1-1-2005

Incorporating semantic and syntactic information into document representation for document clustering

Yong Wang

INCORPORATING SEMANTIC AND SYNTACTIC INFORMATION INTO DOCUMENT REPRESENTATION FOR DOCUMENT CLUSTERING

By

Yong Wang

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

August 2005

INCORPORATING SEMANTIC AND SYNTACTIC INFORMATION INTO

DOCUMENT REPRESENTATION FOR DOCUMENT CLUSTERING

By

Yong Wang

Approved:

Dr. Julia E. Hodges Professor and Department Head of the Department of Computer Science and Engineering (Major Professor)	Dr. Susan M. Bridges Professor of the Department of Computer Science and Engineering (Committee Member)
Dr. Lois Boggess Professor Emerita of the Department of Computer Science and Engineering (Committee Member)	Dr. Eric Hansen Associate Professor of the Department of Computer Science and Engineering (Committee Member)
Dr. Ray Vaughn Professor of the Department of Computer Science and Engineering (Committee Member)	Dr. Ioana Banicescu Associate Professor of the Department of Computer Science and Engineering (Committee Member)

Kirk H. Schulz
Dean of the Bagley College of Engineering

Name: Yong Wang

Date of Degree: August 6, 2005

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Julia E. Hodges

Title of Study: INCORPORATING SEMANTIC AND SYNTACTIC INFORMATION INTO DOCUMENT REPRESENTATION FOR DOCUMENT CLUSTERING

Pages in Study: 121

Candidate for Degree of Doctor of Philosophy

Document clustering is a widely used strategy for information retrieval and text data mining. In traditional document clustering systems, documents are represented as a bag of independent words. In this project, we propose to enrich the representation of a document by incorporating semantic information and syntactic information. Semantic analysis and syntactic analysis are performed on the raw text to identify this information. A detailed survey of current research in natural language processing, syntactic analysis, and semantic analysis is provided. Our experimental results demonstrate that incorporating semantic information and syntactic information can improve the performance of our document clustering system for most of our data sets. A statistically significant improvement can be achieved when we combine both syntactic and semantic information. Our experimental results using compound words show that using only compound words does not improve the clustering performance for our data sets. When the compound words are combined with original single words, the combined feature

set gets slightly better performance for most data sets. But this improvement is not statistically significant. In order to select the best clustering algorithm for our document clustering system, a comparison of several widely used clustering algorithms is performed. Although the bisecting K-means method has advantages when working with large datasets, a traditional hierarchical clustering algorithm still achieves the best performance for our small datasets.

DEDICATION

I would like to dedicate this research to my parents and my wife Li Li.

ACKNOWLEDGMENTS

Here I want to express my deep and sincere gratitude to my major advisor, Professor Julia Hodges, for her patient guidance, constant encouragement and generous help throughout this work and my life as a student in the U.S.A. I also want to express appreciation to the other members of my committee, Drs. Susan Bridges, Eric Hansen, Ray Vaughn, Ioana Banicescu, and Lois Boggess, for their active participation on my committee and their invaluable aid and direction for this dissertation.

Especially, I am indebted to Dr. Lois Boggess and her student Janna Hamaker for their advice and comments about the natural language processing portion of this work. I also want to express my appreciation to Bo Tang, who received his M.S. degree under Dr. Hodges' direction and was a member of our research group.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Clustering is an unsupervised learning problem that intends to "group a set of physical or abstract objects into classes of similar objects" [32]. Document clustering partitions a set of documents into undefined groups according to their similarity. The number of resulting groups may be fixed or variable, depending on the clustering algorithms used. Document clustering is an efficient document organization method for information retrieval [75]. Several classic algorithms have been proposed for the data clustering problem including K-means, HAC (Hierarchical Agglomerative Clustering), and Buckshot. These algorithms will be explained in detail in Chapter III. One of the important prerequisites for document clustering is document representation. The method used to represent a document and capture useful information from raw text will affect the performance of a clustering system directly. In this dissertation, I identify both semantic information and syntactic information for raw text and incorporate both types of information to achieve a document representation of a document that improves the accuracy of clustering algorithms. A document clustering system was constructed to evaluate our hypotheses.

1.1 Statement of Hypothesis

Incorporating semantic and/or syntactic information for document representation will enhance the performance of the clustering process. Currently the most widely used document representation method is the feature vector in the vector space model. Each document is treated as a set of independent terms. In some document clustering systems, a document is represented as a suffix tree [97, 98], frequent term sets [8], or a document index graph [30, 31], but the essence of these representation methods is still to treat a document as a string of characters, terms or phrases with no meaning. However a set of terms is just a part of the meaning of a document. Another important component is the relationships among these terms. These relationships are reflected by the syntactic structure of a sentence and the semantic meaning of a term in a particular context. Since the goal of a clustering system is to group documents according to their content, we think the sense of a word can reflect the content of a document more precisely. The syntactic structure is helpful to identify the key words from a sentence. We predicted that the additional information provided by these relationships would improve clustering effectiveness.

Additionally, we investigated using phrases instead of single words to represent documents. Using phrases is not a new concept; a lot of researchers have worked in this direction. Better performance is reported by using phrases in [6, 30, 97]. But also some negative results have been reported [16, 26, 59]. We expected that using single words would make two unrelated documents appear to be similar because they shared the same words. On the other hand, we expected that using phrases would result in the dissimilarity

of two related documents because it is more difficult to get shared phrases from two documents than to get shared words. We hypothesized that combining the words and phrases together would overcome the weakness of both approaches. We expected that shared words would shorten the distance between two related documents and shared phrases would promote similarity between them.

1.2 Research Problems

First, how can we identify the semantic information and represent this information in the document representation? Given a raw text document, how can we find the correct sense for each word in it? One prerequisite is that a list of candidate senses must be provided. Although various word sense disambiguation (WSD) methods have been proposed (introduced in chapter VII in detail), all of them share the same characteristic, which is making use of the context of this word. Another characteristic of these algorithms is that they are time-consuming. Generally, the higher the precision, the longer the time the algorithm needs. In most research work in the WSD area, only a small set of words is used as the experimental data. But in our document clustering system, this problem is prominent because there are thousands or millions of words needing to be processed. Finding an appropriate tradeoff between the time required and the precision achieved is very important. Another problem is how to represent the sense. A good representation of sense should not only be able to differentiate among senses of a word, but also be able to merge the same sense from different words.

Second, how can we identify the syntactic information and make use of the identified information? Part-of-speech information is a very simple kind of syntactic

information. Intuitively, we assumed nouns are more important than prepositions and adverbs; we assigned a higher weight to nouns for document comparison. In this dissertation, we identified some more complex syntactic information such as syntactic tree structures. Some existing software packages can convert a flat sentence into a tree structure. Then the problem is how to find helpful information from this structure and how to incorporate this information into the document representation.

Third, how can we combine both semantic and syntactic information? We expected that incorporating both semantic information and syntactic information into the document representation would result in further improvement in the performance of our clustering system.

Fourth, how can we use the phrases to improve the representation of documents? A lot of researchers have proposed to use phrases instead of single words to represent the document. But mixed experimental results have been reported in the literature. As we stated in our hypothesis, we proposed to combine compound words with single words. We thought this combination would overcome the weaknesses of both methods.

Fourth, how can we select a proper clustering algorithm for our document clustering system? Although a lot of clustering algorithms have been proposed, most of these algorithms are data related. For our particular data (journal abstracts), we had to determine which one would be best. We worked to find one which could get the best performance and be appropriate for making comparisons.

1.3 Dissertation Outline

In this dissertation, we begin with an introduction of three background areas related to document clustering. We then provide an overview of the basic clustering algorithms, a particular document clustering algorithm, document representation, and related natural language processing (NLP) tools. In Chapter IV, we describe the architecture of our experimental system, our experimental data, and evaluation methods. Chapter V, VI, VII, and VIII present our experimental results for each research problem. Chapter IX provides our final conclusions and our contributions.

CHAPTER II

BACKGROUND

2.1 Introduction

Document clustering is an interdisciplinary subject which involves traditional information retrieval, artificial intelligence, data mining, and natural language processing. Many concepts in document clustering are inherited from these areas. This chapter provides a brief review of these disciplines as background information.

2.2 Information Retrieval

"Information retrieval (IR) is concerned with the representation, storage, organization, and accessing of information items" [75]. The most general information items are narrative information such as various documents, newspapers, books, and research papers. An information retrieval system (IRS) attempts to find relevant documents from a large collection of documents to respond to a user's query. A typical information retrieval system includes three major components: a database, a query submitted by a user, and feedback provided by the IRS [85]. The database, which is generally a collection of documents, is the information source. Each document is described by a sequence of terms. The term is a semantic unit that may be a word, a phrase or the stemmed form of a word. The query is constructed by the users according to their favorite topic. When the query is submitted to the information retrieval system, it

will be analyzed and compared with the documents in the database. The matched documents are selected as the feedback to the users. Figure 2.1 (Adapted from [85]) shows a diagram of a typical IR system.

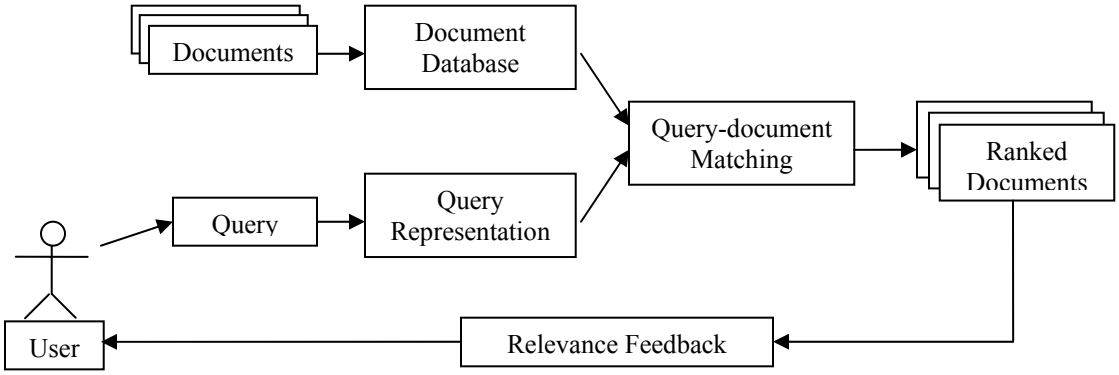

Figure 2.1 A Typical Information Retrieval System

The development of an information retrieval system is promoted by the introduction of different information sources and the increasing volume of information [5]. The most original information sources may be a sentence, an article, or a paper. Since the volume of information contained in these sources is limited, users can find what they want without any assistance. The book is one of the first information sources with an information retrieval system. The content table of a book helps the reader find the page number of a particular topic. Libraries were the most widely used information sources for a long time and are the institutions for which information retrieval systems were widely adopted for retrieving information. The oldest and simplest information retrieval system in a library is the card catalog, by which a customer can query a book or document according to its author name, date, or title. Currently, the information retrieval systems used in libraries are advanced computer management systems. In these systems, friendly

graphical interfaces are designed to help the users to find their favorite items according to more complex query facilities such as subject headings and keywords. The emergence of the World Wide Web erased the sovereign position of the traditional libraries for information sources. The Web caused a revolution in both publishing information and retrieving information. On the Web, everybody can share ideas and information without obstacles. Everyday, millions of Web pages are published and the volume of information is increasing with unprecedented speed. There is no longer the limitation of space for a library. For retrieving information, instead of driving to the nearest library, users can achieve what they want by clicking the mouse at their own home. The Web has already been developed into a universal repository of human knowledge and culture, and the various search engines comprise a huge information retrieval system for this expanding information source.

Two generally used measures for evaluating the efficiency of an information retrieval system are recall and precision. Recall is the portion of the relevant documents in a document collection that are retrieved. Precision is the portion of the retrieved documents that are relevant. With optimal performance, an information retrieval system would achieve one hundred percent for both recall and precision. In practice, high rates of recall and precision are difficult to achieve simultaneously. When the measure of matching is strict, fewer documents are retrieved or a higher precision may be achieved, but the recall will decrease. On the other hand, when more documents are retrieved, the recall will be higher, but the precision will decrease. The F-measure is another widely

used measure for information retrieval system that represents the trade-off between recall and precision.

In order to improve the efficiency of an information retrieval system (higher precision, recall and F-measure), one of the most difficult obstacles is to determine how to keep related document items relatively close together [75]. The problem is not as easy as it appears. When the topics of a collection of documents are different from each other, it is easy to separate them into different groups. But if there are interdisciplinary topics and sub-topics, the problem is complex. The original method for solving this problem was to put the documents about the same topic in the same shelf in a library. With the card catalog method, instead of rearranging the book frequently, each book was assigned a card. The cards were organized according to the content of corresponding books and a user could find the book using the card. Using an index system was another solution for this problem. Indexes are content indicators for these documents. An incoming document was analyzed and an appropriate description was selected to reflect the content. All these descriptions constructed an index system. One of the most widely used index systems is the Library of Congress Classification System (LC System). There are millions of indexes (subject headings) in this index system to describe different topics in twenty-one categories. Similar topics are related with each other in this index system. The same approach has been used for classifying Web documents. Yahoo is a successful example of an index of Web documents. A related Web page can be located by tracing a hierarchy index system easily. Traditionally, all the indexes are generated manually by human experts. This task is boring and time consuming. In order to liberate humans from this

burden, a lot of automatic index generation systems have been proposed. Some of these systems choose indexes for a document from a predefined index system such as the LC system. These systems are called classification systems. Other index generation systems can group the documents according to their content and generate an index for each group. These systems are called clustering systems. Our research is focused on the clustering systems.

2.3 Data Mining and Text Mining

"Data mining is the process of discovering interesting knowledge from large amounts of data stored either in databases, data warehouses, or other information repositories" [32]. Data mining is an interdisciplinary field that involves several different areas, including classical statistics, database technology, machine learning, artificial intelligence, and computer graphics. The most important factor that motivates the development of data mining techniques is rapidly developing database technology. From the early data collection system (primitive file processing) in the 1960s and the database management systems (hierarchical and network database system, relational database system) in the 1970s-1980s, database technology has evolved into advanced databases systems, Web-based database systems, and data warehousing systems of today. The huge amounts of available data from various sources supports the emergence of advanced data mining tools to identify the useful knowledge from raw data. The direct use of the classical statistical methods and measures such as regression analysis, standard distribution, standard deviation, standard variance, and discriminate analysis to analyze the data can be considered the most original type of data mining method. Some advanced

data mining tools used currently are still designed based on classical statistics theory. Some popular data mining methods were developed from the areas of machine learning and artificial intelligence, such as decision trees, neural networks, and Bayesian networks. The techniques of computer graphics are widely used for data visualization in the data mining process. Data mining is an essential step of knowledge discovery in databases (KDD) (sometimes they are treated as synonyms). The main stages of a KDD process can be summarized as the following 7 steps and is shown in Figure 2.2 (Adapted from [32]).

1. Data cleaning: Handling the noisy, erroneous, missing, and irrelevant data in the dataset.
2. Data integration: Integrating the data coming from different sources and in different formats.
3. Data selection: Selecting the relevant data for a particular analysis purpose.
4. Data transformation: Transforming the data into a particular form for the execution of the data mining algorithm.
5. Data mining: Performing the data-mining algorithm on the data to extract knowledge patterns.
6. Pattern evaluation: Evaluating the identified knowledge pattern for its interestingness.
7. Knowledge presentation: Visualizing and presenting the mined knowledge.

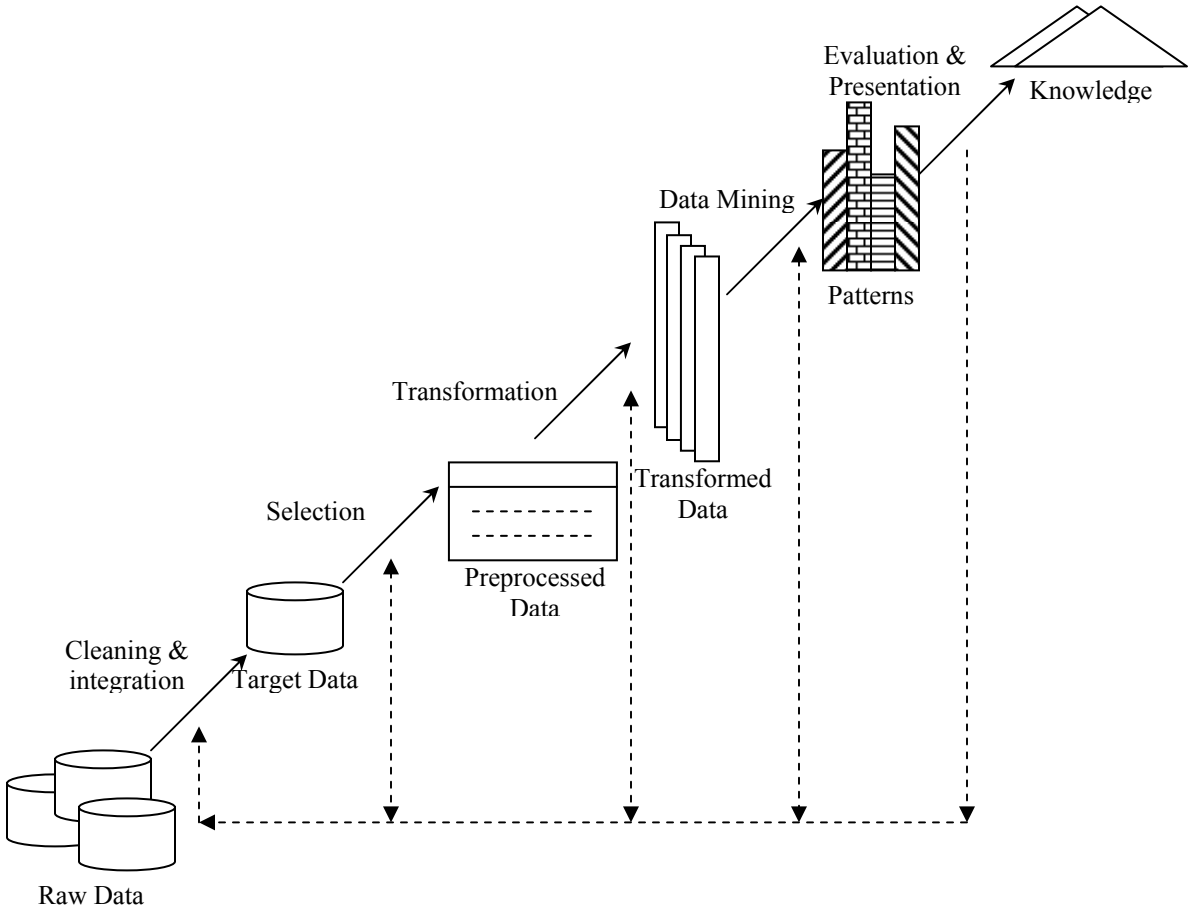

Figure 2.2 The Process of Knowledge Discovery

A data mining system may be distinguished from the others according to the kinds of databases and goals that are mined [32]. The possible mined data are relational databases, data warehouses, transactional databases, object-oriented databases, object-relational databases, spatial databases, temporal databases, time-series databases, text databases, multimedia databases, heterogeneous databases, legacy databases, and the World Wide Web. Among these databases, text databases are “databases that contain

word descriptions for objects" [32], such as papers, reports, messages, notes, Web pages, or others. Text databases may be unstructured, semistructured, or structured [32]. Data stored in most text databases are semistructured. For example, most papers contain some structured fields, such as title, authors, and date. Even most HTML Web documents have some fixed fields, such as title and headings. But most components in a document are unstructured, such as the content. The problem of mining knowledge from the text databases is also called text mining. Document clustering is a kind of text mining problem.

For different data, different data mining goals are pursued [55]. For example, given a transactional database, the purpose of data mining may be mining association rules. Given a time-series database, the most interesting topic is performing a time-series analysis. Depending on the type of data and the interest of the user, there are currently eight major data mining tasks: class description, prediction, identification, association, sequential pattern analysis, classification, clustering, and time-series analysis [55]. Classification and clustering are two popular tasks for text mining on text databases.

2.4 Natural Language Processing

Natural language processing (NLP) "concerns the development of computational models of aspects of human language processing, such as reading and interpreting a textbook, writing a letter, holding a conversation, translating a document, and search for useful information" [21]. These models are very important for developing natural language related programs and accelerating the understanding of human communication. The applications of NLP systems can be divided into two classes: processing written text

and promoting human-machine communication [72]. Applications of processing written text include machine translation, grammar checking, information retrieval, text categorization, and extracting data from text. For human-machine communication, an NLP system can be applied in speech recognition, automatic customer service over the telephone, database access, and some tutoring systems. Our use of an NLP software package in document clustering is an application of processing written text. The functionalities provided by an NLP system can be summarized as the following levels with the lowest level being parsing and phonetic analysis and the highest level being discourse understanding [42].

- Parsing: Identify terms from written text.
- Phonetic analysis: Identify terms from speech language.
- Morphology analysis: Convert a term to its original form, such as stemming and lemmatization.
- Syntactic analysis: Analyse the structure of sentences and discovers how the words in a sentence are related to each other.
- Semantic analysis: Obtain the meaning of terms and sentences.
- Pragmatics analysis: Use context information to get a better understanding of the sentence.
- Discourse understanding: Use general knowledge and knowledge of previous utterances.

In this dissertation, parsing, morphology analysis, syntactic analysis, semantic analysis and pragmatic analysis were used to improve the performance of a document clustering system.

CHAPTER III

DOCUMENT CLUSTERING

3.1 Introduction

Data clustering partitions a set of unlabeled objects into disjoint/joint groups of clusters. In a good cluster, all the objects within a cluster are very similar while the objects in other clusters are very different. When the data processed is a set of documents, it is called document clustering. Document clustering is very important and useful in the information retrieval area. Document clustering can be applied to the document database so that similar documents are related in the same cluster. During the retrieval process, documents belonging to the same cluster as the retrieved documents can also be returned to the user. This may improve the recall of an information retrieval system. Document clustering can also be applied to the retrieved documents to facilitate finding useful documents for the user. Generally, the feedback of an information retrieval system is a ranked list ordered by estimated relevance to the query. When the volume of an information database is small and the query formulated by the user is well defined, this ranked list approach is efficient. But for a very large information source, such as the World Wide Web, and poor query conditions (just one or two key words), it is difficult for the retrieval system to identify the interesting items for the user. Sometimes most of the retrieved documents are of no interest to the users. Applying documenting clustering to the retrieved documents could make it easier for the users to

browse their results and locate what they want quickly. A successful example of this application is VIVISIMO (http://vivisimo.com/), a Web search engine that organizes search results with document clustering. Another application of document clustering is the automated or semi-automated creation of document taxonomies. A good taxonomy for Web documents is Yahoo (www.yahoo.com).

