in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, 2004, pp. 131–142.

- [109] X. Wang, A. McCallum, and X. Wei, "Topical N-grams: Phrase and topic discovery, with an application to information retrieval," in *Proceedings of the 7th IEEE International Conference on Data Mining*, 2007, pp. 697–702.
- [110] X. Wang, C. Zhai, X. Hu, and R. Sproat, "Mining correlated bursty topic patterns from coordinated text streams," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 784–793.
- [111] "We knew web was big," Internet: <http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html>, (Date Last Accessed on Aug. 12, 2008).
- [112] Eric W. Weisstein, "Zipf's Law," *MathWorld—A Wolfram Web Resource*, Available: <http://mathworld.wolfram.com/ZipfsLaw.html>, (Date Last Accessed on Jul. 22, 2011).
- [113] J. Wermter and U. Hahn, "You can't beat frequency (unless you use linguistic knowledge) – A qualitative evaluation of association measures for collocation and term extraction," in *Proceedings of the 21st International Conference on Computational Linguistics*, 2006, pp. 785–792.
- [114] S. B. Wilson and R. Emerson, "Spike detection: a review and comparison of algorithms," *Clinical Neurophysiology*, vol. 113, pp. 1873–1881, Dec. 2002.
- [115] I. H. Witten, "Text mining," in *Practical handbook of Internet computing*, M. P. Singh, ed. Boca Raton, FL: CRC Press, 2004.
- [116] "Yahoo!Buzz," Internet: <http://buzz.yahoo.com/>, (Date Last Accessed on Oct. 24, 2009).
- [117] Y. Yang and C. G. Chute, "An example-based mapping method for text categorization and retrieval," *ACM Transactions on Information Systems*, vol. 12, pp. 252–277, Jul. 1994.
- [118] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 412–420.
- [119] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, 1999, pp. 42–49.
- [120] X. Yin and J. Han, "CPAR: Classification based on predictive association rules," in *Proceedings the Third SIAM International Conference on Data Mining*, 2003, pp. 331–335.

- [121] M. Yousef, S. Jung, L. C. Showe, and M. K. Showe, "Recursive Cluster Elimination (RCE) for classification and feature Selection from gene expression data," *BMC Bioinformatics*, [Online] 8:144, Available: www.ncbi.nlm.nih.gov/pmc/articles/PMC1877816/, (Date Last Accessed on Jul. 22, 2011).
- [122] M. Zaki, C. Carothers, and B. K. Szymanski, "VOGUE: A novel variable order hidden Markov model using frequent sequence mining," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, pp. 5:1–5:31, Jan. 2010.