Although document clustering and document classification (document categorization) are both text mining tasks that share a set of basic principles, they are still different in several essential aspects. Document classification and clustering are document organization methods. Both group related documents according to some common quality or characteristics. But the difference is how these common characteristics are identified and how partition is implemented. For the definition of common characteristics, in document classification, a set of categories is predefined. The categories may be non-hierarchical or hierarchical. The documents can be assigned only one of the labels in the fixed schema. In document clustering, documents are grouped according to their similarity. Different definitions of similarity may result in different clusters. From this view, document clustering "reveals the inherent organization structure of the corpus while document classification imposes a predefined organization scheme to the corpus" [92].

Document classification is a supervised learning problem. A set of labeled documents must be provided to train the classifier. The quality of the labeled example affects the performance of classifier. Document clustering groups the documents without any training. The goal of document clustering is to put similar documents in the same

cluster and dissimilar documents in different clusters. From this view, document classification is “learning from examples” [32] and document clustering is “learning from observation” [32].

In this chapter, a detailed background review of document clustering is provided. In section 3.2, some widely-used distance (similarity) measures are introduced. In section 3.3, some generally used data clustering algorithms are explained and in section 3.4, some particular clustering algorithms for documents are provided. Section 3.5 summarizes several document representation methods. In section 3.6, some popular natural language processing tools in the information retrieval area are introduced according to their different functions.

3.2 Distance (Similarity) Measures

Document clustering divides a set of documents into groups to minimize the inter-similarity and maximize the intra-similarity. The measure for distance or similarity is an important factor that will affect the performance of a clustering system. The distance measures in a data clustering system can be divided into two classes, object distance and cluster distance. Object distance is used to measure the distance or similarity between two objects in the dataset. In a document clustering problem, it measures the distance between two documents. The cluster distance is used to measure the distance between two clusters. Each cluster is a subset of data objects.

Object distance measure may be different for different document representation methods. In the traditional vector space model, each document is represented as a feature vector. The calculation of distance between two documents is transferred to the distance

between two feature vectors. For a qualified distance function in a vector space model, four basic properties should be satisfied [32]. The first one is non-negativity, which means the distance between any two objects should be positive or zero; it is impossible to get a negative distance. The second one is symmetry. The distance between object *A* and object *B* should be equal to the distance between object *B* and object *A*. The third one is self-similarity, which means that the distance between an object and itself should be zero. The last one is triangular inequality. Given three objects, *A*, *B*, and *C*, the distance between A and B should no more than the sum of the distance between A and C and the distance between C and B ($d(A, B) \leq d(A, C)+d(C, B)$). This property comes from the property of a triangle, which states that the length of one edge should be less than the sum of the lengths of the other two edges.

The most popular object distance measures for vectors include Manhattan distance, Euclidean distance, Minkowski distance, and the cosine coefficient method [32]. Given two vectors of length *n*, $\vec{X}_1$ and $\vec{X}_2$, $\vec{X}_1=(x_{11}, x_{12}, x_{13}, \ldots, x_{1n})$ and $\vec{X}_2=(x_{21}, x_{22}, x_{23}, \ldots, x_{2n})$, the Manhattan distance between them is defined as:

$$\text{M\_distance}(\vec{X}_1, \vec{X}_2) = \sum_{i=1}^{n} |x_{1i} - x_{2i}|$$

The Euclidean distance is defined as:

$$\text{E\_distance}(\vec{X}_1, \vec{X}_2) = \sqrt{\sum_{i=1}^{n} |x_{1i} - x_{2i}|^2}$$

Both Manhattan distance and Euclidean distance can be generalized to Minkowski distance, which is defined as:

$$\text{M\_distance}(\vec{X}_1, \vec{X}_2) = \left(\sum_{i=1}^{n} \left| x_{1i} - x_{2i} \right|^q \right)^{\frac{1}{q}}$$

where q is a positive integer. When q is set to 1, it represents Manhattan distance; when q is set to 2, it represents Euclidean distance.

The cosine coefficient method is another frequently used similarity measure for a vector space model. The intuitive understanding of this method is that we put the two vectors into n-dimensional space (n is the degree of the vector) and calculate the cosine value of the angle between these two vectors. The smaller the angle between them, the more similar these two vectors are. Since the cosine of an angle of zero degrees is 1 and it decreases as the angle widens to its maximum, we conclude that the larger the cosine, the closer the two vectors. The cosine coefficient of two vectors is the dot-product of the vectors normalized by the product of the vector lengths:

$$\text{Cosine Coefficient}(\vec{X}_1, \vec{X}_2) = \frac{\vec{X}_1 \bullet \vec{X}_2}{\left|\vec{X}_1\right| \bullet \left|\vec{X}_2\right|} = \frac{\sum_{i=1}^{n} x_{1i} \bullet x_{2i}}{\sqrt{\sum_{i=1}^{n} x_{1i}^{\,2}} \bullet \sqrt{\sum_{i=1}^{n} x_{2i}^{\,2}}}$$

There are three widely used cluster distance measures: single link (minimum distance), complete link (maximum distance), and average link (average distance). Given two clusters C_1 and C_2, suppose the number of objects in C_1 is n_1, and the number of objects in C_2 is n_2. Given a is an object in C_1 and b is an object is C_2, the object distance between a and b is defined as $|a\text{-}b|$. Then these cluster distance measures can be defined as follows:

- Single link: The distance between two clusters is the distance between the two most similar objects in both clusters.

$$\text{Single\_distance } (C_1, C_2) = \min_{a \in C_1, b \in C_2} |a - b|$$

- Complete link: The distance between two clusters is the distance between the two least similar objects in both clusters.

$$\text{Complete\_distance } (C_1, C_2) = \max_{a \in C_1, b \in C_2} |a - b|$$

- Average link: The distance between two clusters is the average distance between all the object pairs cross two clusters.

$$\text{Average\_distance } (C_1, C_2) = \frac{\sum_{a \in C_1} \sum_{b \in C_2} |a - b|}{n_1 n_2}$$

3.3 General Data Clustering Algorithms

Depending on different data types, different application and different user requirements, data clustering algorithms can be divided into hierarchical methods, partitioning methods, model-based methods, density-based methods, and grid-based methods [32]. According to the definition of a cluster, they can be divided into strict clustering algorithms (hard clustering algorithms) and fuzzy clustering algorithms (soft clustering algorithms).

3.3.1 Hierarchical Methods

Hierarchical clustering generates a hierarchical tree of clusters. This tree is also called a dendrogram [10]. Hierarchical methods can be further classified into agglomerative methods and divisive methods. In an agglomerative method, originally,

each object forms a cluster. Then the two most similar clusters are merged iteratively until some termination criterion is satisfied. This is a bottom-up approach. In a divisive method, from a cluster which consists of all the objects, one cluster is selected and split into smaller clusters recursively until some termination criterion is satisfied. A divisive method is a top-down method. Figure 3.1 is the pseudocode for the hierarchical agglomerative clustering (HAC) method. The divisive hierarchical clustering method is the reverse of the agglomerative hierarchical clustering method.

Input: A set of objects $X=\{x_1, x_2, \ldots, x_n\}$ and a termination criterion ε.
Output: A set of clusters $\{C_1, C_2, \ldots, C_k\}$
Step 1: For each element x_i in X, construct a cluster Y_i. The only element in Y_i is x_i. $Y_i=\{x_i\}$
Step 2: Calculate the distance between the each pair of cluster, Y_a and Y_b.
Step 3: Select two most similar (closest) clusters Y_a and Y_b, and merge them into one cluster.
Step 4: Test some criterion ε. If ε is satisfied, then return. Otherwise, goto step 2.

Figure 3.1 Hierarchical Agglomerative Clustering (HAC) Algorithm

In the HAC algorithm, the computational complexity of step 2 is $O(n^2)$. The complexity of step 3 is different for different intercluster distance measures. If the cluster distance computation can be done in constant time, the overall time complexity is still $O(n^2)$. In section 3.2, three different intercluster distance measures were introduced. Steinbach, Karypis, and Kumar compared the performance of three agglomerative clustering algorithms, IST (Intra-Cluster Similarity Technique), CST (Centroid Similarity Technique), and UPGMA (Unweighted Pair Group Method with Arithmatic Mean) [82]. Experimental results show that UPGMA was the best of those studied. Maarek, Fagin,

Ben-Shaul, and Pelleg [54] proposed an optimal complete-link HAC algorithm for on-line ephemeral web document clustering and achieved $O(n^2)$ time complexity.

The most generally used termination criterion in hierarchical clustering methods is obtaining a desired number of clusters. This criterion requires the user to indicate the cluster number in advance. Two other widely used criteria are the maximum diameter (or radius) of the cluster and the minimal centroid distance. The diameter of a cluster is the "maximum distance between any two objects in the cluster" [32]. The centroid distance is defined as "the average distance of each cluster object from the cluster centroid" [32]. Diameter and centroid distance are two measures for the representation of the quality of a cluster. A good clustering should have the minimum diameter and the maximum centroid distance. So in the HAC algorithm, when the diameters of all the clusters are smaller than a predefined min-diameter or the centroid distances are larger than a predefined max-centroid distance, the recursive process can be stop. These two measures are helpful to guarantee the quality of the final clusters.

3.3.2 Partitioning Methods

Partitioning clustering methods allocate data into a fixed number of non-empty clusters. All the clusters are at the same level. In the crisp clustering algorithms, each object must be assigned to exactly one cluster. In fuzzy clustering algorithms, each object can belong to different groups with different degrees (probability). The simplest partitioning clustering algorithm is a single-pass algorithm. In this method, the first object constructs the first cluster with one element. Then each object is processed one by one. The distances between the object under consideration and the existing clusters are

calculated and the object is assigned to the nearest cluster. If this nearest distance exceeds a predefined threshold, this object will create a new cluster. One obvious drawback of this method is that the final cluster is affected by the order of processing the objects and the clustering cannot be improved even it is not good enough. One solution to this problem is creating an initial partitioning. Then different iterative relocation techniques are applied to relocate the objects from one group to another group to achieve optimization. The most well-known partitioning methods following this principle are the K-means method, K-medoids method, and their variants.

The basic K-means method allocates a set of objects into a known number of clusters K to achieve highest intracluster similarity and lowest intercluster similarity [32]. This algorithm is shown in Figure 3.2.

Input: A set of objects $\{x_1, x_2, \ldots, x_n\}$ and the number of the clusters k
Output: A set of clusters $\{C_1, C_2, \ldots, C_k\}$ and their mean value $\{m_1, m_2, \ldots, m_k\}$
Step 1: Select K objects randomly as the seeds for each cluster. The value of the seeds is the initial mean value of each cluster
Step 2: For each object x_i ($1<i<n$), assign it to the cluster C_j such that distance(x_i, m_j) is minimal
Step 3: Update the mean value of each cluster.
Step 4: If no change, exit; otherwise, go back to step 2.

Figure 3.2 Basic K-means Algorithm

The time complexity of the K-means method is *O(nkt)*, where *n* is the number of objects, *k* is the number of clusters, and *t* is the number of iterations. Alsabti, Ranka, and Singh proposed an efficient K-means clustering algorithm to improve the computational complexity of the basic K-means method [3]. In this algorithm, all the objects (pattern vectors) are organized in a k-d tree structure to speed up the finding of the nearest cluster

for each object. It saves time in step 2 of the basic K-means method. The good scalability and efficiency of the K-means method make it a good choice for clustering tasks on large data sets.

A major disadvantage of the K-means method is the requirement of the structure of a cluster. It works only on clusters with a spherical structure. The pre-indication of the cluster number K is also a problem. For a set of unfamiliar data, it is difficult for a user to know the number of clusters in advance. The K-means method also requires the definition of the mean of a cluster. For some particular dataset, it is difficult to define the mean of a group of objects. The use of a mean also makes the algorithm very sensitive to noisy data. One solution for this problem is using the medoid of each cluster instead of the mean value. The medoid of a cluster is the most central object in it [32]. This algorithm is called the K-medoids method. Another disadvantage of the K-means method is that it can find only the local optimal solution. In order to find the approximate global optimal solution, the K-means method can be repeated for several times and keep the best result.

The K-means method is widely used in document clustering [9, 38]. In order to achieve better performance, most researchers have proposed some variations of K-means to improve performance. Likas et al. introduced a global K-means clustering algorithm to find the optimal K value [50]. In this algorithm, a 1-mean procedure was executed on the dataset first to get the centroid of the entire set. Then a 2-mean procedure was performed on the dataset N-1 times (where N is the size of the data set). Each time, the centroid of the 1-mean will be one seed and another point in the dataset will be selected as the

second seed. After N-1 iterations, the best cluster result is selected as the result of the 2-mean step. Then the 3-mean, 4-mean, **…**, k-mean procedures are executed. In the n-mean procedure, the centroids that were found from the (n-1)-mean procedure are used as the n-1 seeds and the n^{th} seed is selected from the remaining N-n+1 points for the optimal one.

For each K value, some criteria were proposed to measure the goodness of clusters so as to stop the iteration. Milligan and Cooper compared thirty different procedures for the selection of K and found the best one to be the VRC measure [58]. Kaufman and Rousseeuw proposed another measure called the Silhouette coefficient [43]. Another measure, partition coefficient, was introduced by Bezdek [12]. This measure is used for fuzzy clustering.

Finding the initial centers (seeds) is an important step in the K-means method. In the basic K-means method, the centers are simply selected randomly. Buckshot and Fractionation are two algorithms intended to improve this step [19]. In Buckshot, $\sqrt{n}$ objects are selected randomly as a sample set of the whole collection. Some basic cluster subroutine, such as HAC, is applied on the sample set. The centers of the K clusters on the sample set are the seeds for the whole collection. In the Fractionation method, the whole collection is broken into a number of buckets with size m, where $m>K$. The basic cluster subroutine is performed to generate ρm clusters in each bucket, where ρ is the desired reduction factor. Then all the clusters are viewed as an object and the cluster subroutine is applied to them again to generate K clusters. The centers of these K clusters are the seeds for the next step. Both the Buckshot and Fractionation methods have been

successfully used in a well-known document clustering system, the Scatter/Gather (SG) system [19]. In SG, the document set will be clustered for several iterations. In each iteration, the user can select the favorite cluster/clusters for the next iteration until the desired document is selected.

Input: A set of objects $\{x_1, x_2, \ldots, x_n\}$ and the number of the clusters k
Output: A set of clusters $\{C_1, C_2, \ldots, C_k\}$
Step 1: Pick a cluster to split.
Step 2: Find 2 sub-clusters using the basic K-means algorithm. (Bisecting step).
Step 3: Repeat step 2, the bisecting step, for ITER times and take the split that produces the clustering with the highest overall similarity.
Step 4: Repeat steps 1, 2 and 3 until the desired number of clusters is reached.

Figure 3.3 Bisecting K-means Algorithm for Finding *K C*lusters

The K-means method can also be used to generate hierarchical clusters. Steinbach, Karypis, and Kumar proposed bisecting K-means algorithm to generate hierarchical clusters by applying the basic K-means method recursively [82]. Bisecting K-means algorithm is actually a kind of divisive hierarchical clustering algorithm. Initially the whole document set is considered one cluster. Then the algorithm recursively selects the largest cluster and uses the basic K-means algorithm to divide it into two sub-clusters until the desired number of clusters is reached. Figure 3.3 (Adapted from [82]) lists the basic steps of the bisecting K-means method. Steinbach, Karypis, and Kumar's experimental results demonstrate that bisecting K-means method is better than the basic K-means approach and as good or better than the best hierarchical approach, UPGMA. Larsen and Aone developed a system to discover topic hierarchies in a huge documents

collection using the K-means algorithm [46]. The most important contribution of this system is using "vector average damping" to refine the center adjustment (step 3 in the basic K-means method).

3.3.3 Model-Based Methods

Model-based clustering methods assume that the membership of each object in each cluster follows some probability distribution [32]. Instead of just classifying the objects, a description for each cluster or model is generated in model-based clustering methods. COBWEB is a popular model-based method following a statistical approach [23]. COBWEB constructs a dendrogram, or classification tree, incrementally. Each incoming new object will be assigned to one of the nodes in the tree, which is a cluster. CLASSIT is a similar algorithm particular for numerical attributes [28]. A self-organized map (SOM) is a kind of model-based clustering method following a neural network approach [45]. A SOM also accepts the objects incrementally and assigns them into a feature map. One characteristic of SOMs are that they can be used to visualize the high-dimensional data in 2-D or 3-D space.

Wong and Fu proposed a tree structure called the DC-tree (Document Cluster tree) to facilitate the incremental document clustering process [89]. In a DC-tree, each cluster is represented with a triplet which contains the summary information for this cluster. The strategies for inserting a new document to the DC-tree, splitting a node, deleting a node, and merging nodes is similar to those of the B+ tree. Then the final clusters are identified from the DC-tree according to some measures. A DC-tree is similar to the CF (Clustering Feature) tree in the BIRCH algorithm, which an efficient data

clustering method for very large databases [99]. One drawback of this system is that once a new cluster (a new node in the DC-tree) is created, all the previous assigned documents cannot be re-assigned. Then the order of processing of the documents affects the final clustering result. Bakus, Hussin, and Kamel have reported their use of the SOM algorithm for document clustering [6].

3.3.4 Density-Based Methods and Grid-Based Methods

A density-based clustering method is designed for partitioning data in arbitrary structure, which is difficult for the K-means method [32]. In density-based clustering methods, each object keep an attribute called an ε-neighborhood, which is the number of neighborhoods with a radius ε. Instead of grouping the objects according to the inter-distances among them, density-based clustering methods extend each cluster in all directions until the objects on the boundary cannot maintain a minimum ε-neighborhood. All the objects within a cluster are density-connected. Some well-known density-based clustering methods are DBSCAN (Density-based Spatial Clustering of Applications with Noise) [44], GDBSCAN (Generalized Density Based Spatial Clustering of Applications with Noise) [76], OPTICS (Ordering Points To Identify the Clustering Structure) [4], DBCLASED (Distribution Based Clustering of Large Spatial Databases) [91], and DENCLUE (DENsity-based CLUstEring) [36].

A grid-based clustering algorithm tries to handle the data in an indirect way in order to speed up the clustering procedure [10]. A grid-based clustering method consists of four steps: input-data, grid-data, space-partitioning, and data-partitioning. The data space that contains all the input objects is quantized into a number of cells to create a grid

structure. Then instead of performing the operations on input objects, the clustering procedures are executed on the grid structure, which is called space-partitioning. The generation of the final clustering is based on the membership of each object to each grid and the results of space-partitioning. One advantage of a grid-based clustering algorithm is its speed. Since the number of grids is independent from the number of objects and is significantly less than the number of objects, the time for clustering space partitions is much less than the time for clustering the data objects. The most well-known grid-based clustering algorithm is STRING (STatistical INformation Grid-based method) which was proposed by Wang, Yang, and Muntz [87]. Another one is CLIQUE (Clustering In QUEst), which is a combination of a density-based method and a grid-based method for clustering high dimensional data [1].

3.4 Particular Document Clustering Algorithms

3.4.1 SuffixTree Method

Zamir has described the use of a suffix tree for document clustering [97, 98]. A suffix tree is a rooted, directed tree structure for representing a sequence of characters with all of its suffixes. This structure facilitates the solving of many problems on strings, such as exact string matching and the identification of a maximum match substring. Currently the most advanced algorithm for constructing a suffix tree is Ukkonen's algorithm, which just uses linear time [86]. In the SuffixTree Clustering method, each document is considered a string of words. All the documents are merged together to construct a suffix tree. Each node in this suffix tree represents a group of documents that share a common word string, which is addressed as the base cluster. All the base clusters

with a high overlap in their document sets are combined together to generate the final partitions. The major advantage of this method is that the use of a suffix tree facilitates the identification of documents that share the same word string. The problem is two documents without any common word strings may also be similar in content. These clusters are difficult to identify by the SuffixTree Clustering method. Additionally, constructing a suffix tree is also time-consuming.

3.4.2 Frequent Itemset Method

Beil, Ester, and Xu proposed two clustering methods, FTC (Frequent Term-based Clustering) and HFTC (Hierarchical Frequent Term-based Clustering), based on frequent term sets [8]. FTC generates flat clusters and HFTC creates hierarchical clusters. The concept of frequent term sets is derived from the frequent item sets in data mining for association rule mining. Each document is taken as a transaction and the terms in this document are items. The first step of the FTC method is the identification of all the frequent term sets from the documents set with the Apriori algorithm [2]. The coverage of a frequent term set is the set of documents that contains it. The coverage of a set of frequent term sets is the union of the coverage of all of them. The FTC algorithm uses a greedy algorithm to pick up a subset from all the frequent term sets whose coverage is the whole document set and the overlap among them is minimal. The coverage of the selected frequent term sets generates the final clusters; each cluster is represented by the corresponding frequent term set. The overlap documents can be assigned explicitly to achieve strict clustering. The HFTC algorithm is based on the FTC algorithm. All the frequent term sets are divided according to their cardinality, such as frequent 1-subsets

and frequent 2-subsets. The FTC is applied on all the frequent k-subsets to get the level k hierarchical clustering. Then for each cluster in level k, the FTC algorithm is applied again using the frequent (k+1)-subsets to get the level k+1 clustering. The major drawback of these two algorithms is that the clustering result is affected by the order of picking up the next frequent term set.

Fung proposed another hierarchical document clustering method based on the frequent term set, HIFC (Frequent Itemset-based Hierarchical Clustering), to solve the problem of HFTC [25]. Instead of selecting the frequent term set according to an order, HIFC constructs a cluster tree for all the frequent term sets. Frequent k-sets are mapped to the level k of the cluster tree. Then a pruning algorithm based on the similarity of the clusters is performed on the cluster tree to get the final result. HIFC achieved a higher accuracy and better efficiency and scalability than the UPGMA, Bisecting K-means, and HFTC algorithm in Fung's experiments.

3.4.3 Other Methods

Hammouda proposed an incremental clustering algorithm based on the similarity matrix of all documents [30, 31]. For each cluster, a similarity histogram is defined to represent the distribution of pair-wise document similarity in this cluster. The purpose of the document insertion strategy is to maintain a good similarity histogram for each cluster. Once a new document is assigned, a previously assigned document may be re-assigned to solve the insertion order problems. Hammouda used F-measure and entropy to measure the performance of their system and achieved better results than the traditional K-means and HAC methods. But the problem is that the test dataset for

evaluation is not a standard data set and the size is small, only 26 documents in 3 clusters. It is difficult to compare the performance of their method with the others.

Weiss, White, and Apte described a lightweight document clustering method in [88]. Different from K-means and the HAC method, this method classifies the document according to its k nearest neighborhoods in the whole collection. Experimental results demonstrate its efficiency for large document collections. But the problem is that the performance of this algorithm is affected by the goodness of group similarity measures and a single parameter, the minimum score threshold.

Lin and Ravikumar developed a soft document clustering system, WBSC (Word-based Soft Clustering) [52]. Instead of constructing a feature vector for each document and comparing the documents directly with each other, WBSC constructs the initial clusters for each unique word that occurred in the document set. All the documents containing this word belong to its cluster. Then a hierarchical-based algorithm is performed on the initial clusters to merge the similar ones iteratively until a predefined criterion is reached. The newsgroups data downloaded from the UCI KDD archive [34] and another set of Web-downloaded documents were used for experiments. Their experimental results demonstrate that WBSC system outperforms the traditional K-means method, Buckshot method, and Fractionation method.

3.5 Document Representation

3.5.1 Vector Space Model

The vector space model is probably the most widely used method for document representation in information retrieval. In this model, each document is represented by a

feature vector. The unique terms occurring in the whole document collection are identified as the attributes (or features) of the feature vector. Here terms may be single words or phrases. Sometimes some pre-processing may be applied to the terms, such as stemming, abbreviation expansion, or spelling normalization. Then the whole document set is represented as a matrix X, in which each row is a term A_i ($1<i<M$) and each column is a document D_j ($1<j<N$), where N is the total number of documents and M is the number of terms in the feature vector.

In the vector space model, different term weighting methods may be used to represent w_{ij}, the value of term A_i in document D_j. The simplest method for determining w_{ij} is a binary weighting function in which the only possible value for each term is 0 or 1. A value of 1 for term A_i indicates that term A_i occurred in document D_j (w_{ij} =1). A value of 0 indicates that term A_i did not occur in document D_j (w_{ij} = 0). A successful use of this function for document clustering is has been reported by Wong and Fu [89].

Another term weighting method for w_{ij} is term frequency (*tf*). In this method, each feature vector is represented by a list of occurrence frequencies of all terms. The value 0 of term A_i means that this term did not occur in the document. Otherwise the value of term A_i for document D_j is the frequency of term A_i in document D_j, $freq_{ij}$. In order to avoid the effect of the varying lengths of the documents, all the occurrence frequencies should be normalized before being used [74]. Then the formulation for tf_{ij} is:

$$tf_{ij} = \frac{\log(freq_{ij} + 1)}{\log(length_j)}$$

where $length_j$ is the length of document D_j. Generally it is the number of words in document D_j.

The term frequency method is not a very accurate method because the importance of a term is not decided solely by its frequency. It is also related to the entire document collection. An important term may be the medium frequency words instead of high frequency words (too common) and low frequency words (too particular). Inverse document frequency (*idf*) is a method that tries to filter those terms that occur too frequently.

$$idf_i = log(\frac{N}{n_i})$$

where N is the total number of documents and n_i is the number of documents in which term A_i occurs.

Another widely used term weighting method in the vector space model is *tf-idf* which was introduced by Salton in 1971 [73]. This method is a combination of *tf* and *idf*. The formulation is

$$W_{ij}. = tf_{ij} \times idf_i = \frac{\log(freq_{ij}+1)}{\log(length_j)} \times log(\frac{N}{n_i})$$

One obstacle of using the vector space model in IR is the problem of high dimensionality. Even for a moderate application, the number of terms will be more than tens of thousands and the number of documents may grow into the millions. It is difficult or impossible to perform any calculations on such a huge matrix X. Latent Semantic Indexing (LSI) is a good alleviation for this problem since it finds a low rank approximate matrix to substitute for the original one [20]. In the original LSI approach, matrix X is replaced by its "truncated singular value decomposition (TSVD)" [37]. This new subspace representation is a dense approximation of the original matrix. Holt

developed another technology, TRUST (Text Representation Using Subspace Transformation), to improve the traditional LSI method [37].

3.5.2 Suffix Tree and N-gram

Another representation of documents is a sequence or a set of characters or terms. Different researchers have applied different techniques to these sequences for IR tasks. In work reported by Suen [83], each document is treated as a sequence of characters and a sliding window of size n is scanned to find all the n-character sub-sequences (n-grams). The similarity between two documents is based on the number of shared n-grams. Zamir treated each document as a sequence of terms and constructed a suffix tree for an entire documents set [97]. The suffix tree facilitates the identification of common terms and phrases among documents for clustering. The order of terms in the documents will affect the similarity among them. In work reported by Beil, Ester, and Xu [8], each document is treated as a transaction and each term in it is an item. All the frequent term sets are identified. The clustering of documents is based on the common frequent item sets they shared. Both the suffix tree method and frequent term set method are introduced in section 3.3.

3.5.3 Document Index Graph

Hammouda proposed a novel structure to represent a document called DIG (Document Index Graph) [30, 31]. DIG is a directed graph consisting of a set of nodes and edges. Each node represents a unique word in the entire document set. Two successive words in any sentence are connected with a directed edge in the graph. Then a

sentence is represented by a path on a sequence of nodes corresponding to the words in that sentence. Two kinds of information are kept in the node. The first is the document information. Each node keeps a document table that records all the documents that it appears in and its frequency information. The second is the sentence path information. One word may be contained in many documents and many sentences in one document. In each entry of the document table, a list of outgoing edges is maintained for distinguishing among them. DIG facilitates the detecting of matching phrases among the documents. Once a new document is processed, a list of similarities between this document and all the previous documents is generated.

3.5.4 Universal Networking Language (UNL)

Shah, Choudhary, and Bhattacharyya proposed a new method for document feature vector construction [78]. They used the Universal Networking Language (UNL)[1] to represent each sentence in a document as a semantic graph. All the words occurring in the document are translated into concepts called Universal Words (UWs). UW describes the sense of a word in a particular context. Each UW is represented as a node in the semantic graph and the arcs among these nodes capture the semantic relations between them. The weight of each node is the number of arcs linked to it. In order to represent a document, a feature vector is constructed. Instead of using words to construct the structure of the feature vector, the identified UWs are used. This document feature vector construction method was used for document clustering by Choudhary and Bhattacharyya [17]. Given such a feature vector structure, one strategy they used for constructing the

[1] The website of UNL is located at http://www.unlc.undl.org/unlsys/ds.html

document vector was UNL graph links. The component of the feature vector is the number of links attached to that node, which represents a UW. The assumption behind this method is the more UWs related to this UW, the more important the UW. The second strategy they used was UNL relation labels, which consider the labels of the links in the graph. An n*n matrix is constructed to count the links between each pair of UWs, where n is the number of UWs in the corpus. Then all the columns of the matrix are added up to form a single dimension vector to represent the document. The clustering algorithm used in their experiments is Self Organizing Maps (SOM). The experimental results demonstrate the high accuracy of the new document representation methods.

3.6 NLP Tools

The fast development of NLP techniques provides good support for the improvement of information retrieval tasks. For the document clustering problem, some related topics in the NLP area include tokenization, morphological analysis, part-of-speech tagging, phrase boundary identification, syntactic analysis, semantic analysis, and thesaurus construction.

3.6.1 Tokenization

Tokenization is the very first step involved in most NLP processing tasks. A tokenizer separates a text into a set of component elements called tokens. Generally a token is a word, although sometimes it may be some other unknown symbol. The simplest tokenization method is splitting the text according to blanks and punctuation marks. A simple sed script implementation of a tokenizer is provided by the Penn

Treebank project group[2]. Qtoken[3] is a portable tokenizer implemented in Java. It takes a raw text file as input and generates a tokenized file as output. MXTERMINATOR[4] is a JAVA tokenizer implemented by Adwait Ratnaparkhi [71]. It is used to identify the sentence boundaries and separate the sentences in the text.

3.6.2 Morphological Analysis

Morphology analysis converts the morphological variations of a word, such as inflections and derivations, to its base form. One of the traditional methods used is a stemmer. Stemmers try to identify the stem of a raw word in a text to reduce all such similar words to a common form, making the statistical data more useful. The process of stemming removes the commoner morphological and inflexional endings from words in English. For example, the phrases *analysis, analyzer,* and *analyzing* all have the stem form *analy*. The two most widely used stemmers are the Porter[5] stemmer [66] and Lovins[6] stemmer [53]. Another morphology analysis technique is lemmatization. A lemmatizer is a linguistic suffix stripper that is more accurate than a stemmer. Instead of identifying the stem of a word, a lemmatizer converts a word to its normalized form, called a lemma. The implementation of a lemmatizer requires part-of-speech tagging, an extensive lexicon, and case normalization. For example, given the words *compute*, *computer*, *computing*, *computers*, and *computed*, a stemmer will convert all of them into *comput*. But for a lemmatizer, *compute*, *computing*, and *computed* have the same lemma

[2] Penn Treebank tokenizer can be downloaded at http://www.cis.upenn.edu/~treebank/.
[3] Qtoken software package can be downloaded at http://web.bham.ac.uk/O.Mason/software/tokeniser/.
[4] MXTERMINATOR can be downloaded at http://www.cis.upenn.edu/~adwait/statnlp.html.
[5] Porter stemmer can be downloaded at http://www.tartarus.org/~martin/PorterStemmer/index.html.
[6] Lovins stemmer can be downloaded at http://www.cs.waikato.ac.nz/~eibe/stemmers/index.html.

compute, whereas *computer* and *computers* have the same lemma *computer*. A widely used lemmarizer for English is call Morph, which is implemented with a lexical scanner based on finite-state techniques. Morph is included in a human language processing system called GATE[7].

3.6.3 Part-of-Speech Tagging

Part-of-Speech (POS) tagging is the fundamental procedure for many NLP tasks; it is also called grammatical tagging. In this process, each word is assigned a POS tag to reflect its syntactic category. POS tagging is often seen as the first stage of some other more comprehensive syntactic annotation such as phrase boundary identification. The most well known POS tagger is Brill's TBL Tagger[8]. This is a simple rule based tagger using transform-based learning [13]. Another one is the MXPOST[9] (Maximum Entropy POS Tagger developed by Adwait Ratnaparkhi [68]. LT POS is a POS tagger developed by the Language Technology Group (LTG) [56]. LT POS[10] is implemented with a Hidden Markov Model (HMM) disambiguation strategy and can be used to handle plain ASCII text and SGML marked-up text. Some other POS taggers include fnTBL tagger[11], Qtag[12], Tree-Tagger[13], and Memory Based Tagger[14].

---

[7] The homepage of GATE is located at http://gate.ac.uk/.
[8] The homepage of Brill's tagger is located at http://www.cs.jhu.edu/~brill/.
[9] MXPOST can be downloaded at http://www.cis.upenn.edu/~adwait/statnlp.html.
[10] LT POS can be downloaded at http://www.ltg.ed.ac.uk/software/pos/index.html.
[11] fnTBL can be downloaded at http://nlp.cs.jhu.edu/~rflorian/fntbl/.
[12] Qtag can be downloaded at http://web.bham.ac.uk/O.Mason/software/tagger/.
[13] Tree-Tagger can be downloaded at http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html.
[14] Memory Based Tagger can be downloaded at http://ilk.uvt.nl/software.html#mbt.

3.6.4 Phrase Boundary Identification

Identifying phrases from text is much more difficult than parsing words and sentences. There is no explicit separator between two phrases. A phrase may occur anywhere within a sentence. Another obstacle is that one phrase may include several modifiers and it is difficult to assign all the modifiers correctly. Since a phrase, especially a noun phrase, can express a notion more accurately, most researchers prefer to use it as the basic unit instead of the word [6, 97]. In order to identify phrases correctly, a phrase detector must make use of the part-of-speech information and the context information. LT CHUNK[15] is such a syntactic chunker that is capable of recognizing boundaries of simple noun and verb groups.

3.6.5 Syntactic Parser

A syntactic parser is a more complex syntactic analysis tool. The purpose of such a parser is to identify the syntactic structure of a sentence and generate a syntactic tree structure for it. The core of a syntactic parser is a set of syntactic rules and knowledge about that language called a syntactic grammar. Sleator and Temperley proposed a syntactic parser called Link Grammar Parser[16] for English in 1991 [80]. This parser applied link grammar, an original theory of English syntax, as the syntactic grammar. A robust syntactic analysis tool, some rare and idiomatic syntactic constructions are also contained in the Link Grammar Parser. Some intelligent strategies are also provided for unknown words and symbols. Michael Collins described his parser in 1999 [18]. In the

[15] LT CHUNK can be downloaded at http://www.ltg.ed.ac.uk/software/chunk/index.html.
[16] Link Grammar Parser can be downloaded at http://www.link.cs.cmu.edu/link/.

Collins parser[17], some statistical methods are applied to a training set to construct a function between a sentence and a syntactic tree structure. Another parser, LoPar[18], was proposed by Schmid [77]. Head-Lexicalised Probabilistic Context-Free Grammar (HL-PCFG), which was trained on the British National Corpus (BNC), is used in LoPar.

3.6.6 Semantic Analysis and Thesaurus/Dictionary

An important problem that most IR systems encounter is semantic analysis such as the synonym problem and polysemy problem [32]. An efficient solution to these problems is the construction of large-scale thesaurus and dictionary. In a thesaurus, various associations are created among the terms so as to refine and broaden the interpretation of these terms [75]. Thesauri use a set of internal class identifiers to organize the terms so that the retrieval of term associations is efficient and effect. Currently, the most well-known dictionary for English is the *Merriam-Webster Online Dictionary & Thesaurus*[19]. The *Merriam-Webster Online Dictionary* is based on the print version of *Merriam-Webster's Collegiate Dictionary*. Besides the huge volume of vocabulary and query functions, the Merriam-Webster dictionary also provides word function and hot-linked synonyms. Besançon, Chappelier, Rajman, and Rozenknop [11] used the synonymy relations provided by the Merriam-Webster dictionary to improve the representation of documents.

Another free online Thesaurus is WordNet[20]. WordNet is a widely used lexical database for English that provides the sense information of words [22, 57]. Different

[17] Collins parser can be downloaded at http://www.ai.mit.edu/people/mcollins/.
[18] LoPar can be downloaded at http://www.ims.uni-stuttgart.de/projekte/gramotron/resources.html.
[19] Direct access to Merriam-Webster dictionary is http://www.m-w.com.
[20] The homepage of WordNet is http://www.cogsci.princeton.edu/~wn/.

from traditional dictionaries, WordNet is organized as a semantic network. The whole corpus consists of four lexical databases for nouns, verbs, adjectives, and adverbs. The basic unit in each database is a set of synonyms called a synset. A synset represents a meaning, and all words that have such a meaning will be included in this synset. If a word occurs in several synsets, then it is polysemous. Each synset is assigned a definition or gloss to explain its meaning. There are various relationships defined between two synsets. For example, the relationship hypernymy indicates one synset is a kind of another synset (IS-A relationship); the hyponym relationship is the inverse of hypernymy. In the semantic network, each synset is represented as a node and the relationships among the synsets are represented as arcs. Hatzivassiloglou, Klavans, and Eskin used *synset*s in WordNet to improve the similarity measure among the texts [33].

CHAPTER IV

SYSTEM DESIGN

4.1 Introduction

In this chapter, the designs of our document clustering system and experiments are presented. In section 4.2, the architecture of our clustering system is introduced. Section 4.3 and 4.4 give a detailed explanation about our experimental data and the preprocessing procedures. The evaluation methods for our experiments are given in section 4.5.

4.2 Architecture

Some generally used document clustering algorithms such as K-means, HAC, and BuckShot were used in our document clustering system. Semantic analysis and syntactic analysis were performed to identify the semantic and syntactic information from the documents. This information was incorporated into the representation of a document to improve the performance of clustering. Since the semantic analysis and syntactic analysis are executed on sentences, the first step of data preprocessing is to cut each document into a set of sentences. Then semantic analysis and syntactic analysis are performed in parallel. Semantic analysis performs the sense disambiguation on each word and replaces each word with its sense. Syntactic analysis generates a syntactic tree for each sentence to identify the important parts of a sentence. The third step is to incorporate the identified

semantic information and syntactic information into the document representation. The document clustering is performed as the fourth step. The last step is evaluation to test the performance of the system. This procedure is shown in Figure 4.1.

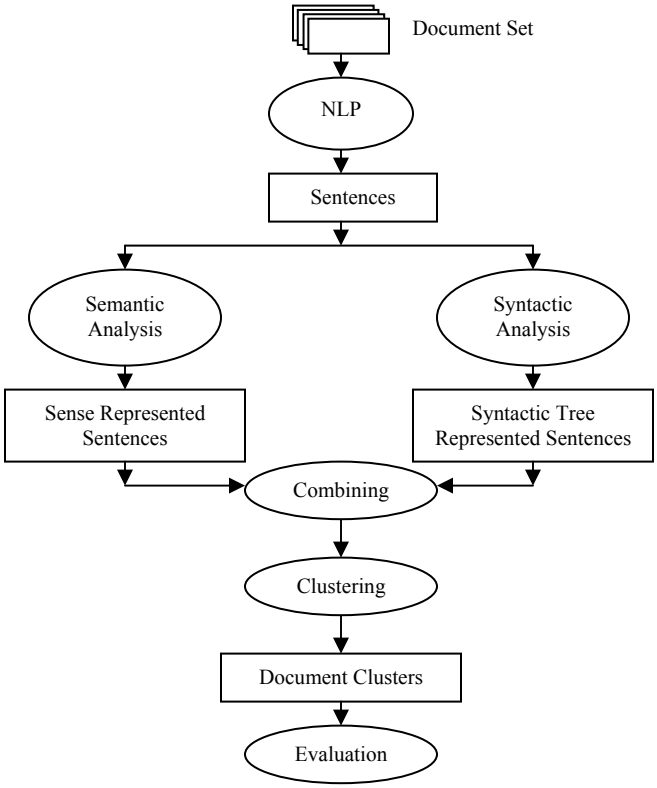

Figure 4.1 Architecture of Document Clustering System

4.3 Experimental Data

We collected 10,000 abstracts from journals belonging to ten different areas. For each area, 1000 abstracts were collected. Table 4.1 lists the areas and the names of the journals. This full data set was divided evenly into 5 subsets. Each subset contained 200 abstracts from each category, making the final size of each subset 2000. These subsets were named LDS (Large DataSet) 1 - 5.

In semantics analysis, we must identify the correct sense for each word, a procedure that is very time-consuming. In order to make our experiments more feasible, we selected another small dataset from the large dataset. The final size for each small-dataset was 200, with 20 abstracts from each category. Eight non-overlapping small-datasets were constructed and named SDS 1 - 8.

Table 4.1 Journal Abstracts Data Set

Area	Size	Journal Name
Artificial Intelligence	1000	Artificial Intelligence
Ecology	1000	Journal of Ecology
Economy	1000	Economic Journal
History	1000	Historical Abstracts
Linguistics	1000	Journal of Linguistics
Material	1000	Journal of Electronic Materials
Nuclear	1000	IEEE Transactions on Nuclear Science
Proteomics	1000	PubMed
Sociology	1000	Journal of Sociology
Statistics	1000	Journal of Applied Statistics Regression Analysis

4.4 Data Preprocessing

Since our experimental data are text documents, some of the NLP tools introduced in section 3.6 were used for data preprocessing. The basic steps include sentence parsing, tokenization (parsing words), morphological analysis (lemmatization), and Part-of-Speech Tagging. Figure 4.2 lists the detailed procedures of data preprocessing. All the documents (abstracts) were cut into the sentences with MXTERMINATOR. Then the tokens were identified from each sentence with the Penn Treebank tokenizer. The tokenized sentences were sent into MXPOST, which is a Part-Of-Speech tagger, to assign the POS tags to each token. Then the lemmatizer in WordNet was used to convert each token into a lemma and all the stop words were filtered. Finally,

a document was converted into a list of lemmas. These lemmas were used to construct the feature vector for each document.

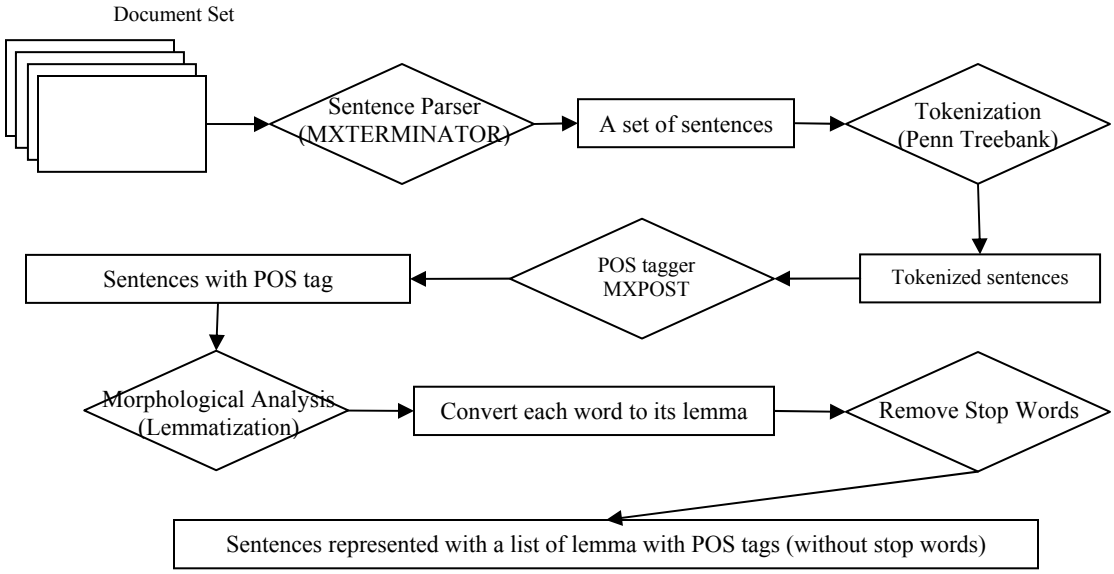

Figure 4.2 Data Preprocessing

Here an example is given for explanation. Suppose we have the sentence:

I'm the teacher of the course "Algorithms".

The Penn Treebank tokenizer will be used to generate:

I 'm the teacher of the course `` Algorithms '' .

Notice that all the words and punctuations are separated by blanks. The quotes (") are changed to doubled single forward quotes and backward quotes. The possessive noun is split into its components. For example *I'm* is split into *I* and *'m*.

The tokenized sentence generated by the Penn Treebank tokenizer is the required input format for Ratnaparkhi's part-of-speech tagger, MXPOST. The resulting sentence with part-of-speech tags generated by MXPOST is:

I_PRP 'm_VBP the_DT teacher_NN of_IN the_DT course_NN ``_`` Algorithms_NNP "_" ._.

For each token in the sentence, a part-of-speech tag is appended. The Penn Treebank Tag-Set[21] is used in MXPOST. For example, *course* is a common noun (*NN*) and *'m* is a verb in present tense (*VBP*). After the lemmatization and stop-words filtering, this sentence is represented with three lemmas with POS tag: *teacher_NN*, *course_NN*, and *algorithms_NNP*.

4.5 Evaluation Method

There are many different performance measures used by different research groups for evaluation of clustering algorithms and systems. Different measures may result in different comparison results. Generally, most evaluation methods can be divided into internal quality measures and external quality measures [82].

In internal quality measures, some external information, such as correct class labels, is unavailable. The goodness of clustering results can be measured based only on the resulting partitions. Steinbach, Karypis, and Kumar proposed to use the overall similarity as an internal quality [82]. Overall similarity is the weighted similarity of the internal cluster similarity, where internal cluster similarity is the average pairwise similarity in a cluster. Given a cluster C with size N and any two members a and b, the average pairwise similarity is calculated as:

$$\textit{Average pairwise similarity of cluster } C = \frac{\sum_{a \in C, b \in C} \textit{Similarity}(a,b)}{N^2}$$

[21] A detailed explanation of Penn Treebank Tag-Set is located at http://www.ltg.ed.ac.uk/~mikheev/tagger_demo.html#PennTagset.

The overall similarity measure emphasizes the 'cohesiveness' of a cluster. For a good cluster result, besides a minimum intra-cluster distance, a maximum inter-cluster distance is also pursued. Another internal quality measure, squared-error criterion, takes both conditions into consideration [32]. Given a set of objects and k clusters, the squared-error criterion is the average distance of all the objects to their corresponding cluster center. A better clustering result is expected to achieve a lesser value. The formulation of the squared-error criterion is [32]:

$$\text{squared-error criterion} = \sum_{i=1}^{k} \sum_{p \in C_i} \left| p - m_i \right|^2$$

where p is an object in the entire set and m_i is the mean of cluster C_i. This measure emphasizes the 'compactness' of each cluster and 'separateness' among the clusters.

In external quality measures, the known label information is combined to evaluate the performance. In these measures, the resulting clusters are compared with the original labeled classes. It is expected that the members in the same class could be allocated into the same cluster. The first widely-used external measure is entropy. Entropy is a concept from information theory which is used to characterize the homogeneity of a collection of examples [6]. A high entropy value indicates the impurity of the examples. The F-measure is the second criteria for evaluation, which also comes from information retrieval. As mentioned in section 2.1, the F-measure is a tradeoff between recall and precision.

Given a set of labeled documents belonging to I classes, assume the clustering algorithm partitions them into J clusters. Let n be the size of document set; n_i be the size

of class i; n_j be the size of cluster j; and n_{ij} be the number of documents belonging to both class i and cluster j. Then for a document in cluster j, the probability that it belongs to class i is $P(i,j)$.

$$P(i,j) = \frac{n_{ij}}{n_j}$$

The entropy of cluster j is:

$$E(j) = -\sum_{i=1}^{I} P(i,j) \log_2 P(i,j)$$

The entropy of all the clusters is the sum of the entropy of each of the clusters weighted by its size:

$$\mathrm{E} = \sum_{j=1}^{J} \frac{n_j}{n} E(j)$$

The F-measure value of each class is calculated first, and then they are combined to get the F-measure of the entire set. Given a cluster j and a class i, assume cluster j is the retrieval results of class i. Then the recall, precision, and F-measure of this retrieval are:

$$Recall(i, j) = \frac{n_{ij}}{n_i}$$

$$Precision(i, j) = \frac{n_{ij}}{n_j}$$

$$F(i, j) = \frac{2 \times Recall\ (i, j) \times Precision(i,j)}{Recall(i, j)\ +\ Precision(i, j)}$$

Since there is no one-to-one mapping relationship between each class and each cluster, any cluster can be considered as the candidate retrieval result of a class. The best F-measure among all the clusters is selected as the F-measure for the query of a particular class:

$$F(i) = MAX_{0<j<J}F(i, j)$$

The F-measure of all of the clusters is the sum of the F-measures of each class weighted by its size:

$$\mathrm{F} = \sum_{i=1}^{I} \frac{n_i}{n} F(i)$$

For our dataset, since we intentionally downloaded the abstracts from ten different categories, the label information is already known. The F-measure and entropy were used as the evaluation criterion for our experiments.

4.6 Wilcoxon Signed Rank Test

Using a statistical analysis method to evaluate the experimental results is helpful to determine if the difference between two methods is statistically significant or just a random variance. The t-test and Wilcoxon signed rank test are two of the most widely used methods for paired data. In both of them, the null hypothesis H_0 is that two groups of data sets, *A* and *B* have the same median, which means the distribution of data *A* is the same as the distribution of data *B*. For the left-side test, the alternative hypothesis is that the distribution of data set *A* is larger than that of data set *B*. For the right-side test, the alternative hypothesis is that the distribution of data set *A* is smaller than that of data set *B*. And for two-side test, the alternative hypothesis is that the distribution of data set A is

not equal to that of data set *B*. Since we wanted to determine whether one method is significantly better than the other, the one-side test was used in our experiments.

There are two prerequisites for the data in order to use the t-test. The first the one is the population of data set should be large enough. Generally it should be larger than 20. The second one is that the variance of the data should satisfy a normal distribution. For our experiments, we have eight small data sets to do the comparison, which is too small for the t-test. Additionally our experimental results are observed from different data sets and do not satisfy the normal distribution. Because of these reasons, the Wilcoxon signed rank test was selected for statistical analysis in our research work.

The Wilcoxon signed rank test is "an example of a non-parametric or distribution free test [79]." Generally, it is used to test whether the median of a distribution is equal to some particular value or whether the median of two distributions is equal or not. The Wilcoxon signed rank test can be divided into five steps [96]:

Step 1: Calculate the difference between each pair of data (experimental results). Any differences which are zero will be omitted.

Step 2: Put all non-zero differences in ascending order according to their absolute value, which means that the signs of the differences are omitted in this step, too.

Step 3: Assign the ranks to these differences. The smallest one is assigned 1. If two differences are the same, they will share the same average rank.

Step 4: Count the sum of ranks for all positive differences (W^+) and negative differences (W^-). The smaller sum is denoted as W, $W = \min(W^+, W^-)$.

Step 5: Check the value W in the table of critical values for the Wilcoxon signed-rank test to determine whether the difference between the two groups are statistically significant.

Table 4.2 Critical Values for the Wilcoxon Signed-Rank Test

One-Sided	Two-Sided	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
p = 0.05	p = 0.1	1	2	4	6	8	11
p = 0.025	p = 0.05		1	2	4	6	8
p = 0.01	p = 0.02			0	2	3	5
p = 0.005	p = 0.01				0	2	3
One-Sided	Two-Sided	n = 11	n = 12	n = 13	n = 14	n = 15	n = 16
p = 0.05	p = 0.1	14	17	21	26	30	36
p = 0.025	p = 0.05	11	14	17	21	25	30
p = 0.01	p = 0.02	7	10	13	16	20	24
p = 0.005	p = 0.01	5	7	10	13	16	19
One-Sided	Two-Sided	n = 17	n = 18	n = 19	n = 20	n = 21	n = 22
p = 0.05	p = 0.1	41	47	54	60	68	75
p = 0.025	p = 0.05	35	40	46	52	59	66
p = 0.01	p = 0.02	28	33	38	43	49	56
p = 0.005	p = 0.01	23	28	32	37	43	49
One-Sided	Two-Sided	n =23	n = 24	n = 25	n = 26	n = 27	n = 28
p = 0.05	p = 0.1	83	92	101	110	120	130
p = 0.025	p = 0.05	73	81	90	98	107	117
p = 0.01	p = 0.02	62	69	77	85	93	102
p = 0.005	p = 0.01	55	68	68	76	84	92

According to the definition of W^+ and W^-, $W^+ + W^- = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$, where n is the number of all non-zero differences. If the original null hypothesis H_0 is true, then both W^+ and W^- should follow a binomial distribution, which means $W^+ \approx W^- \approx \frac{n(n+1)}{4}$. If H_0 is false, then W^+ or W^- should be close to 0 or $\frac{n(n+1)}{2}$. Generally, the minimum value W between W^+ and W^- is used. If H_0 is true, then W is

close to $\frac{n(n+1)}{4}$. Otherwise, if H_0 is false, then W is close to 0. According to the value of W, an appropriate *p* value for this hypothesis can be calculated from the table of critical values for the Wilcoxon signed-rank test. This table is presented in table 4.2.

CHAPTER V

COMPARISON OF CLUSTERING ALGORITHMS

5.1 Introduction

We introduced various document clustering algorithms in Chapter 3. In this chapter, four of the most widely used were selected as the candidates for our clustering system: K-means, Buckshot, HAC, and bisecting K-means. All these methods have been widely used by other investigators and achieved good performance for different systems and different datasets. The purpose of our work here is to select an appropriate algorithm to construct a document clustering system for our dataset. Both good performance and stability are pursued so that our following research work can be evaluated efficiently and fairly on this system. A large data set and a small dataset are used in our experiments; our results indicated that the bisecting K-means method has advantages for large datasets. This result is consistent with reports by other researchers. But for small datasets, the traditional hierarchical clustering algorithm HAC still achieves the best performance. The different cluster distance measures and different term weighting methods in the vector space model are also evaluated in this chapter. The average-link inter-cluster distance measure and TFIDF weighting function outperformed the other candidates and are used in our final clustering system. The experimental results reported here were also published in [94].

This chapter is organized as follows: Section 5.2 presents our experimental results and analysis for different cluster distance measures in the HAC method. Section 5.3 presents our experimental results and analysis for different term weighting methods. A detailed comparison results of four different clustering algorithms and an analysis are given in section 5.4. Section 5.5 lists our conclusions.

5.2 Comparison of Different Cluster Distance Measures in HAC Method

Three different inter-cluster distance measures introduced in section 3.2 - single-link method, complete-link method and average-link method - were evaluated in this experiment. The results are listed in Table 5.1. In both F-measure (where larger is better) and entropy (where smaller is better), the average-link method achieved the best performance for all full data sets and mini data sets. Steinbach, Karypis, and Kumar compared the average-link method (UPGMA) with centroid similarity measures and found that the intra-cluster similarity method and the average-link method also outperformed the others [82]. Different from the single-link and complete-link, the average-link measure considers every pairwise distance between two elements in two clusters and averages them. This measure reflects a global relatedness between two clusters. The single-link method and complete-link method use just the shortest distance and the longest distance. This may result in bias on a pair of particular elements. Since the documents belonging to different topics are the objects for clustering in our system, the boundary between different clusters is not so explicit. There are a lot of noise documents which share features with other documents belonging to different clusters. Using the distance between two particular documents to estimate the distance between

two clusters results in the loss of comprehensive information about the whole cluster. The average-link measure is helpful to balance the side-effect of noise documents.

Table 5.1 Comparison of Inter-Cluster Distance Measures

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
Single-link	0.19	0.23	0.25	0.19	0.19	0.52	0.35	0.40	0.37	0.50
Complete-link	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.19
Average-link	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.77**	**0.78**	**0.63**	**0.77**
	Entropy									
Single-link	2.17	2.08	2.04	2.19	2.17	1.29	1.70	1.60	1.62	1.33
Complete-link	2.29	2.29	2.29	2.29	2.29	2.20	2.20	2.20	2.20	2.17
Average-link	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**

Table 5.2 Wilcoxon Signed-Rank Test for Single-Link and Average-Link Method

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
Single-link	0.19	0.23	0.25	0.19	0.19	0.52	0.35	0.40	0.37	0.50
Average-link	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.77**	**0.78**	**0.63**	**0.77**
Difference	0.55	0.32	0.35	0.4	0.47	0.25	0.42	0.38	0.26	0.27
Rank	10	4	5	7	9	1	8	6	2	3
W^+	55									
W^-	0									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									
	Entropy									
Single-link	2.17	2.08	2.04	2.19	2.17	1.29	1.70	1.60	1.62	1.33
Average-link	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**
Difference	-1.44	-0.89	-1.06	-1.1	-1.26	-0.71	-1.19	-1.08	-0.78	-0.69
Rank	10	4	5	7	9	2	8	6	3	1
W^+	0									
W^-	55									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									

The results of the Wilcoxon signed rank test for this experiment are presented in tables 5.2 and 5.3. Table 5.2 provides the test results for both F-measure and entropy between single-link method and average-link method. For F-measure, since all the

differences are positive values, we get W^+ to be 55, W^- to be 0, and W to be 0. In table 4.2, for n equals 10, if W equals 3, then the null hypothesis can be rejected with p-value of 0.005. Since we get W to be 0, we can conclude that the average-link method produced results that were significantly better than the single-link method with a p-value smaller than 0.005. For entropy, we get the same results. The improvement achieved from using the average-link method is also strongly significant with a very small p-value. Table 5.3 provides the results of the Wilcoxon signed rank test between the complete-link method and average-link method. For both F-measure and entropy, we can conclude that the average method is better than the complete-link method and the difference between them is statistically significant.

Table 5.3 Wilcoxon Signed-Rank Test for Complete-Link and Average-Link Method

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
Complete-link	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.19
Average-link	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.77**	**0.78**	**0.63**	**0.77**
Difference	0.56	0.37	0.42	0.41	0.48	0.59	0.59	0.6	0.45	0.58
Rank	6	1	3	2	5	8.5	8.5	10	4	7
W^+	55									
W^-	0									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									
	Entropy									
Complete-link	2.29	2.29	2.29	2.29	2.29	2.20	2.20	2.20	2.20	2.17
Average-link	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**
Difference	-1.56	-1.1	-1.31	-1.2	-1.38	-1.62	-1.69	-1.68	-1.36	-1.53
Rank	7	1	3	2	5	8	10	9	4	6
W^+	0									
W^-	55									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									

5.3 Comparison of Term Weighting Methods

In section 3.5.1, several different term weighting functions in the vector support model were introduced. In this experiment, Binary, Term Frequency (TF), and Term Frequency Inverse Document Frequency (TFIDF) were selected for evaluation. The HAC clustering method using average-link cluster distance measure was used as the clustering method. The results of this experiment are listed in table 5.4. As we expected, the TFIDF measure outperformed the other two measures for all 10 datasets in terms of both F-measure and entropy. The purpose of using different term weighting functions is to assign different weights to the features so that the important features which can distinguish different documents can obtain a higher weight. All three methods are based on statistical information (occurrence frequency) about the features. One major weakness of the binary method and TF method is the limitation of using local statistical information. These methods assume that a feature with a high frequency in a document is the key word for that document and should be useful to distinguish this document from the others. But actually, if such a feature is the key word for most or all documents in the whole collection, it will be useless. Good features are the terms which are important for a particular document but not important for the others. The IDF measure is intended to collect the statistical information about a term within the whole collection. The features with a low distribution among the whole collection are emphasized. The TFIDF measure is an excellent tradeoff between the TF and IDF methods. The features which are important for both the individual document and the whole collection are selected. The good performance of the TFIDF method is also found with our dataset. Since all our

documents are collected from technical journals, they share a lot of common terms which are widely used in technical literature. But since they belong to different areas, each of them contains some specific terms. The TFIDF method is helpful in avoiding those general terms and giving prominence to the area-related terms.

Table 5.4 Comparison of Different Term Weighting Methods

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
Binary	0.27	0.23	0.30	0.36	0.19	0.51	0.65	0.44	0.43	0.31
TF	0.27	0.23	0.31	0.44	0.18	0.73	0.64	0.53	0.55	0.55
TFIDF	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.78**	**0.78**	**0.63**	**0.77**
	Entropy									
Binary	2.00	2.16	1.96	1.75	2.24	1.12	0.81	1.40	1.39	1.79
TF	2.03	2.15	1.81	1.52	2.28	0.59	0.78	1.04	1.03	1.04
TFIDF	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**

Table 5.5 Wilcoxon Signed-Rank Test for Binary Method and TFIDF Method

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
Binary	0.27	0.23	0.30	0.36	0.19	0.51	0.65	0.44	0.43	0.31
TFIDF	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.78**	**0.78**	**0.63**	**0.77**
Difference	0.47	0.32	0.3	0.23	0.47	0.26	0.13	0.34	0.2	0.46
Rank	9.5	6	5	3	9.5	4	1	7	2	8
W^+	55									
W^-	0									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									
	Entropy									
Binary	2.00	2.16	1.96	1.75	2.24	1.12	0.81	1.40	1.39	1.79
TFIDF	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**
Difference	-1.27	-0.97	-0.98	-0.66	-1.33	-0.54	-0.3	-0.88	-0.55	-1.15
Rank	9	6	7	4	10	2	1	5	3	8
W^+	0									
W^-	55									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									

The Wilcoxon signed rank test for different term weighting methods is given in table 5.5 and table 5.6. Table 5.5 is the test results between the binary method and the TFIDF method, and Table 5.6 is the test results between the TF method and TFIDF method. Since the TFIDF method achieved improvement over the binary and TF methods in all ten data sets for both F-measure and entropy, we get W to be 0 for all these tests. This indicates that the TFIDF method is better than the binary method and TF method and the differences are statistically different with a p-value smaller than 0.005.

Table 5.6 Wilcoxon Signed-Rank Test for TF Method and TFIDF Method

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
TF	0.27	0.23	0.31	0.44	0.18	0.73	0.64	0.53	0.55	0.55
TFIDF	**0.74**	**0.55**	**0.60**	**0.59**	**0.66**	**0.77**	**0.78**	**0.78**	**0.63**	**0.77**
Difference	0.47	0.32	0.29	0.15	0.48	0.04	0.14	0.25	0.08	0.22
Rank	9	8	7	4	10	1	3	6	2	5
W^+	55									
W^-	0									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									
	Entropy									
TF	2.03	2.15	1.81	1.52	2.28	0.59	0.78	1.04	1.03	1.04
TFIDF	**0.73**	**1.19**	**0.98**	**1.09**	**0.91**	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**
Difference	-1.3	-0.96	-0.83	-0.43	-1.37	-0.01	-0.27	-0.52	-0.19	-0.4
Rank	9	8	7	5	10	1	3	6	2	4
W^+	0									
W^-	55									
$W=\min(W^+,W^-)$	0									
p-value	< 0.005, strongly significant (n=10)									

5.4 Comparison of Clustering Algorithms

Algorithm selection is an important prerequisite task for our research work. We decided to select a widely used clustering algorithm to compare the performance of our system with others. Based on our survey of various clustering algorithms summarized in chapter 3, four generally used algorithms were selected as candidates for comparison in

this experiment: the basic K-means method, Buckshot method, HAC method, and bisecting K-means method. All the experimental results are presented in table 5.5. In order to alleviate the effect of random seed selection in the initial step of the K-means method, Buckshot method, and bisecting K-means method, these algorithms were executed 20 times. The average F-measure and entropy are given in table 5.7.

Table 5.7 Comparison of Different Clustering Algorithms

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure									
K-means	0.77	0.72	0.79	0.72	0.74	0.41	0.48	0.42	0.39	0.42
Buckshot	0.73	0.73	0.75	0.74	0.72	0.51	0.51	0.47	0.48	0.48
HAC	0.74	0.55	0.60	0.59	0.66	**0.77**	**0.77**	**0.78**	**0.63**	**0.77**
Bisecting K-means	**0.90**	**0.85**	**0.87**	**0.86**	**0.88**	0.38	0.40	0.34	0.36	0.37
	Entropy									
K-means	0.60	0.73	0.60	0.74	0.67	1.50	1.35	1.50	1.61	1.50
Buckshot	0.67	0.70	0.65	0.72	0.71	1.28	1.24	1.38	1.37	1.34
HAC	0.73	1.19	0.98	1.09	0.91	**0.58**	**0.51**	**0.52**	**0.84**	**0.64**
Bisecting K-means	**0.40**	**0.50**	**0.46**	**0.51**	**0.45**	1.60	1.55	1.71	1.63	1.63

From these results, we found that, for all five large data sets, the bisecting K-means method outperformed the other three methods. The results of the K-means method and Buckshot method were almost the same. The HAC method had the worst F-measure and entropy for almost all of the large data sets. But for the small data sets, the results were completely different. The HAC method achieved the best performance for all five small data sets. All three of the other methods had similar unsatisfactory performance for the small data sets.

The good performance of the bisecting K-means method is consistent with the results found by Steinbach, Karypis, and Kumar [82]. Steinbach et al. concluded that “the bisecting K-means technique is better than the standard K-means approach and as good

as or better than the hierarchical approaches" [82]. According to the comprehensive analysis provided by Steinbach et al., the advantages of the bisecting K-means method resulted from characteristics of the document clustering problem. One important characteristic is the loss of the transitivity property. Generally two documents that share more common words will be considered to be more similar to each other. One situation is that document A is similar to document B and document B is similar to document C, yet document A may be totally different from document C. In the traditional clustering algorithms, the lack of transitivity may result in unexpected partitions. Another problem is that the nearest neighbors of a document may not belong to the same cluster as that document because they may be more similar to the other documents. In the HAC method, the two most similar clusters are merged in each iteration. But once two documents are grouped into the same cluster, they will not be separated in the remaining procedures. Considering the characteristics of document, we know that a reasonable clustering system should consider all the documents as a whole. The HAC method tries to achieve optimality in each step. But this local optimality may lead to the loss of global optimality. For example, suppose we have n documents. Initially we create n clusters. This must be optimal because each document represents a cluster and this is the only possible combination. Then we try to merge two of them and get n-1 clusters. This step is also optimal because the two nearest documents (document A and document B) are merged. Then a problem happens for the n-2 clusters. In the optimal combination of n-2 clusters, documents A and B may belong to different clusters to achieve optimality among the whole collection. But we can only generate n-2 clusters from the previous n-1 clusters.

This indicates that it is impossible to get the optimal n-2 clusters from the current n-1 clusters. This problem will be extended into the remaining iterations for n-3 to k clusters. Actually, the higher the level in this hierarchy structure, the fewer combination we can explore. The bisecting K-means method, Buckshot method, and basic K-means method provide a strategy to alleviate this problem. In each iteration, a set of new center elements is calculated and all the documents are reassigned to create a new combination. The side-effect of some noise documents can be controlled. An improper group can also be separated completely in the future. The strategy caters to the characteristics of document data and generates approximate optimal results for the whole collection.

As explained in Section 4.3, in our experiments, we expected to generate ten clusters for both large data sets and small data sets. But for large data sets, the initial number of clusters is 2000 and 1990 iterations will be executed. From the above analysis, we conclude that the merging operation in each iteration of the HAC method may not be helpful, and may even be harmful, to achieve the final optimality. It is more likely to move away from the correct combination in the iteration procedure for large data sets. On the contrary, the large number of documents is an advantage for K-means related methods. One weakness of K-means related methods is the selection of seeds in the initial step. A set of good seeds may lead to the final optimal results quickly while a set of poor seeds may terminate the iteration at a local peak. When the size of a data set is large, the method is more likely to identify proper seeds. One the other hand, it is easier to find the shared characteristics of each cluster when you have enough documents for common information collection. Then even if the initial seeds are not good enough, we have

enough documents to move the centers to their correct position. For the small data sets, the situation will be different. In our small data sets, we have only 200 clusters in the initial stage and only 190 merge operations are needed. The local optimality of the HAC method still dominates its side-effect. For example, if the target number of clusters equals n-1, which means only one merge is needed, the HAC method can always get the best result in theory. But for the bisecting K-means method, basic K-means method, and Buckshot method, the performance heavily depends on the random selection of seeds. And it is also difficult to identify the similarity within a cluster and dissimilarity between different clusters because of the ambiguous shape of all of the clusters. We checked the detailed execution information for the K-means, Buckshot, and bisecting K-means methods. We found that for the small data sets, generally they perform the iteration only 1 or 2 times. This indicates that, for those small datasets, the results of these K-means related methods are similar to that of a random partitioning. In order to confirm our analysis, we checked the data sets of Steinbach, Karypis, and Kumar's experiments. There were eight data sets used for evaluation by Steinbach et al. [82]. The largest data set contained about 3000 documents and the smallest one contained about 1000 documents. This size is similar to the size of our large data set. This demonstrates that the good performance of the bisecting K-means method achieved by Steinbach, Karypis, and Kumar's experiments is also based on large data sets.

Table 5.8 Wilcoxon Signed-Rank Test for Clustering Algorithms (Large Data Sets)

Data Set	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5	LDS 1	LDS 2	LDS 3	LDS 4	LDS 5
	F-measure					Entropy				
Bisecting K-means	**0.90**	**0.85**	**0.87**	**0.86**	**0.88**	**0.40**	**0.50**	**0.46**	**0.51**	**0.45**
K-means	0.77	0.72	0.79	0.72	0.74	0.60	0.73	0.60	0.74	0.67
Difference	-0.13	-0.13	-0.08	-0.14	-0.14	0.2	0.23	0.14	0.23	0.22
Rank	1.5	1.5	5	3.5	3.5	2	4.5	1	4.5	3
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				
Buckshot	0.73	0.73	0.75	0.74	0.72	0.67	0.70	0.65	0.72	0.71
Difference	-0.17	-0.12	-0.12	-0.12	-0.16	0.27	0.2	0.19	0.21	0.26
Rank	5	2	2	2	4	5	2	1	3	4
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				
HAC	0.74	0.55	0.60	0.59	0.66	0.73	1.19	0.98	1.09	0.91
Difference	-0.16	-0.3	-0.27	-0.27	-0.22	0.33	0.69	0.52	0.58	0.46
Rank	1	5	3.5	3.5	2	1	5	3	4	2
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				

Since the bisecting K-means method and HAC method have obviously different performance on large data sets and small data sets, the Wilcoxon signed rank test was performed for large data sets and small data sets separately. For large data sets, we compared the F-measure and entropy results of the bisecting K-means method with that of all the other methods. For small data sets, the F-measure and entropy results of the HAC method were compared with that of the other methods. The test results for large data sets are given in table 5.8 while the test results for small data sets are given in table 5.9. In this test, the size n of the sample was five. According to the table 4.2, the value W

should be smaller than or equal to one for the null hypothesis to be rejected with a significance level of 0.05. For all our tests, W equals zero. This indicates that for large data sets, the bisecting method is better than the other methods with a significance level smaller than 0.05 for both F-measure and entropy. For small data sets, the HAC method is the best one. The Wilcoxon signed rank test also showed that the differences are statistically significant.

The different performance of K-means method and Buckshot method can demonstrate the advantage of HAC method on small dataset again. In table 5.7, we notice for large dataset, the K-means method has a similar performance to that of the Buckshot method. But for small data sets, the Buckshot method is better than the K-means method for all five datasets. As we introduced in section 3.3.2, the only difference between Buckshot method and K-means method is the initial seeds selection. In the initial step of Buckshot method, the HAC method is applied to select the initial seeds for the following iteration. The advantage of the HAC method on small datasets is very helpful to find the better seeds.

5.5 Summary and Conclusions

In this chapter, we presented our experimental results for comparison of four basic clustering algorithms: K-means method, Buckshot method, HAC method, and bisecting K-means method. Two basic aspects of document clustering problems were also studied. From the results of our experiments we can conclude:

- In the HAC clustering method, the average-link inter-clustering distance measure is better than the single-link method and complete-link method.

- For different term weighting methods in the vector space model, the TFIDF method is better than the binary method and TF method. The TFIDF method assigns a weight to a feature by combining its importance in a document and its distinguishability for the whole document set. An important term may be a medium frequency word instead of a high frequency word (too common) or a low frequency word (too particular).

Table 5.9 Wilcoxon Signed-Rank Test for Clustering Algorithms (Small Data Sets)

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5
	F-measure					Entropy				
HAC	**0.90**	**0.85**	**0.87**	**0.86**	**0.88**	**0.40**	**0.50**	**0.46**	**0.51**	**0.45**
K-means	0.77	0.72	0.79	0.72	0.74	0.60	0.73	0.60	0.74	0.67
Difference	-0.13	-0.13	-0.08	-0.14	-0.14	0.2	0.23	0.14	0.23	0.22
Rank	1.5	1.5	5	3.5	3.5	2	4.5	1	4.5	3
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				
Buckshot	0.73	0.73	0.75	0.74	0.72	0.67	0.70	0.65	0.72	0.71
Difference	-0.17	-0.12	-0.12	-0.12	-0.16	0.27	0.2	0.19	0.21	0.26
Rank	5	2	2	2	4	5	2	1	3	4
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				
Bisecting K-means	0.74	0.55	0.60	0.59	0.66	0.73	1.19	0.98	1.09	0.91
Difference	-0.16	-0.3	-0.27	-0.27	-0.22	0.33	0.69	0.52	0.58	0.46
Rank	1	5	3.5	3.5	2	1	5	3	4	2
W+	15					0				
W-	0					15				
W	0					0				
p-value	<0.05, significant (n=5)					<0.05, significant (n=5)				

- For large data sets, the bisecting algorithm outperforms all the other methods. But for small data sets, the HAC method gets the best performance.

- The K-means method has a performance that is similar to that of the Buckshot method for large data sets. But for small data sets, the Buckshot method is better than the K-means method. Since the HAC method is used to select seeds in the initial step of Buckshot method, the better performance of HAC method in small data sets is helpful for identifying the better seeds for the following iteration in the Buckshot method.
- All the experimental results presented in this chapter were evaluated using the Wilcoxon signed rank test and the improvements were determined to be statistically significant.

According to our experimental results and conclusions, the HAC method was selected to construct our document clustering system for several reasons. The first reason was that our research focuses on the semantic analysis and syntactic analysis of raw text documents. These natural language analysis procedures are complex operations and time consuming. It is infeasible and inefficient to perform these operations on large data sets. The good performance of the HAC method on small data sets in our experiments made it an attractive choice. The second reason was the stability of the HAC method. There is no random factor involved in the method. Given a data set and similarity measure, fixed partitions are generated. This was very helpful in our evaluation of whether the semantic information and syntactic information identified from documents can improve the similarity measures between the documents.

CHAPTER VI

USING COMPOUND WORDS

6.1 Introduction

Traditionally single words occurring in documents are identified to determine the similarities among documents. In this chapter, we report the experimental results of using compound words as features for document clustering. Our experimental results demonstrate that using compound words alone cannot improve the performance of clustering systems significantly. Better results are achieved when the compound words are combined with the original single words as the features. But when a statistical analysis is applied to our results, it is demonstrated that the differences between them are all not statistically significant. In section 6.2, some related work about using phrases for information retrieval is described. The detailed experimental results are presented in section 6.3. Section 6.4 gives our analysis and conclusions. The experimental results reported here were also published in [95].

6.2 Related Work

Zamir proposed to use a suffix tree to find the maximum word sequences (phrases) between two documents [97]. Two documents sharing more common phrases are more similar to each other. Hammouda proposed a graph structure, Document Index

Graph (DIG), to represent documents [30]. The shared word sequences (phrases), which form a path within this graph, are identified to measure the distances among documents.

Bakus, Hussin, and Kamel used a hierarchical phrase grammar extraction procedure to identify phrases from documents and used these phrases as features for document clustering [6]. The self-organizing map (SOM) method was used as the clustering algorithm. An improvement was demonstrated when using phrases rather than single words as features for document clustering.

Mladenic and Grobelnik used a Naive Bayesian method to classify documents based on word sequences of different lengths [61]. Experimental results showed that using the word sequences whose length was no more than 3 words improved the performance of a text classification system. But when the average length of used word sequences was longer than 3 words, there was no further improvement detected. Caropreso, Matwin, and Sebastiani reported their evaluation results from using bigrams (statistical phrases with length 2) for automated text categorization tasks independent of the classifier used [16]. The positive effectiveness of using bigrams cannot be completely confirmed from their experimental results. On the contrary, in some particular situation, the performance may be decreased.

Furnkranz, Mitchell, and Riloff investigated the use of phrases to classify text on the WWW [26]. An information extraction system, AUTOSLOG-TS, was used to extract linguistic phrases from web documents. A naive Bayes classifier, “RAINBOW,” and a rule learning algorithm, “RIPPER,” were used to evaluate the use of phrasal features with the measures ‘precision’ and ‘recall’. The results showed that phrasal features can

improve the precision at the low recall end but not at the high recall end. Mitra et al. investigated the impact of both syntactic and statistical phrases in IR [59]. Their results demonstrated that averagely there is no significant improvement was detected for precision at high ranks and there is no significant difference between using syntactic phrases and using statistical phrases either.

6.3 Experimental Results

The compound words corpora used in this experiment was provided by WordNet. A brief introduction about WordNet is provided in section 4.5. Besides the general single words, WordNet also provides a set of widely used compound words. There are 63,218 compound words collected in WordNet. All these compound words are partitioned into four different lexical databases and organized into a semantic network. There are 58,856 noun compound words, 2,794 verb compound words, 682 adjective compound words, and 886 adverb compound words. In this experiment, we investigated using these compound words instead of the original single words as features to cluster the documents.

According to the conclusions of the previous chapter, the document clustering system used in this experiment was constructed with the HAC method. The TF-IDF weighting function and average-link distance measure were adapted. Three different term sets were retrieved from the whole collection of documents as features. The first one was the set of all single words; the second one was the set of all identified compound words; and the third one was the union of them. For example, given the phrase “artificial intelligence” in a document, two features were created from it, the adjective “artificial” and the noun “intelligence”. But in the second feature set, only one feature was created,

the compound word “artificial intelligence”. The third feature set was a combination of the first one and second one and resulted in the creation of three features, the adjective “artificial,” the noun “intelligence,” and the compound word noun “artificial intelligence”. The experimental results are listed in table 6.1.

Table 6.1 Experimental Results of Using Compound Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single word	0.6900	0.7420	0.6967	0.6445	0.6973	0.7755	0.6944	0.6436
Compound word	0.5575	0.6388	**0.7738**	0.5706	**0.7662**	0.7547	0.7644	**0.6452**
Combined	**0.7737**	**0.7536**	0.6766	**0.6458**	0.5968	**0.8381**	**0.7752**	0.5800
	Entropy							
Single word	0.7440	**0.5129**	0.8033	0.7901	0.7516	0.5985	0.7686	0.9630
Compound word	1.0077	0.8237	**0.5740**	0.9220	**0.6238**	0.6141	0.6404	**0.9539**
Combined	**0.5771**	0.5266	0.8324	**0.7728**	0.9157	**0.4655**	**0.5925**	1.0989

We found that for the eight data sets, using combined words got the best F-measure for five of the data sets while using compound words alone outperformed the others on only three data sets. Only in the second data set did original single words get slightly better entropy than the combined words. This demonstrates that using combined words is a better choice than using the original single words or using compound words alone for most of our data sets.

Actually, the effect of using compound word is tightly related to the experimental data. Compound words can provide more detailed information about the content of a document. We expected to take advantage of this information to get more precise partitions. Given a set of documents belonging to different categories, if the distances among these categories are small, then using compound words will be helpful. In this situation, using compound words can increase the intra-similarity between documents

within the same group and increase the distance among different clusters. On the contrary, if these categories are totally different, then the side-effect of using compound words will emerge. The distance between two documents within the same cluster will be enlarged because of using compound words. In other words, using compound words will reduce the density of each cluster and make the boundary among all clusters ambiguous. For example, given two documents, suppose one of them contains 'Biological Sciences' and the other contains 'Computer Sciences'. These two compound words will play different roles in different clustering problems. If all the documents within the collection are related to only research topics belonging to the computer area and biology area and we want to distinguish them, obviously using these two compound words will be helpful. But if this collection contains documents related to various topics, but belonging to the same category, then using the compound words will be harmful. When using single words, the shared word 'science' can move these two documents closer. But when using compound words, the distance between these two compound words is the same as the distance between any two unrelated words. The common characteristics between them can not be identified. When we use the combined words set, this problem can be alleviated. Since both single words and compound words are used, the single words can capture the basic similarity between them and the compound words can be used for further refinement. Going back to our example, in the first situation, the shared word 'science' will result in some similarity between the documents and the compound word will be used to distinguish them. In the second situation, compound words are useless. But the shared single words will pull the technical documents together. Table 6.2 lists the

average pairwise similarity within different data sets when using single words, compound words, and combined words. From this table we can find that for all data sets, when the compound words are used, the average similarities are decreased. This means the density of the collection is diluted. When the combined words are used, the distances between some documents are reduced. Then we get a higher average pairwise similarity as shown in table 6.2. From table 6.2, we see that the average similarities of using combined words are always the tradeoff between using single words or compound words alone.

Table 6.2 Average Pairwise Similarity Within Different Data Sets

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
Single word	0.01962	0.01946	0.01872	0.02017	0.01907	0.01867	0.01802	0.01766
Compound word	0.01859	0.01863	0.01792	0.01912	0.01811	0.01785	0.01708	0.01690
Combined	0.01922	0.01911	0.01837	0.01975	0.01875	0.01842	0.01771	0.01736

Table 6.3 Wilcoxon Signed-Rank Test for Using Single Words and Compound Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single Words	**0.6900**	**0.7420**	0.6967	**0.6445**	0.6973	**0.7755**	0.6944	0.6436
Compound Words	0.5575	0.6388	**0.7738**	0.5706	**0.7662**	0.7547	**0.7644**	**0.6452**
Difference	-0.1325	-0.1032	0.0771	-0.0739	0.0689	-0.0208	0.0700	0.0016
Rank	-8	-7	6	-5	3	-2	4	1
W^+	14							
W^-	22							
$W=\min(W^+,W^-)$	14							
p-value	H_0 cannot be rejected							
	Entropy							
Single Words	**0.7440**	**0.5129**	0.8033	**0.7901**	0.7516	**0.5985**	0.7686	0.9630
Compound Words	1.0077	0.8237	**0.5740**	0.9220	**0.6238**	0.6141	**0.6404**	**0.9539**
Difference	0.2637	0.3108	-0.2293	0.1319	-0.1278	0.0156	-0.1282	-0.0091
Rank	7	8	-6	5	-3	2	-4	-1
W^+	22							
W^-	14							
$W=\min(W^+,W^-)$	14							
p-value	H_0 cannot be rejected							

Table 6.4 Wilcoxon Signed-Rank Test for Using Single Words and Combined Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single Words	0.6900	0.7420	**0.6967**	0.6445	**0.6973**	0.7755	0.6944	**0.6436**
Combined Words	**0.7737**	**0.7536**	0.6766	**0.6458**	0.5968	**0.8381**	**0.7752**	0.5800
Difference	0.0837	0.0116	-0.0201	0.0013	-0.1005	0.0626	0.0808	-0.0636
Rank	7	2	-3	1	-8	4	6	-5
W^+	16							
W^-	20							
$W=\min(W^+,W^-)$	16							
p-value	H_0 cannot be rejected							
	Entropy							
Single Words	0.7440	**0.5129**	**0.8033**	0.7901	**0.7516**	0.5985	0.7686	**0.9630**
Combined Words	**0.5771**	0.5266	0.8324	**0.7728**	0.9157	**0.4655**	**0.5925**	1.0989
Difference	-0.1669	0.0137	0.0291	-0.0173	0.1641	-0.1330	-0.1761	0.1359
Rank	-7	1	3	-2	6	-4	-8	5
W^+	15							
W^-	21							
$W=\min(W^+,W^-)$	15							
p-value	H_0 cannot be rejected							

Next we investigated whether the differences between using single words, using compound words, and using combined words were significantly different. Table 6.3 and 6.4 present the Wilcoxon signed rank test applied to the experimental results listed in table 6.1. Table 6.3 is the comparison between using single words and using compound words, and table 6.4 is the comparison between using single words and using combined words. Since the sample size n is 8, from table 4.2 we know that the value of W must be equal to or smaller than 6 in order to reject the null hypothesis. But instead, we got high W values ranging from 14 to 16. This indicates that although there are differences between the performance of using single words, compound words, and combined words in our experiments, we cannot draw the conclusion that using combined words is significantly better than using single words and compound words. The major reason for this is still the data-related property of using compound words. Notice that either in using

compound words or in using combined words, obvious improvement can be observed for some data sets. But a big decrease can also be detected for some other data sets.

6.4 Summary

In this chapter, we investigated using compound words provided by WordNet as features for document clustering. After describing some related work about using phrases for information retrieval, the results of our experiments were presented. From these results, we found that using compound words alone did not improve the clustering performance for most of our data sets. When the compound words and the original single words were used together, the combined feature set achieved better performance for most of our data sets. But the statistical analysis indicated that this improvement is not statistically significant.

CHAPTER VII

USING SEMANTIC INFORMATION

7.1 Introduction

The vector space model is one of the most widely used document representation methods. Traditionally, the unique words occurring in a document set are used as the features in a vector space model. But because of the synonym problem and the polysemy problem, a bag of original words cannot represent the content of a document precisely. In this chapter, we investigate using the sense of words to construct the feature vector. The sense disambiguation method used in our experiments is based on the semantic relatedness among the senses. Different semantic relatedness measures were evaluated in our experiments. Our experimental results show that, for most of our data sets, using sense can improve the performance of our document clustering system. But the comprehensive statistical analysis performed indicates that the differences between using original single words and using senses of words are not statistically significant.

The rest of this chapter is organized as follows: An introduction about word sense disambiguation methods is described in section 7.2. Section 7.3 presented a detailed survey about various semantic relatedness measures. We present our experimental results and analysis in section 7.4. Section 7.5 is a brief summary of this chapter.

7.2 Word Sense Disambiguation (WSD)

The problem of finding the correct sense of a word in a context is called word sense disambiguation (WSD) [48]. There are two general types of methods for sense determination [24]. Some methods use the definition of each sense in a dictionary. In WordNet, the definition is the gloss assigned to each synset. Other approaches use the semantic relatedness in an existing semantic network.

Lesk proposed to disambiguate the sense of a polysemous word based on its context words and the definitions of all its senses in a dictionary [48]. For each sense of a word, a lexicon definition can be found in a dictionary. All the words occurring in this definition compose the sense bag for this sense. All the words occurring in the context of this word compose the context bag for this word. The context bag of a word is compared to each sense bag of all candidate senses. The sense with the maximum match is selected.

Fragos, Maistros, and Skourlas [24] proposed to improve Lesk's method by enriching the sense bag and the context bag. For each word in the sense bag, all the words that have a hypernymy/hyponymy relationship with it are also identified. The definitions of these newly identified words are incorporated into the sense bag. For the context words, all the words that have a hypernymy relationship with the current context word and their definitions are added into the context bag.

Gomes et al. [29] present a word sense disambiguation method based on the semantic distance among the context words in WordNet. Given a set of context words $\{W_1, W_2, \ldots W_n\}$ from WordNet, we can get a set of senses for each word. Suppose the number of senses for word i is s_i, then we get a set of senses $\{S(1, 1), S(1, 2), \ldots, S(1, s_1),$

S(2, 1), S(2, 2), ...,S(2, s_2), ..., S(n, 1), S(n, 2), ...S(n, s_n)}. With different measures, we can get the semantic distance between two senses, *SemanticDist(S(i, j), S(k, l))*. The shortest semantic distance between a sense *S(i, j)* and a word W_k is:

$$\text{ShortestDist}(S(i, j), W_k) = \underset{1<l<sl}{Min} \{\text{semanticSim}(S(i, j), S(k, l))\}$$

For this sense, its synset score is defined as the sum of all shortest semantic distances between this sense and all the other words. The synset score reflects the context semantic distance between this sense and all context words.

$$\text{SynsetScore}(S(i, j)) = \sum_{k=1}^{n} ShortestDist(S(i, j), W_k)$$

Then for each word, the sense which has the minimum synset score will be selected as its sense.

$$\text{Sense}(W_k) = \underset{1<l<sl}{Min} \{\text{SynsetScore}(S(k, l))\}$$

Sussna [84] proposed a similar disambiguation method based on semantic distance. Instead of finding the shortest distance between one sense and one word, Sussna's method uses all senses and tries to find a combination of candidate senses to achieve the minimum total pairwise semantic distance among the selected senses.

Li, Szpakowicz, and Matwin [49] proposed eight heuristic rules for sense disambiguation based on WordNet semantic relatedness. The basic idea in this method is to try to find verb-noun pairs in the context with semantic relatedness in WordNet. Those related verb-noun pairs were used to determine the sense for each other.

Ramakrishnanan and Bhattacharyya [67] also proposed the use of synsets in WordNet to replace the original word for text representation. But they are using a soft

sense disambiguation method in which each word will be assigned several senses instead of a single sense. Three algorithms, hubs and authorities, page ranking, and Bayesian inferencing, were described to rank the candidate synsets.

7.3 Semantic Relatedness Measures

A semantic relatedness measure is a criterion to scale the relatedness of two senses in a semantic network. It is also called semantic similarity or semantic distance in the literature. In a lot of word sense disambiguation algorithms, a semantic relatedness measure is a very important factor in the algorithm's performance. Budanitsky presents a comprehensive overview of various semantic relatedness measures [14, 15]. Jiang and Conrath [39] classify those methods into two categories, edge-based methods and node-based (information content-based) methods.

7.3.1 Edge-Based Methods

Edge-based methods attempt to measure the distance between two senses according to the length of the path between them in the semantic networks. The simplest method is to count the number of edges or nodes between them.

Hirst and St-Onge proposed two assumptions about semantic relatedness [35]. The first one is that two concepts are semantically close if the length of the path between them is not too long. The second one is that two concepts are semantically close if the path between them does not change direction too often. Based on these two assumptions, the semantic similarity formula for two concepts is:

$$SemanticSim(c1, c2) = C - PathLength(c1, c2) - k \times d$$

where d is the number of changes of direction and C and k are constant. If there is no path between them, the similarity will be zero.

Leacock and Chodorow have described their semantic similarity formula for two words [47]. Given a semantic network, the length between two senses will be the length of the shortest path between them. For two words, w_1 and w_2, the semantic similarity between them is:

$$Sim(w_1, w_2) = \max_{c1,c2} [-log \frac{length(c_1, c_2)}{2 \times D}]$$

where c_1 is a sense of w_1 and c_2 is a sense of w_2. D is the maximum depth of the taxonomy.

Sussna's approach is also based on the path length between two senses [84]. Suppose c_1 and c_2 are two neighbor nodes in the semantic network. For a particular relationship r between c_1 and c_2, the weight from c_1 to c_2 for relationship r is:

$$w(c_1 \rightarrow_r c_2) = max_r - \frac{max_r - min_r}{n_r(c_1)}$$

where max_r and min_r are predefined maximum and minimum weights for relationship r and $n_r(c_1)$ is the number of relationships r leaving from c_1.

In WordNet, the relationship r may be hypernymy, hyponymy, holonymy, or antonymy, among others. Each relationship r has an inverse relationship r', such as hypernymy and hyponymy. The weight for the edge between two neighbor nodes c_1 and c_2 is defined as:

$$w(c_1, c_2) = \frac{w(c_1 \rightarrow_r c_2) + w(c_2 \rightarrow_{r'} c_1)}{2d}$$

where d is the depth of the deeper node within c_1 and c_2. Finally, the semantic distance between any two nodes in the semantic network is the sum of the weights of the edges on the shortest path between them.

Wu and Palmer [90] defined conceptual similarity between two concepts based on the path between these two concepts in a semantic network. The term *concept* in Wu and Palmer's paper is equivalent to the term *sense* here. Given two concepts c_1 and c_2, the path length between c_1 and c_2 is defined as the number of nodes on the path from c_1 and c_2. Suppose c_3 is the least common superconcept of c_1 and c_2; then the concept similarity between c_1 and c_2 is:

$$ConSim(c_1, c_2) = \frac{2 \times PathLength(c_3, root)}{PathLength(c_1, c_3) + PathLength(c_2, c_3) + 2 \times PathLength(c_3, root)}$$

where root is the root concept in the semantic network. The semantic distance can be calculated as:

$$ConDistance(c_1, c_2) = 1 - ConSim(c_1, c_2)$$

7.3.2 Node-Based (Information-Based) Methods

Resnik's semantic similarity measure is based on the information content of each node in the semantic network [69, 70]. For a concept *c*, *p(c)* is the probability of encountering an instance of concept *c*. The information content of concept *c* is defined as –log(p(c)).

$$IC(c) = -log(p(c))$$

For two concepts, c1 and c2, the semantic similarity is the information content of their least common super-concept:

$$Sim(c_1, c_2) = -log(p(lso(c_1, c_2)))$$

where $lso(c_1, c_2)$ is the lowest super-concept of c_1 and c_2.

There are many different methods used to calculate $p(c)$ based on a corpus. The Brown Corpus of American English was used by Resnik [69]. In this corpus, $words(c)$ is defined as the set of words whose senses are subsumed by concept c in the semantic network. Then the occurrence frequency of concept c is:

$$freq(c) = \sum_{n \in words(c)} count(n)$$

The information content of concept c is:

$$p(c) = \frac{freq(c)}{N}$$

where N is the total number nouns in the corpus and also included in WordNet (Resnik's similarity measure is used for calculating the semantic similarity between two nouns).

Lin attempted to define a universal and theoretically justified similarity measure based on information theory in [51]. Lin proposed a similarity theorem that the similarity between two objects A and B should be the ratio between the information content associated to describe the commonality in them and the information content needed to fully describe all of them,

$$IT\_Sim(A, B) = \frac{IC(common(A, B))}{IC(descripton(A, B))}$$

When this theorem is applied for semantic similarity, we can get the following semantic similarity formula for two concepts:

$$Sim(c_1, c_2) = \frac{IC(lso(c_1, c_2))}{IC(c_1) + IC(c_2)} = \frac{2 \times \log(p(lso(c_1, c_2)))}{\log(p(c_1)) + \log(p(c_2))}$$

where $lso(c_1, c_2)$ is the lowest super-concept of c_1 and c_2.

7.3.3 Combined Methods

Jiang and Conrath proposed a method to combine an edge-based method and a node-based method [39]. Jiang and Conrath's method included several factors, such as local density, node depth, node information content, and link type. Given a concept node c and its parent node p, the weight of the edge between them is:

$$wt(c, p) = \left(\beta + (1-\beta)\frac{\overline{E}}{E(p)}\right)\left(\frac{d(p)+1}{d(p)}\right)^{\alpha}[IC(c) - IC(p)]T(c, p)$$

The first part of this formula is the local density factor. $E(p)$ is the number of children of node p, which reflects the local density. $\overline{E}$ is the average local density over the entire hierarchy network. The second part is node depth factor. $d(p)$ is the depth of node p in the hierarchy. The third part is node information content. $IC(c)$ is the information content of node c and $IC(p)$ is the information content of node p. $T(c, p)$ is the coefficient for a type of link. Jiang and Conrath [39] set this coefficient for the IS-A link to 1. The parameters α and β are used to control the contribution of density factor and depth factor.

Given this weight for each edge, the semantic distance between two concepts c_1 and c_2 is:

$$SemanticDist(c_1, c_2) = \sum_{c \in \{path(c1-c2)-LSuper(c1,c2)\}} wt(c, parent(c))$$

where $LSuper(c_1, c_2)$ is the lowest super-ordinate of c_1 and c_2. *LSuper* is equivalent to the *lso* in Resnik's method.

7.3.4 Other Methods

Banerjee and Pedersen [7, 64] proposed a simple method using the gloss of each sense in WordNet. The semantic relatedness between two synsets is measured by the gloss overlaps between them. Patwardhan [63] proposed a measure based on context vectors in his thesis.

7.3.5 WordNet::Similarity Package

WordNet::Similarity[22] is a perl software package which consists of several sub-modules to implement different semantic relatedness measures [65]. Table 7.1 lists these modules, their references, and their types.

Table 7.1 Perl Modules Included in WordNet::Similarity Package

Module Name	Reference	Type
WordNet::Similairy::path	Counting the nodes in the path	Edge-based
WordNet::Similairy::lch	Leacock and Chodorow(1998) [47]	Edge-based
WordNet::Similairy::wup	Wu and Palmer(1994) [90]	Edge-based
WordNet::Similairy::res	Resnik (1995) [69, 70]	Information content-based
WordNet::Similairy::lin	Lin (1998) [51]	Information content-based
WordNet::Similairy::jcn	Jiang and Conrath (1997) [39]	Edge and information content
WordNet::Similairy::lesk	Banerjee and Pedersen (2002) [7, 64]	Using gloss
WordNet::Similairy::random	Using a random measure	

[22] WordNet::Similarity package can be downloaded at http://search.cpan.org/dist/WordNet-Similarity/

7.4 Experimental Results and Analysis

A method which is similar to Gomes et al.'s method was used in our experiments [29]. Since this method is a context-words-based sense disambiguation method, the most important component of this sense disambiguation method is the semantic relatedness measure. The WordNet semantic network is used in our system. Eight different semantic relatedness measures implemented in Pedersen's WordNet::Similarity package were evaluated in our experiments.

Sense representation is another issue in this experiment. In WordNet, each word contains multiple senses and each sense has a sense number. One possible solution is using this sense number to represent this sense explicitly. One example is 'course#n#1' which represents the first sense of the noun 'course'. This method is helpful for distinguishing the different senses of a word. But for the same sense of different words, this method cannot merge them. The concept of synset in WordNet can be used to overcome this problem. In WordNet, each synset has a unique offset in the database. This offset can be used as the unique ID for this sense. All the synonyms in this synset share the same offset. For example, the offset of the first sense of the noun 'course' is '00831838'. The offset representation format is helpful for both the synonym problem and the polysemous problem. In our experiments, the offset of synset is used as the representation of a sense. Table 7.2 lists the detailed experimental results for each data set. For each different semantic relatedness measure, the F-measure and entropy results are provided. In this table, the bold font indicates this result is better than using the

original single words. The italic bold font means this result is the best one for this data set.

Table 7.2 Experimental Results of Using Word Sense

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Original	0.6900	0.7420	***0.6967***	0.6445	0.6973	0.7755	0.6944	0.6436
Jcn	**0.7161**	**0.7929**	0.6450	0.5652	**0.7858**	0.6649	**0.7969**	0.6298
Lch	**0.7663**	**0.8009**	0.6629	**0.6709**	**0.7765**	0.7054	**0.7709**	**0.6877**
Path	**0.7845**	**0.8009**	0.6748	**0.6709**	**0.7637**	0.7039	**0.6967**	***0.7436***
Wup	***0.7848***	**0.7693**	0.6546	0.5889	0.6610	0.6949	**0.7888**	0.6110
Res	0.5997	0.6851	0.6526	0.6080	***0.7946***	0.7047	0.6888	**0.6575**
Lin	**0.7006**	***0.8068***	0.6079	0.5693	**0.7706**	0.7121	***0.7979***	**0.6867**
Lesk	**0.7840**	**0.7769**	0.6653	***0.6717***	**0.7411**	***0.7836***	**0.7477**	0.5748
Random	0.6861	0.7140	0.5175	0.6089	0.5801	0.7180	0.7674	0.6180
	Entropy							
Original	0.7440	0.5129	***0.8033***	***0.7901***	0.7516	0.5985	0.7686	0.9630
Jcn	**0.6458**	0.5195	0.8204	1.0376	***0.5070***	0.8285	***0.5431***	**0.9280**
Lch	**0.6153**	**0.4884**	0.8258	0.8085	**0.5687**	0.7555	**0.6092**	**0.8109**
Path	**0.5629**	**0.4884**	0.8100	0.8085	**0.6155**	0.7738	0.7700	***0.7006***
Wup	***0.5338***	0.5807	0.8212	0.9859	0.8446	0.7972	**0.5741**	1.0050
Res	0.9313	0.6621	0.8266	0.9161	**0.5303**	0.7669	0.8055	**0.9444**
Lin	**0.7105**	***0.4525***	0.9055	0.9749	**0.6127**	0.6950	**0.5447**	**0.8319**
Lesk	**0.5508**	0.5634	0.8333	0.7984	**0.6432**	***0.5528***	**0.7065**	1.1348
Random	**0.7430**	0.6003	1.1266	0.9784	0.9989	0.7253	**0.6191**	**0.9472**

From the results listed in the table 7.2, we see that different semantic relatedness measures result in different performance. Consider the F-measures first. The *Random* method is the worst one since it decreases the F-measure for all datasets. *Lch*, *Path*, *Lin*, and *Lesk* measures are the best performers. They achieve better F-measures than the original word form for five/six datasets within eight datasets. *Jcn*, *Wup* and *Res* measures do not get better performance for most of the datasets. But for some particular datasets, they still get the best F-measure (*Wup* measure for dataset 1 and *Res* measure for dataset 5). The *Lesk* measure is the best measure according to the F-measure results. It improves the performance on six datasets, and two of them are the best F-measure for that dataset.

Our results indicate that, except for the random semantic relatedness measure, all of the measures have advantages for some particular datasets.

Now consider another aspect of the performances on different datasets. Within the eight datasets, dataset 1, 2, 5, and 7 are improved with our sense disambiguation method. Within eight different semantic relatedness measures, six of them improve the final F-measure on these datasets. Datasets 3 and 6 are the most difficult ones. Our WSD method cannot improve the performance on dataset 3 at all. The best result for this dataset is achieved by using the original words. For dataset 6, only the *Lesk* measure gets a slightly higher F-measure. Datasets 4 and 8 get intermediate results. About half of our methods increase the F-measure while the other half decrease it.

The entropy results for the experiments are similar to those for the F-measure. Within eight different semantic relatedness measures, the *Jcn*, *Lch*, *Path*, *Lin*, and *Lesk* measures got better entropy than using the original words for most datasets. Just as with F-measure, the *Wup* and *Res* measures improve the entropy for only two datasets. Different from the results with F-measure, the *Random* method also improved the entropy slightly for three datasets. Within eight different datasets, the performance on datasets 1, 5, 7, and 8 was improved by six/seven different semantic relatedness measures. The performance on datasets 3, 4, and 6 were not good. Using the original words got the best entropy on datasets 3 and 4; only the *Lesk* measure decreased the entropy on dataset 6.

Table 7.3 Wilcoxon Signed-Rank Test of F-measure for Using Senses

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
Original	0.69	0.742	***0.6967***	0.6445	0.6973	0.7755	0.6944	0.6436
Jcn	**0.7161**	**0.7929**	0.645	0.5652	**0.7858**	0.6649	**0.7969**	0.6298
Difference	0.0261	0.0509	-0.0517	-0.0793	0.0885	-0.1106	0.1025	-0.0138
Rank	2	3	-4	-5	6	-8	7	-1
W+	18	W-	18	W	18	p-value	**Cannot reject H_0**	
Lch	**0.7663**	**0.8009**	0.6629	**0.6709**	**0.7765**	0.7054	**0.7709**	**0.6877**
Difference	0.0763	0.0589	-0.0338	0.0264	0.0792	-0.0701	0.0765	0.0441
Rank	6	4	-2	1	8	-5	7	3
W+	29	W-	7	W	7	p-value	**Cannot reject H_0**	
Path	**0.7845**	**0.8009**	0.6748	**0.6709**	**0.7637**	0.7039	**0.6967**	***0.7436***
Difference	0.0945	0.0589	-0.0219	0.0264	0.0664	-0.0716	0.0023	0.1
Rank	7	4	-2	3	5	-6	1	*8*
W+	28	W-	8	W	8	p-value	**Cannot reject H_0**	
Wup	***0.7848***	**0.7693**	0.6546	0.5889	0.661	0.6949	**0.7888**	0.611
Difference	0.0948	0.0273	-0.0421	-0.0556	-0.0363	-0.0806	0.0944	-0.0326
Rank	8	1	-4	-5	-3	6	7	-2
W+	22	W-	14	W	14	p-value	**Cannot reject H_0**	
Res	0.5997	0.6851	0.6526	0.608	***0.7946***	0.7047	0.6888	**0.6575**
Difference	-0.0903	-0.0569	-0.0441	-0.0365	0.0973	-0.0708	-0.0056	0.0139
Rank	-7	-5	-4	-3	*8*	-6	-1	2
W+	10	W-	26	W	10	p-value	**Cannot reject H_0**	
Lin	**0.7006**	***0.8068***	0.6079	0.5693	**0.7706**	0.7121	***0.7979***	**0.6867**
Difference	0.0106	0.0648	-0.0888	-0.0752	0.0733	-0.0634	0.1035	0.0431
Rank	1	4	-7	-6	5	-3	8	2
W+	20	W-	16	W	16	p-value	**Cannot reject H_0**	
Lesk	**0.784**	**0.7769**	0.6653	***0.6717***	**0.7411**	***0.7836***	**0.7477**	0.5748
Difference	0.094	0.0349	-0.0314	0.0272	0.0438	0.0081	0.0533	-0.0688
Rank	8	4	-3	2	5	1	6	-7
W+	26	W-	10	W	10	p-value	**Cannot reject H_0**	
Random	0.6861	0.714	0.5175	0.6089	0.5801	0.718	0.7674	0.618
Difference	-0.0039	-0.028	-0.1792	-0.0356	-0.1172	-0.0575	0.073	-0.0256
Rank	-1	-3	-8	-4	-7	-5	-6	-2
W+	0	W-	-36	W	0	p-value	=0.005	

Table 7.4 Wilcoxon Signed-Rank Test of Entropy for Using Senses

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
Original	0.744	0.5129	***0.8033***	***0.7901***	0.7516	0.5985	0.7686	0.963
Jcn	**0.6458**	0.5195	0.8204	1.0376	***0.507***	0.8285	***0.5431***	**0.928**
Difference	-0.0982	0.0066	0.0171	0.2475	-0.2446	0.23	-0.2255	-0.035
Rank	-4	1	2	8	-7	6	-5	-3
W+	17	W-	19	W	17	p-value	**Cannot reject H_0**	
Lch	**0.6153**	**0.4884**	0.8258	0.8085	**0.5687**	0.7555	**0.6092**	**0.8109**
Difference	-0.1287	-0.0245	0.0225	0.0184	-0.1829	0.157	-0.1594	-0.1521
Rank	-4	-3	2	1	-8	6	-7	-5
W+	9	W-	27	W	9	p-value	**Cannot reject H_0**	
Path	**0.5629**	**0.4884**	0.81	0.8085	**0.6155**	0.7738	0.77	***0.7006***
Difference	-0.1811	-0.0245	0.0067	0.0184	-0.1361	0.1753	0.0014	-0.2624
Rank	-7	-4	2	3	-5	6	1	-8
W+	12	W-	24	W	12	p-value	**Cannot reject H_0**	
Wup	***0.5338***	0.5807	0.8212	0.9859	0.8446	0.7972	**0.5741**	1.005
Difference	-0.2102	0.0678	0.0179	0.1958	0.093	0.1987	-0.1945	0.042
Rank	-8	3	1	6	4	7	-5	2
W+	23	W-	13	W	13	p-value	**Cannot reject H_0**	
Res	0.9313	0.6621	0.8266	0.9161	**0.5303**	0.7669	0.8055	**0.9444**
Difference	0.1873	0.1492	0.0233	0.126	-0.2213	0.1684	0.0369	-0.0186
Rank	7	5	2	4	-8	6	3	-1
W+	27	W-	9	W	9	p-value	**Cannot reject H_0**	
Lin	**0.7105**	***0.4525***	0.9055	0.9749	**0.6127**	0.695	**0.5447**	**0.8319**
Difference	-0.0335	-0.0604	0.1022	0.1848	-0.1389	0.0965	-0.2239	-0.1311
Rank	-1	-2	4	7	-6	3	-8	-5
W+	14	W-	22	W	14	p-value	**Cannot reject H_0**	
Lesk	**0.5508**	0.5634	0.8333	0.7984	**0.6432**	***0.5528***	**0.7065**	1.1348
Difference	-0.1932	0.0505	0.03	0.0083	-0.1084	-0.0457	-0.0621	0.1718
Rank	-8	4	2	1	-6	-3	-5	7
W+	14	W-	22	W	14	p-value	**Cannot reject H_0**	
Random	**0.743**	0.6003	1.1266	0.9784	0.9989	0.7253	**0.6191**	**0.9472**
Difference	-0.001	0.0874	0.3233	0.1883	0.2473	0.1268	-0.1495	-0.0158
Rank	-1	3	8	6	7	4	-5	-2
W+	28	W-	8	W	8	p-value	**Cannot reject H_0**	

According to our analysis above, we found that using the sense disambiguation method to find the sense of a word and using the senses as the features to represent a document improved our document clustering system for most of our datasets. But this improvement is data-related. In both F-measure and entropy, an obvious improvement is detected for five datasets (1, 2, 5, 7, and 8) within eight datasets. But for the other datasets (3, 4, and 6), using the original words was still a better choice.

Table 7.3 and 7.4 show the Wilcoxon test our experimental results. The F-measure and entropy of using the original single words was compared with that of using each different semantic relatedness measure. According to table 4.2, if W is equal to or smaller than 6, then we can say that using the word sense disambiguation method improved the system performance. But in tables 7.3 and 7.4, a high W value was calculated for each semantic relatedness measure. The *Lch* and *Path* measures had relatively smaller W values, but they were still not small enough to reject the null hypothesis. These results indicate that, although using the sense disambiguation method improved the performance of our clustering system for some data sets, our current evidence was still not sufficient to conclude that using sense is significantly better than using original single words.

7.5 Summary

In this chapter, we investigated using the sense of a word to replace the original word form to solve the synonym problem and the polysemy problem in text data mining. The sense disambiguation method used in our experiments is based on the semantic relatedness among the senses. Eight different semantic relatedness measures implemented

in WordNet:Similarity packages were used in our experiments. From our experimental results, we found that using the senses of words improved the clustering performance for most of our datasets. But the statistical analysis performed on our results indicated that this improvement is not significant.

CHAPTER VIII

USING SYNTACTIC INFORMATION

8.1 Introduction

The purpose of syntactic analysis is to uncover the syntactic structure of a sentence and identify syntactic information from it. Traditionally, for information retrieval and text data mining tasks, the syntactic information is ignored and a document is simply considered a bag of independent words. In this chapter, we investigate the use of syntactic information to identify the key words from a sentence and use them to construct the feature vector to improve the performance of a document clustering system. Our results demonstrate that syntactic information is helpful under most conditions.

The rest of this chapter is organized as follows: In section 8.2, some related work about using syntactic information for information retrieval is presented. Section 8.3 lists the basic steps of syntactic analysis and introduces some related software packages. We present our experimental results and analysis in section 8.4. Section 6 is a brief summary of this chapter.

8.2 Related Work

Smeaton, O'Donnell, and Kelledy investigated using syntactic analysis to improve the performance of an information retrieval system [81]. A language analyzer, ENGCG, which was developed at the research unit for computational linguistics at the University

of Helsinki, was used to identify the syntactic structure of user queries and document texts. Each document or query is separated into sentences and clauses. ENGCG constructs a syntactic tree structure (TSA) for each clause. Then each document is represented as a list of TSAs and each query is also represented as one or more TSAs. The similarity between a candidate document and a query is measured by pairwise matching between each document TSA and each query TSA. Regretfully this method did not improve the retrieval accuracy in Smeaton, O'Donnell, and Kelledy's experiments.

Mittendorfer and Winiwarter presented a method to use the syntactic structure of a query for information retrieval [60]. In this algorithm, each query is converted into a syntactic tree or lattice by a Linear Grammar Parser (LGP). The connectedness between two words within this tree is measured based on the path between them. The final score of a candidate document is a combination of positional score factor, structure score factor, and basic word score. The final experimental results showed that this method works better in some cases but worse in others.

Choudhary and Bhattacharyya used the Universal Networking Language (UNL)[23] to represent each sentence in a document as a graph [17]. Each word is represented as a node in this graph and arcs between nodes reflect the syntactic relationship among words. A package, EnConverter, contained in the UNL development set, can be used to covert a sentence into UNL format. The UNL graph is used for document clustering in two ways. The first strategy is using the UNL graph links to construct the feature vector. The feature value is the number of links attached to that node. It is assumed that the node with more

[23] The website of UNL is located at http://www.unlc.undl.org/unlsys/ds.html

arcs connected to it is more important. The second strategy they used was UNL relation labels, which consider the labels of the links in the graph. An n*n matrix is constructed to count the links between each pair of nodes, where n is the size of the corpus. Then all the columns of the matrix are added up to form a single dimension vector to represent the document. The clustering algorithm used in their experiments is Self Organizing Maps (SOM). The experimental results demonstrate the high accuracy of the new document representation methods.

8.3 Syntactic Analysis

A syntactic parser is a complex syntactic analysis tool. The purpose of such a parser is to identify the syntactic structure of a sentence and generate a syntactic tree structure for it. The core of a syntactic parser is a set of syntactic rules and knowledge about that language called a syntactic grammar. Sleator and Temperley proposed a syntactic parser called the Link Grammar Parser[24] for English in 1991 [80]. This parser applied link grammar, an original theory of English syntax, as the syntactic grammar. A robust syntactic analysis tool, some rare and idiomatic syntactic constructions are also contained in the Link Grammar Parser. Some intelligent strategies are also provided for unknown words and symbols. Michael Collins described his parser in 1999 [18]. In the Collins parser[25], some statistical methods are applied to a training set to construct a function between a sentence and a syntactic tree structure. Another parser, LoPar[26], was

[24] Link Grammar Parser can be downloaded at http://www.link.cs.cmu.edu/link/.
[25] Collins parser can be downloaded at http://www.ai.mit.edu/people/mcollins/.
[26] LoPar can be downloaded at http://www.ims.uni-stuttgart.de/projekte/gramotron/resources.html.

proposed by Schmid [77]. Head-Lexicalised Probabilistic Context-Free Grammar (HL-PCFG), which was trained on the British National Corpus (BNC), is used in LoPar.

```
PROB 1052 -44.9285 0
TOP -44.9285 S -38.1121 NP -0.00613073 NPB -0.00172718 PRP 0 I
    VP -37.7751 VBP 0 'm
      NP -23.3464 NPB -2.06808 DT 0 the
           NN 0 teacher
        PP -9.07698 IN 0 of
          NP -2.54743 NPB -2.29342 DT 0 the
               NN 0 course
        NP -1.15926 NPB -0.869919 NNP 0 Algorithms
```

Figure 8.1 Raw Output Format of Collins' Parser

```
(TOP~'m~1~1 (S~'m~2~2 (NPB~I~1~1 I/PRP ) (VP~'m~2~1 'm/VBP (NP~teacher~3~1
(NPB~teacher~2~2 the/DT teacher/NN ) (PP~of~2~1 of/IN (NPB~course~2~2 the/DT course/NN
``/PUNC`` ) ) (NPB~Algorithms~1~1 Algorithms/NNP "/PUNC" ./PUNC. ) ) ) ) )
```

Figure 8.2 Full Parser Output Format of Collins' Parser

In our experiment, Collins' syntactic parser was used to perform the syntactic analysis. The input for Collins' parser is the sentences with part-of-speech tags which is generated by the data preprocessing procedures introduced in section 4.4. In the next step, the sentence with part-of-speech tags is sent to a syntactic parser to find the syntactic structure. In this step, a syntactic tree is constructed for each sentence which reflects the syntactic relations among the tokens in that sentence. Collins' parser is used in our system for this purpose. The results of Collins's parser have two formats: raw output format and full parse output format. In raw output, the sentence is represented with a dendrogram with each token in each line. The raw output format of a sample sentence is given in Figure 8.1. In the full parse output format, the syntactic structure of the sentence

is represented with a string. The hierarchical relationships among the tokens are represented with brackets. The full output format of the sample sentence is given in Figure 8.2. Detailed information about these two formats is explained in the dissertation by Collins [18]. The logical representation of the syntactic tree for the sample sentence is given in Figure 8.3.

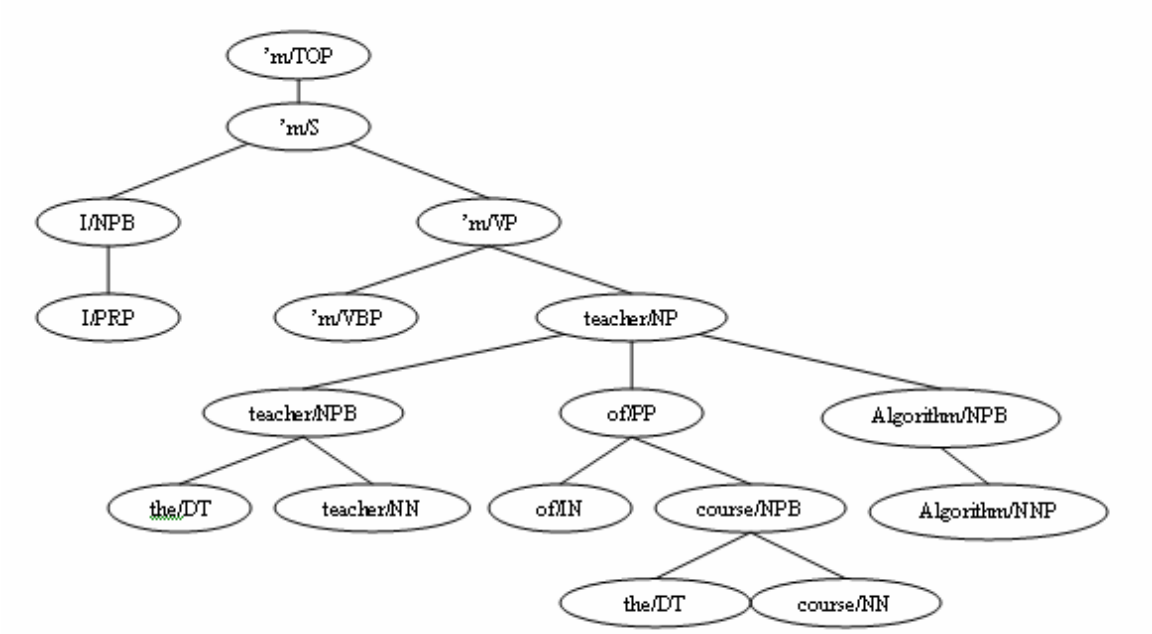

Figure 8.3 An Example of Syntactic Tree

8.4 Experimental Results and Analysis

8.4.1 Using Syntactic Information

In section 8.2, different researchers tried to use such a syntactic tree structure in different ways to improve the system. Here, we tried to use this tree to find words within the sentence which were more important in terms of reflecting the meaning of this sentence. From the sample syntactic tree in section 8.3, all the noun phrases (NP) and

verb phrases (VP) were parsed. A kind of special noun phrase, a non-recursive noun phrase, was also identified by Collins' parser. Collins defined non-recursive noun phrases as NPs that "do not directly dominate an NP themselves, unless the dominated NP is a possessive NP" [18]. The non-recursive NPs, also referred to as 'baseNPs', were tagged with the label 'NPB' in the syntactic tree. In our sample sentence, there were three NPB words, 'teacher/NPB', 'course/NPB', and 'Algorithm/NPB' and these words were the key words in this sentence. We assumed that NPB words are more important than the other words for reflecting the meaning of a sentence. In order to verify our assumption, we constructed the feature vector with single words and NPB words separately and compared their performance for document clustering. The results of this experiment are listed in table 8.1.

Table 8.1 Comparison of Different Feature Vector Methods

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single	0.69	**0.742**	0.6967	0.6445	0.6973	0.7755	0.6944	**0.6436**
NPB	**0.7971**	0.7387	**0.7071**	**0.7024**	**0.7811**	**0.8012**	**0.7288**	0.6052
	Entropy							
Single	0.744	0.5129	0.8033	0.7901	0.7516	0.5985	0.7686	**0.963**
NPB	**0.5111**	**0.4898**	**0.6557**	**0.6979**	**0.5941**	**0.5905**	**0.6796**	0.9799

Within eight datasets, NPB words achieved the best performance in six datasets for F-measure and seven datasets for entropy. The original single words got the best results in two datasets for F-measure and only one dataset for entropy. This result demonstrated that using the NPB words in the syntactic tree can improve the performance of our clustering system in most conditions.

The Wilcoxon signed rank test was applied again to evaluate the experimental results (see table 8.2). For F-measure, we got a W value equal to 6, which indicates we can reject the null hypothesis at a significance level of 0.05. For entropy, a better p-value is achieved. The probability that we incorrectly reject the null hypothesis is only 1%. Given this evidence, we can conclude that using NPB words significantly improved the performance of our clustering system.

Table 8.2 Wilcoxon Signed-Rank Test for Using Original Words and NPB Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single Words	0.69	**0.742**	0.6967	0.6445	0.6973	0.7755	0.6944	**0.6436**
Compound Words	**0.7971**	0.7387	**0.7071**	**0.7024**	**0.7811**	**0.8012**	**0.7288**	0.6052
Difference	0.1071	-0.0033	0.0104	0.0579	0.0838	0.0257	0.0344	-0.0384
Rank	8	-1	2	6	7	3	4	-5
W^+	30							
W^-	6							
$W=\min(W^+,W^-)$	6							
p-value	= 0.05, H_0 can be rejected.							
	Entropy							
Single Words	0.744	0.5129	0.8033	0.7901	0.7516	0.5985	0.7686	**0.963**
Compound Words	**0.5111**	**0.4898**	**0.6557**	**0.6979**	**0.5941**	**0.5905**	**0.6796**	0.9799
Difference	-0.2329	-0.0231	-0.1476	-0.0922	-0.1575	-0.008	-0.089	0.0169
Rank	-8	-3	-6	-5	--7	-1	-4	2
W^+	2							
W^-	34							
$W=\min(W^+,W^-)$	2							
p-value	= 0.01, H_0 can be rejected.							

8.4.2 Combining Semantic and Syntactic Information

In chapter 7, semantic information was used to improve the performance of our clustering system instead of using the original single words. In this experiment, the word sense disambiguation methods were applied to the NPB words and the senses of NPB words were used as the features. Eight different semantic relatedness measures introduced in chapter 7 were used again here. The results are list in table 8.3.

Table 8.3 Experimental Results of Using Sense of NPB Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
	F-measure							
Single	0.69	0.742	0.6967	0.6445	0.6973	0.7755	0.6944	0.6436
NPB-Jcn	0.7	0.7773	0.7403	0.7009	0.7184	0.8934	0.739	0.685
NPB-Lch	**0.7925**	0.7611	0.7321	0.7237	0.744	0.8899	0.7435	0.6366
NPB-Path	0.7857	0.7611	0.7321	0.7237	0.744	0.8899	0.7435	0.6366
NPB-Wup	0.6815	0.7718	0.7011	0.725	0.7524	0.895	0.6781	0.6802
NPB–Res	0.6875	0.7714	0.7324	0.6956	0.7531	0.895	0.7435	**0.6948**
NPB-Lin	0.7001	**0.7949**	**0.7693**	0.6776	0.7093	**0.9204**	**0.7435**	0.6788
NPB–Lesk	0.7509	0.7782	0.7294	**0.732**	**0.7935**	0.8308	0.6795	0.6805
NPB–Random	0.7175	0.6794	0.756	0.7039	0.6832	0.8135	0.6791	0.6029
	Entropy							
Single	0.744	0.5129	0.8033	0.7901	0.7516	0.5985	0.7686	0.963
NPB-Jcn	0.6489	0.4551	0.6042	0.6927	0.7152	0.3217	0.6892	0.8049
NPB-Lch	**0.482**	0.4994	0.6443	0.6949	0.6763	0.3101	0.6702	0.9042
NPB-Path	0.4904	0.4994	0.6443	0.6949	0.6743	0.3101	0.6702	0.9042
NPB-Wup	0.713	0.4776	0.7194	0.6823	0.6926	0.2908	0.8008	0.8136
NPB–Res	0.7226	0.4685	0.5833	0.7	0.6237	0.2908	0.6702	**0.7661**
NPB-Lin	0.6674	**0.426**	**0.5354**	0.7069	0.7274	**0.2603**	**0.6702**	0.8144
NPB–Lesk	0.5791	0.4567	0.6066	**0.5837**	**0.5946**	0.4434	0.7999	0.8074
NPB–Random	0.6228	0.6866	0.5871	0.7067	0.7676	0.4551	0.8162	0.9904

From table 8.3, we find that encouraging improvements are achieved within this experiment. Except for the random measure, all of the measures improved both F-measure and entropy for almost all of the datasets. Most of the best F-measures and entropy were achieved by the *Lin* and *Lesk* measures. In chapter 6, we found that just using the sense of words to replace the original words improved the performance with some data sets but not all of them, and the improvements were not statistically significant. But when we combined the semantic information with the syntactic information, a better improvement was observed. There are two possible reasons for this. The first one is the importance of syntactic information. In the previous experiment, just using the syntactic information resulted in a significant improvement. In this experiment, the advantages of syntactic information played an important role again. The second

possible reason is the correctness of our sense disambiguation method. The WSD adapted in our system is not a perfect one. Using the syntactic information decreased the size of the feature vector and a lot of unimportant terms were discarded. This also avoided a lot of mistakes incurred in the sense disambiguation procedure.

The Wilcoxon signed rank test for this experiment is presented in table 8.4 and table 8.5. Table 8.4 is the comparison of the F-measures and Table 8.5 is the comparison of the entropy measures. The performance of using the original single words was compared with that of using the sense of NPB words one by one. As we expected, except for the random measure, all the measures had significantly different F-measures and entropies from the original single words. For most of them, the W value is zero, which indicates that the probability for incorrect rejection is 0.5%. This p-value demonstrates a strong significance level.

Table 8.4 Wilcoxon Signed-Rank Test of F-measure for Using Senses of NPB Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
Original	0.69	0.742	0.6967	0.6445	0.6973	0.7755	0.6944	0.6436
Jcn	**0.7**	**0.7773**	**0.7403**	**0.7009**	**0.7184**	**0.8934**	**0.739**	**0.685**
Difference	0.01	0.0353	0.0436	0.0564	0.0211	0.1179	0.0446	0.0414
Rank	1	3	5	7	2	8	6	4
W+	0	W-	36	W	0	p-value	**= 0.005**	
Lch	**0.7925**	**0.7611**	**0.7321**	**0.7237**	**0.744**	**0.8899**	**0.7435**	0.6366
Difference	0.1025	0.0191	0.0354	0.0792	0.0467	0.1144	0.0491	-0.007
Rank	7	2	3	6	4	8	5	-1
W+	35	W-	1	W	1	p-value	**0.005< p < 0.01**	
Path	**0.7857**	**0.7611**	**0.7321**	**0.7237**	**0.744**	**0.8899**	**0.7435**	0.6366
Difference	0.0957	0.0191	0.0354	0.0792	0.0467	0.1144	0.0491	-0.007
Rank	7	2	3	6	4	8	5	-1
W+	35	W-	1	W	1	p-value	**0.005< p < 0.01**	
Wup	0.6815	**0.7718**	**0.7011**	**0.725**	**0.7524**	**0.895**	0.6781	**0.6802**
Difference	-0.0085	0.0298	0.0044	0.0805	0.0551	0.1195	-0.0163	0.0366
Rank	-2	4	1	7	6	8	-3	5
W+	31	W-	5	W	5	p-value	**0.025 < p < 0.05**	
Res	0.6875	**0.7714**	**0.7324**	**0.6956**	**0.7531**	**0.895**	**0.7435**	**0.6948**
Difference	-0.0025	0.0294	0.0357	0.0511	0.0558	0.1195	0.0491	0.0512
Rank	-1	2	3	5	7	8	4	6
W+	35	W-	1	W	1	p-value	**0.005< p < 0.01**	
Lin	**0.7001**	**0.7949**	**0.7693**	**0.6776**	**0.7093**	**0.9204**	**0.7435**	**0.6788**
Difference	0.0101	0.0529	0.0726	0.0331	0.012	0.1449	0.0491	0.0352
Rank	1	6	7	3	2	8	5	4
W+	36	W-	0	W	0	p-value	**= 0.005**	
Lesk	**0.7509**	**0.7782**	**0.7294**	**0.732**	**0.7935**	**0.8308**	0.6795	**0.6805**
Difference	0.0609	0.0362	0.0327	0.0875	0.0962	0.0553	-0.0149	0.0369
Rank	6	3	2	7	8	5	-1	4
W+	35	W-	1	W	1	p-value	**0.005< p < 0.01**	
Random	**0.7175**	0.6794	**0.756**	**0.7039**	0.6832	**0.8135**	0.6791	0.6029
Difference	0.0275	-0.0626	0.0593	0.0594	-0.0141	0.038	-0.0153	-0.0407
Rank	3	-8	6	7	-1	4	-2	5
W+	25	W-	11	W	11	p-value	**Cannot reject H_0**	

Table 8.5 Wilcoxon Signed-Rank Test of Entropy for Using Senses of NPB Words

Data Set	SDS 1	SDS 2	SDS 3	SDS 4	SDS 5	SDS 6	SDS 7	SDS 8
Original	0.744	0.5129	0.8033	0.7901	0.7516	0.5985	0.7686	0.963
Jcn	**0.6489**	**0.4551**	**0.6042**	**0.6927**	**0.7152**	**0.3217**	**0.6892**	**0.8049**
Difference	-0.0951	-0.0578	-0.1991	-0.0974	-0.0364	-0.2768	-0.0794	-0.1581
Rank	-4	-2	-7	-5	-1	-8	-3	-6
W+	0	W-	36	W	0	p-value	**= 0.005**	
Lch	**0.482**	**0.4994**	**0.6443**	**0.6949**	**0.6763**	**0.3101**	**0.6702**	**0.9042**
Difference	-0.262	-0.0135	-0.159	-0.0952	-0.0753	-0.2884	-0.0984	-0.0588
Rank	-7	-1	-6	-4	-3	-8	-5	-2
W+	0	W-	36	W	0	p-value	**= 0.005**	
Path	**0.4904**	**0.4994**	**0.6443**	**0.6949**	**0.6743**	**0.3101**	**0.6702**	**0.9042**
Difference	-0.2536	-0.0135	-0.159	-0.0952	-0.0773	-0.2884	-0.0984	-0.0588
Rank	-7	-1	-6	-4	-3	-8	-5	-2
W+	0	W-	36	W	0	p-value	**= 0.005**	
Wup	**0.713**	**0.4776**	**0.7194**	**0.6823**	**0.6926**	**0.2908**	0.8008	**0.8136**
Difference	-0.031	-0.0353	-0.0839	-0.1078	-0.059	-0.3077	0.0322	-0.1494
Rank	-1	-3	-5	-6	-4	-8	2	-7
W+	2	W-	34	W	2	p-value	**= 0.01**	
Res	**0.7226**	**0.4685**	**0.5833**	**0.7**	**0.6237**	**0.2908**	**0.6702**	**0.7661**
Difference	-0.0214	-0.0444	-0.22	-0.0901	-0.1279	-0.3077	-0.0984	-0.1969
Rank	-1	-2	-7	-3	-5	-8	-4	-6
W+	0	W-	36	W	0	p-value	**= 0.005**	
Lin	**0.6674**	**0.426**	**0.5354**	**0.7069**	**0.7274**	**0.2603**	**0.6702**	**0.8144**
Difference	-0.0766	-0.0869	-0.2679	-0.0832	-0.0242	-0.3382	-0.0984	-0.1486
Rank	-2	-4	-7	-3	-1	-8	-5	-6
W+	0	W-	36	W	0	p-value	**= 0.005**	
Lesk	**0.5791**	**0.4567**	**0.6066**	**0.5837**	**0.5946**	**0.4434**	0.7999	**0.8074**
Difference	-0.1649	-0.0562	-0.1967	-0.2064	-0.157	-0.1551	0.0313	-0.1556
Rank	-6	-2	-7	-8	-5	-3	1	-4
W+	1	W-	35	W	1	p-value	**0.005< p < 0.01**	
Random	**0.6228**	0.6866	**0.5871**	**0.7067**	0.7676	**0.4551**	0.8162	0.9904
Difference	-0.1212	0.1737	-0.2162	-0.0834	0.016	-0.1434	0.0476	0.0274
Rank	-5	7	-8	-4	1	-6	3	2
W+	13	W-	23	W	13	p-value	**Cannot reject H_0**	

8.5 Summary

In this chapter, we reported our experiments of using syntactic information in sentences to improve the performance of our document clustering system. Collins' syntactic parser was used in our experiments and the NPB words identified by Collin's parser were used to construct the feature vector to represent documents. Our experimental results showed that, in most conditions, NPB words outperformed the traditional single words. A further improvement was achieved when the sense of the NPB words were used. The Wilcoxon signed rank test was used to demonstrate that the improvements achieved here are statistically significant.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

9.1 Introduction

In this chapter, a comprehensive summary of our work in this dissertation is provided. Our conclusions and contributions are listed. Some interesting and promising future work is proposed.

9.2 Summary and Conclusions

Document clustering is a research topic with a long history and for which various algorithms have been proposed. Although these algorithms are different, most of them share an important basis, the similarity measures between documents. Since the purpose of document clustering is to group similar documents together, the goodness of the similarity measure is the primary factor for the performance of a clustering algorithm. The goal of our research work in this dissertation was to identify more information from raw text and incorporate this information into the representation of the documents. Our work incorporates both semantic information and syntactic information. We expected that this would improve the precision of document similarity and further improve the performance of our document clustering system.

Our experiments can be divided briefly into three steps, identification of semantic information, identification of syntactic information, and a combination of both of them. Some work about using compound words was also studied. We implemented four different versions of clustering systems based on four different clustering algorithms: K-means method, Buckshot method, HAC method, and bisecting K-means method. According to their performance for our dataset, the HAC method was selected to evaluate the performance of our other research work. Based on the detailed experimental results presented in Chapters V, VI, VII, and VIII, we conclude:

- In the HAC clustering method, the average-link inter-clustering distance measure is significantly better than the single-link method and complete-link method.
- For different term weighting methods in the vector space model, the TFIDF method is significantly better than the binary method and TF method. The TFIDF method assigns a weight to a feature by combining its importance in a document and its distinguishability for the whole document set. An important term may be a medium frequency word instead of a high frequency word (too common) or a low frequency word (too particular).
- For large data sets, the bisecting algorithm outperforms all the other methods. But for small data sets, the HAC method gets the best performance.
- The K-means method has a performance that is similar to that of the Buckshot method for large data sets. But for small data sets, the Buckshot method is better than the K-means method.

- Just using compound words does not improve the clustering performance for our data sets. But when the compound words are combined with original single words, the combined feature set gets the better performance for most data sets. But this improvement is still not strong enough to determine which method is significantly better than the others in the statistical analysis.
- Using the word sense disambiguation algorithm to identify the correct sense of each word and using the sense as the feature to represent document can improve the F-measure and entropy of our document clustering system for most datasets. Eight different semantic relatedness measures are evaluated. Except for the random method, all of the methods had similar positive results. But this improvement was not statistically significant.
- Syntactic information is important for identifying key words from sentences. Using the base noun phrases (NPBs) as the features for the document representation outperforms the traditional use of single words in most of our datasets. The Wilcoxon signed rank test also showed that the differences between using original single words and using NPB words were statistically significant with a small p-value.
- A further improvement can be achieved when the correct sense of these NPBs is identified and used as features. Except for the random measure, all of the semantic relatedness measures achieved similar good, statistically significant performances.

9.3 Contributions

The significant contributions of this dissertation work are described below:

1. The sense of each word is distinguished according to its context and is used to to represent a document. Word sense disambiguation is another research area with a long history. We incorporated the work in WSD into information retrieval. Since our text data mining work was based on the content of a document, we expected that a list of senses can reflect the document content more precisely. Our experimental results demonstrated the correctness of our hypothesis.
2. Feature selection is an important topic for text data mining since generally the size of a feature vector is huge. It is reported in [93] that some general feature selection methods such as information gain (IG), mutual information (MI), and χ^2 did not improve the performance of a document classification system significantly. One possible reason is that all these methods are based on the word occurrence frequency. In this dissertation work, we used the syntactic tree, which is constructed from a sentence with some syntactic analysis tool, to identify the important words from flat level sentences. The syntactic tree can reflect the structure of a sentence and distinguish the central words in a sentence from the modifier words. In our experiments, the NPB words identified by Collins' parser improved the performance of our document clustering system from the original all-single words.

3. We identified the NBP words from the syntactic tree and then identified the sense of these key words to represent the document. Our approach of incorporating the semantic information with the syntactic information achieved a further improvement.
4. Using compound word instead of single words to represent the document has been studied by a lot of researchers and different results have been reported. We combined the compound words with the original single words to represent the document. From our results, we can conclude that using compound words does not improve the clustering performance for our data sets. But when the compound words are combined with single words, the combined feature set gets better performance for most data sets.
5. We constructed a document clustering system to demonstrate our methods. In order to evaluate the performance of our hypotheses, a practical clustering system was implemented. In order to select a better clustering algorithm for our system, a comparison between several widely used clustering algorithms was made. Our experimental results show that the bisecting K-means method is the best choice for large datasets. But for small datasets, the traditional hierarchical clustering method still outperforms the other methods.

9.4 Future Work

Based on our current experimental results, our future work will focus on the following four aspects:

Make use of more syntactic information. Currently only the NPB words within the syntactic trees are identified to construct the feature vector. There is a lot of other syntactic information within the syntactic tree structure. A potential source of information is the linkage among the words in the tree. With the help of these links, the words will not be independent. A comprehensive understanding of document content should consider the relationship among the words within it. We plan to utilize more syntactic information and expect to get further improvement.

Combine the syntactic information and semantic information with the compound words. In our current work, it is concluded that using the compound words and single words together can get a better performance than using each of them individually. Another attempt could be to identify the sense of these compound words and combine it with the senses of single words.

Use some other document representation methods. Currently, although some semantic and syntactic information are utilized in our document representation, we are still using the traditional vector space model. We plan to use another concept, which is called syntactic component, to represent a document. A subject or object is a kind of syntactic component. Generally a syntactic component consists of a central word and a list of modifiers. Semantic information will be incorporated into a syntactic component again to replace the word with its sense. A document will be represented with a list of syntactic components. Other similarity measures should be proposed to evaluate the similarity between two syntactic components.

Use other widely used datasets. Our current datasets are the journal abstracts we have collected. It is difficult to compare our performance with others since we are using different datasets. We had planned to use the Reuters data and newsgroups data which can be downloaded from UCI KDD Archive[27] [34]. But the problem is that all of these datasets are the articles from newsgroup and bbs board. There are a lot of grammar errors and typographical errors in the articles. These errors make the semantic analysis and syntactic analysis almost impossible. In the future, we will try to find some other well-formatted and widely used datasets.

[27] The website of UCI KDD archive is located at http://kdd.ics.uci.edu/ .

REFERENCES:

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proceedings of the ACM SIGMOD Conference,* Seattle, WA, 1998, pp. 94-105.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Database," *Proceedings of the 20th International Conference on Very Large Databases* (VLDB 94), Santiago de Chile, Chile, 1994, pp. 487-499.

[3] K. Alsabti, S. Ranka, and V. Singh, "An Efficient K-means Clustering Algorithm," *First Workshop on High-Performance Data Mining,* 1998.

[4] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify Clustering Structure," *Proceedings of the ACM SIGMOD Conference,* Philadelphia, PA, 1999, pp. 49-60.

[5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval,* Addison-Wesley-Longman, May, 1999.

[6] J. Bakus, M. F. Hussin, and M. Kamel, "A SOM-Based Document Clustering Using Phrases," *Proceedings of 9th International Conference on Neural Information Processing* (ICONIP'2002), Singapore, Nov. 2002.

[7] S. Banerjee and T. Pedersen, "An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet," *Proceedings of the Fourth International Conference on Computational Linguistics and Intelligent Text Processing (CICLING-02)*, Mexico City, 2002.

[8] F. Beil, M. Ester, and X. Xu, "Frequent Term-Based Text Clustering," *Proceeding of the 8th International Conference on Knowledge Discovery and Data Mining* (KDD 2002)*,* Edmonton, Alberta, Canada, 2002.

[9] P. Bellot and M. El-Beze, *A Clustering Method for Information Retrieval,* Technical Report IR-0199, Laboratoire d'Informatique d'Avignon, France, 1999.

[10] P. Berkhin, "Survey of Clustering Data Mining Techniques," Accrue Software, http://citeseer.nj.nec.com/berkhin02survey.html (Accessed on 12 Feb. 2004).

[11] R. Besançon, J.-C. Chappelier, M. Rajman, and A. Rozenknop, "Improving Text Representations through Probabilistic Integration of Synonymy Relations," *Proceedings of the 10th International Symposium on Applied Stochastic Models and Data Analysis* (ASMDA'2001), vol. 1, 2001, pp. 200-205.

[12] D. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum Press, New York, NY, 1981.

[13] E. Brill, "A Simple Rule-Based Part of Speech Tagger," *Proceedings of the Third Conference on Applied Natural Language Processing,* (ACL), Trento, Italy, 1992.

[14] A. Budanitsky, *Lexical Semantic Relatedness and Its Application in Natural Language Processing*, Technical Report CSRG-390, Computer Systems Research Group, University of Toronto, 1999.

[15] A. Budanitsky and G. Hirst, "Semantic Distance in WordNet: An Experimental, Application-Oriented Evaluation of Five Measures," In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA, 2001.

[16] M. F. Caropreso, S. Matwin, and F. Sebastiani, "A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization," In A.G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pp. 78-102, Hershey, USA, 2001, Idea Group Publishing.

[17] B. Choudhary and P. Bhattacharyya, "Text Clustering using Semantics," *Proceedings of 7th International Word Wide Web Conference,* Honolulu, Hawaii, USA, May, 2002.

[18] M. Collins, *Head-Driven Statistical Models for Natural Language Parsing,* Ph.D. Dissertation, University of Pennsylvania, 1999.

[19] D. Cutting, D. Karger, J. Pedersen, and J. Tukey, "Scatter/Gather: a Clusterbased Approach to Browsing Large Document Collection," *Proceedings of the 15th ACM SIGIR Conference,* Copenhagen, Denmark, 1992, pp. 318-329.

[20] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by Latent Semantic Analysis," *Joumal of the American Society of Infonation Science,* vol. 41, no. 6, 1990, pp.391-407.

[21] H. Feili, 2003, Natural Language Processing, Accessed from http://ce.sharif.edu/~nlplab/files/sem1.ppt

[22] C. Fellbaum, *WordNet: An Electronic Lexical Database*, MIT Press, 1998.

[23] D. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning,* vol. 2, 1987, pp. 139-172.

[24] K. Fragos, Y. Maistros, and C. Skourlas, "Word Sense Disambiguation using WordNet Relations," *Proceedings of the 1st Balkan Conference in Informatics*, Thessaloniki, Greece, 2003.

[25] B. C. M. Fung, *Hierarchical Document Clustering Using Frequent Itemsets,* Master Thesis, Dept. Computer Science, Simon Fraser University, Canada, 2002.

[26] J. Furnkranz, T. Mitchell, and E. Rilogg, "A Case Study in Using Linguistic Phrases for Text Categorization on WWW," *Proceedings of the 1st AAAI Workshop on Learning for Text Categorization*, 1998, pp. 5-12.

[27] A.L. Gançarski, "Using Attribute Grammars to Uniformly Represent Structured Documents - Application to Information Retrieval," *Proceedings of the Third DELOS Network of Excellence Workshop on Interoperability and Mediation in Heterogeneous Digital Libraries,* Darmstadt, Germany, September 2001, pp. 8-9.

[28] J. Gennari, P. Langley, and D. Fisher, "Models of Incremental Concept Formation," *Artificial Intelligence,* vol. 40, 1989, pp. 11-61.

[29] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. Ferreira, and C. Bento, "Noun Sense Disambiguation with WordNet for Software Design Retrieval," *Proceedings of the Sixteenth Canadian Conference on Artificial Intelligence (AI'03)*, Halifax, Canada, June-2003.

[30] K. M. Hammouda, *Web Mining: Identifying Document Structure for Web Document Clustering,* Master's Thesis, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 2002.

[31] K. Hammouda and M. Kamel, "Document Similarity Using a Phrase Indexing Graph Model," *Knowledge and Information Systems,* Springer, In Press, May 2003.

[32] J. Han and M. Kamber, *Data Mining: Concepts and Techniques,* Morgan Kaufmann Publishers, 2001.

[33] V. Hatzivassiloglou, J. L. Klavans, and E. Eskin, "Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning," *Proceedings of Empirical Methods in Natural Language Processing and Very Large Corpora* (EMNLP'99), MD, USA, 1999.

[34] S. Hettich and S. D. Bay, *The UCI KDD Archive,* Irvine, CA: University of California, Department of Information and Computer Science, 1999, http://kdd.ics.uci.edu.

[35] G. Hirst and D. St-Onge, "Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms," 1998, In C. Fellbaum, editor, WordNet: An Electronic Lexical Database, pp. 305-332, MIT Press.

[36] A. Hinneburg and D. Keim, "An Efficient Approach to Clustering Large Multimedia Databases with Noise," *Proceedings of the 4th ACM SIGKDD,* New York, NY, 1998, pp. 58-65.

[37] F.B. Holt, "Subspace Representations of Unstructured Text", *Proceedings of the 2001 IEEE International Conference on Data Mining* (IEEE ICDM-2001), San Jose, California, USA, 2001.

[38] I. Iliopoulos, A.J. Enright, and C.A. Ouzounis, "Textquest: Document Clustering of Medline Abstracts for Concept Discovery in Molecular Biology," *Proceedings of the Sixth Annual Pacific Symposium on Biocomputing* (PSB 001), 2001.

[39] J. Jiang and D. Conrath, "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," *Proceedings of International Conference on Research in Computational Linguistics*, Taiwan, 1997.

[40] T. Joachims, A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, Computer Science Technical Report CMU-CS-96-118, Carnegie Mellon University, 1996.

[41] Á. R. P. Jr and N. Ziviani, "Syntactic Similarity of Web Documents," *Proceedings of the First Latin American Web Congress* (LA-WEB 2003), 2003.

[42] D. Jurafsky and J. Martin, *Speech and Language Processing,* Prentice hall, 2000.

[43] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis,* John Wiley and Sons, New York, NY, 1990.

[44] H.-P. Kriegel, B. Seeger, R. Schneider, and N. Beckmann, "The R*-tree: an Efficient Access Method for Geographic Information Systems," *Proceedings of International Conference on Geographic Information Systems,* Ottawa, Canada, 1990.

[45] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE,* vol. 9, 1990, pp. 1464-1479.

[46] B. Larsen and C. Aone, "Fast and Effective Text Mining Using Linear-time Document Clustering", *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining,* San Diego, California, United States, 1999, pp. 16-22.

[47] C. Leacock and M. Chodorow, "Combining Local Context and WordNet Similarity for Word Sense Identification," 1998, In C. Fellbaum, editor, WordNet: An Electronic Lexical Database, pp. 265-283, MIT Press.

[48] M. Lesk, "Automatic Sense Disambiguation: How to Tell a Pine Cone from an Ice Cream Cone," *Proceedings of the 1986 SIGDOC Conference,* New York, 1986, pp. 24-26. Association of Computing Machinery.

[49] X.B. Li, S. Szpakowicz, and S. Matwin, "A Wordnet-Based Algorithm for Word Sense Disambiguation," *Proceedings of IJCAI-95*, Montral, Canada, 1995.

[50] A. Likas, N. Vlassis, and J.J. Verbeek, "The Global K-means Clustering Algorithm," *Pattern Recognition,* vol. 36, no. 2, 2003, pp. 451-461.

[51] D. Lin, "An Information-Theoretic Definition of Similarity," *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, 1998.

[52] K. Lin and R. Kondadadi, "A Word-Based Soft Clustering Algorithm for Documents," *Proceedings of 16th International Conference on Computers and Their Applications,* Mar. 2001.

[53] J.B. Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics,* vol. 11, 1968, pp. 22-31.

[54] Y.S. Maarek, R. Fagin, I.Z. Ben-Shaul, and D. Pelleg, *Ephemeral Document Clustering for Web Applications,* Technical Report RJ 10186, IBM Research, 2000.

[55] E. Marchiori, "Data Mining," *Encyclopedia of Life Support Systems,* to appear, 2000.

[56] A. Mikheev, "Part-of-Speech Guessing Rules: Learning and Evaluation," to appear in *Computational Linguistics*, http://www.ltg.ed.ac.uk/software/pos (Accessed on 12 Feb. 2004).

[57] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An On-Line Lexical Database," *International Journal of Lexicography,* vol. 3, no. 4, 1990, pp. 235-312.

[58] G. Milligan and M. Cooper, “An examination of procedures for determining the number of clusters in a data set,” *Psychometrika,* vol. 50, 1985, pp. 159-179.

[59] M. Mitra, C. Buckley, A. Singhal, and C. Cardie, “An Analysis of Statistical and Syntactic Phrases,” *Proceedings of RIAO-97, 5th International Conference "Recherche d'Information Assistee par Ordinateur"*, Montreal, CA, 1997, pp. 200-214.

[60] M. Mittendorfer and W. Winiwarter, “Exploiting Syntactic Analysis of Queries for Information Retrieval,” *Data & Knowledge Engineering*, vol. 42, no. 3, 2002.

[61] D. Mladenic and M. Grobelnik, “Word Sequence as Features in Text-learning,” *Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK-98)*, Ljubljana, Slovenia, 1998.

[62] N. J. Nilsson, *Artificial Intelligence: A New Synthesis,* Morgan Kaufmann Publishers, 1998.

[63] S. Patwardhan, *Incorporating Dictionary and Corpus Information into a Vector Measure of Semantic Relatedness*, Master Thesis, University of Minnesota, Duluth, 2003.

[64] S. Patwardhan, S. Banerjee and T. Pedersen, “Using Semantic Relatedness for Word Sense Disambiguation,” *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, 2002.

[65] T. Pedersen, S. Patwardhan, and J. Michelizzi, "WordNet::Similarity - Measuring the Relatedness of Concepts," *Proceedings* of the Nineteenth National Conference on Artificial Intelligence, San Jose, 2004.

[66] M.F. Porter, “An Algorithm for Suffix Stripping,” *Program,* vol. 14, no. 3, 1980, pp. 130-137.

[67] G. Ramakrishnanan and P. Bhattacharyya, “Text Representation with WordNet Synsets using Soft Sense Disambiguation,” *Proceedings of 8th International Conference on Applications of Natural Language to Information Systems (NLDB 2003)* , Burg (Spreewald), Germany, 2003, pp. 214-227.

[68] A. Ratnaparkhi, “A Maximum Entropy Part-Of-Speech Tagger,” *Proceedings of the Empirical Methods in Natural Language Processing Conference,* University of Pennsylvania, May 1996, pp. 17-18.

[69] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, 1995, pp. 448-453.

[70] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Applications to Problems of Ambiguity in Natural Language," *Journal of Artificial Intelligence Research*, vol. 11, 1999, pp. 95-130.

[71] J. C. Reynar and A. Ratnaparkhi, "A Maximum Entropy Approach to Identifying Sentence Boundaries," *Proceedings of the Fifth Conference on Applied Natural Language Processing,* Washington, D.C., March 31-April 3, 1997.

[72] S. Russell and P. Norvig, *Artificial Intelligence: A modern Approach,* Prentice hall, Inc., New Jersey, 1995.

[73] G. Salton, *The SMART Retrieval System – Experiments in Automatic Document Retrieval,* New Jersy, Englewood Cliffs: Prentice Hall Inc., 1971.

[74] G. Salton and C. Buckley, "Term-Weighting Approach in Automatic Text Retrieval," *Information Processing & Management,* vol. 24, no. 5, 1988, pp. 513-523.

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

[76] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-Based Clustering in Spatial Databases: the Algorithm GDBSCAN and its Applications," *Data Mining and Knowledge Discovery,* vol. 2, no. 2, 1998, pp. 169-194.

[77] H. Schmid, "LoPar: Design and Implementation," *Arbeitspapiere des Sonderforschungsbereiches,* vol. 340, no. 149, IMS Stuttgart, July 2000.

[78] C. Shah, B. Chowdhary, and P. Bhattacharyya, "Constructing Better Document Vectors Universal Networking Language (UNL)," *Proceedings of International Conference on Knowledge-Based Computer Systems* (KBCS), 2002.

[79] R. Shier, "Statistics: 2.2 The Wilcoxon signed rank sum test", Mathematics Learning SUpport Centre, Loughborough University, UK, 2004. Available online http://mlsc.lboro.ac.uk/pages/help_with_statistics.html.

[80] D. Sleator and D. Temperley, *Parsing English with a Link Grammar,* Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991.

[81] A. F. Smeaton, R. O'Donnell, and F. Kelledy, “Indexing Structures Derived from Syntax in TREC-3: System Description,” *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, 1994.

[82] M. Steinbach, G. Karypis, and V. Kumar, “A Comparison of Document Clustering Techniques," *KDD Workshop on Text Mining,* 2000.

[83] C.Y. Suen, “N-gram Statistics for Natural Language Understanding and Text Processing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 1, no. 2, 1979, pp. 164-172.

[84] M. Sussna, “Word Sense Disambiguation for Free-text Indexing using a Massive Semantic Network,” *Proceedings of the second international conference on Information and knowledge management*, Washington, D.C., United States, 1993, pp. 67 – 74.

[85] A. Tombros, *The Effectiveness of Query-based Hierarchic Clustering of Documents for Information Retrieval,* Ph.D. dissertation, Dept. Computer Science, Univ. of Glasgow, 2002.

[86] E. Ukkonen, “On-line Construction of Suffix Trees,” *Algorithmica,* vol. 14, no. 3, 1995, pp. 249-260.

[87] W. Wang, J. Yang, and R. Muntz, “STING: a Statistical Information Grid Approach to Spatialdata Mining,” *Proceedings of the 23rd Conference on VLDB,* Athens, Greece, 1997, pp. 186-195.

[88] S. Weiss, H. White, and C. Apt'e, “Lightweight Document Clustering,” *Proceedings of PKDD-2000,* Springer, 2000, pp. 665-672.

[89] W. Wong and A. W. Fu, “Incremental Document Clustering for Web Page Classification,” *IEEE 2000 Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges* (IS2000), Nov 5-8, 2000, Japan.

[90] Z. Wu and M. Palmer, “Verb Semantics and Lexical Selection,” *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, 1994.

[91] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, “A Distribution-Based Clustering Algorithm for Mining Large Spatial Datasets,” *Proceedings of the 14th ICDE,* Orlando, FL, 1998, pp. 324-331.

[92] K. Yang, "Literature review of dissertation," http://www.ils.unc.edu/yangk/dissertation/litrevcontent.htm (Accessed on 12 Feb. 2004).

[93] Y. Wang, *A comparative study of Web document classification methods*, M.S. project report, Mississippi State University, 2002.

[94] Y. Wang and J. Hodges, "A Comparison of Document Clustering Algorithms," *Proceedings of the Fifth International Workshop on Pattern Recognition in Information Systems (PRIS 2005)*, Miami, USA, May 24-25, 2005.

[95] Y. Wang and J. Hodges, "Document Clustering using Compound Words," *Proceedings of the 2005 International Conference on Artificial Intelligence (ICAI2005)*, Las Vegas, Nevada, USA, June 27-30, 2005.

[96] E. Whitley and J. Ball, "Statistics review 6: Nonparametric methods," September 2002. Available online http://ccforum.com/content/6/6/509.

[97] O. Zamir, *Clustering Web Documents: A Phrase-Based Method for Group Search Engine Results,* Ph.D. dissertation, Dept. Computer Science & Engineering, Univ. of Washington, 1999.

[98] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR'98), 1998, pp. 46-54.

[99] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Database," *Proceedings of the ACM SIGMOD Conference on Management of Data,* 1996, pp. 103-114.