A Comparative Study of Text Data Mining Algorithms and its Applications

A Thesis submitted to

The Maharaja Sayajirao University of Baroda

for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Ms. Anjali Ganesh Jivani

Research Guide

Prof. B. S. Parekh

Department of Computer Science and Engineering
Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda
Vadodara 390 001

December 2011

Dedicated to

Kashmira

My friend from childhood – till the cruel hands of nature took her away from me …

Declaration

I, Anjali Ganesh Jivani, hereby declare that the work reported in this thesis entitled 'A Comparative Study of Text Data Mining Algorithms and its Applications' submitted for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

is original and was carried out by me at the Department of Computer Science and Engineering, Faculty of Technology and Engineering, The M. S. University of Baroda, Vadodara. I further declare that this thesis is not substantially the same as one which has already been submitted in part or in full for the award of any degree or academic qualification of this University or any other Institution or examining body in India or abroad. Due acknowledgements have been made in the thesis for all other sources referred.

_____

Ms. Anjali Ganesh Jivani
Department of Computer Science and Engineering
Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda
Vadodara 390001.

Certificate

This is to certify that Ms. Anjali Ganesh Jivani has worked under my guidance to prepare the thesis entitled 'A Comparative Study of Text Data Mining Algorithms and its Applications' which is being submitted herewith towards the requirement for the degree of Doctor of Philosophy in Computer Science and Engineering.

_____

Prof. B. S. Parekh
Department of Computer Science and Engineering
Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda
Vadodara 390 001.

_____

Shri P. R. Bhavsar
Offg. Head
Department of Computer Science and Engineering
Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda
Vadodara 390 001.

_____

Prof. A. N. Mishra
Dean
Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda
Vadodara 390 001.

Approval Sheet

This thesis entitled ‘A Comparative Study of Text Data Mining Algorithms and its Applications’ submitted by Ms. Anjali Ganesh Jivani in Computer Science and Engineering is hereby approved for the degree of Doctor of Philosophy in Computer Science and Engineering.

EXAMINERS:

_____

_____

GUIDE:

_____

(Prof. B. S. Parekh)

Acknowledgements

With a deep sense of gratitude I would like to acknowledge the support of all those people who have directly or indirectly been an integral part of my work.

It is a matter of privilege and pleasure to express my appreciation towards my guide Prof. B. S. Parekh for his motivation, technical guidance and constant support throughout this work. His vision and direction helped me select this subject of research and his critical comments and innovative suggestions encouraged me to complete this work successfully.

I am highly indebted to Shri. P. R. Bhavsar, the Head of my department for his sincere support and co-operation during this period. His silent and strong assistance in providing me all the required facilities has made it possible for me to complete this work in time.

My family has been the backbone for all that I have done till now in my life, especially my two sets of parents – Aai-Baba and Ba-Bapuji. Their unconditional love and moral support are the main reasons for all that has kept me going. Absolutely nothing would have been possible without the unvarying encouragement and consistent inspiration from my husband Ganesh.

I take this opportunity to thank my friends and associates who have supported me during this research – Amisha, Hetal, Neha and Toral.

My work place – Department of Computer Science and Engineering has been a place of worship as well as a place of great joy and encouragement for me because of all my colleagues as well as non-teaching staff members. I would like to thank all of them for their cooperation and assistance during this work.

I would like to express my special gratitude and thanks to Dr. S. K. Vij for his timely suggestions and interest in my work.

Anjali Jivani

Abstract

Text Data Mining, also known as Text Mining or Knowledge Discovery from Textual Data, refers to the process of extracting interesting and non-trivial patterns or knowledge from text documents. Regarded by many as the next wave of knowledge discovery, Text Mining has very high commercial value. Text Mining framework consists of two components: Text refining that transforms unstructured text documents into an intermediate form; and knowledge distillation that deduces patterns or knowledge from the intermediate form.

The enormous amount of information stored in unstructured texts cannot simply be used for further processing by computers, which typically handle text as simple sequences of character strings. Therefore, specific pre-processing methods and algorithms are required in order to extract useful patterns.

There is more to Text Mining than just extracting information from single documents. In fact Text Mining leaps from the old fashioned information retrieval to information and knowledge discovery. The motivation behind this study of Text Data Mining algorithms was to bring out the gold hidden in any organization's data – be it a company or a university. Probably more than 90% of an organization's data are never being looked at: letters from customers, email correspondence, patents, contracts, complaints, etc.

This thesis / research basically involves the study of the existing Text Mining algorithms in different areas like Text Clustering, Text Categorization and Text Summarization and their comparatives in terms of implementation and applications. The initial Text Pre-processing and Text Transformation techniques have also been discussed.

Some innovative algorithms have been developed which are either based on the existing ones with some changes so as to have a better output or represent a novel method of Text Mining. The developed algorithms have been published in Conference Proceedings or International Journals and the details of the publications are mentioned in the respective sections.

Contents

List of Figures

List of Tables

Chapter 1: Text Mining Overview

1.1 Introduction

Text Mining is a flourishing and thriving field that attempts to find meaningful information from textual or rather unstructured data. It may be loosely characterized as the process of analyzing text to extract information that is useful for particular purposes. Compared with the kind of data stored in databases, text is unstructured, amorphous, and difficult to deal with algorithmically. Nevertheless, in modern culture, text is the most common vehicle for the formal exchange of information. The field of Text Mining usually deals with texts whose function is the communication of factual information or opinions, and the motivation for trying to extract information from such text automatically is compelling - even if success is only partial.

In 1999, Hearst wrote that the nascent field of 'Text Data Mining' had a name and a fair amount of hype, but as yet almost no practitioners. Hearst defines Data Mining, information access, and corpus-based computational linguistics and discusses the relationship of these to Text Data Mining. I would be referring to Text Data Mining as Text Mining.

To understand Text Mining it was necessary to understand the concepts, theory and model of Data Mining first. Since the literature on Data Mining is far more extensive, and also more focused: there are numerous textbooks and critical reviews that trace its development from roots in machine learning and statistics. The book 'Data Mining Concepts' by Han and Kamber served as a platform to comprehend the various aspects of Data Mining, its applications and methodologies. This book however contains only a few pages on the concept of Text Mining. There are many other good books which have a very extensive coverage of different Data Mining techniques. They have been mentioned in the bibliography.

There are a number good academic journals on Data Mining – some which are free and some are payable. The 'Data Mining and Knowledge Discovery'

journal of SpringerLink allows abstracts to be accessed by guest. This journal has many latest research papers on Data Mining. Apart from this other journals like ‘Knowledge and Information Systems’, ‘Machine Learning’, ‘IEEE Transactions on Knowledge and Data Engineering’ etc. are other sources of Data Mining related material.

Text Mining emerged at an unfortunate time in history. Data Mining was able to ride the back of the high technology extravaganza throughout the 1990s, and became firmly established as a widely-used practical technology—though the dot com crash may have hit it harder than other areas. Text Mining, in contrast, emerged just before the market crash—the first workshops were held at the International Machine Learning Conference in July 1999 and the International Joint Conference on Artificial Intelligence in August 1999—and missed the opportunity to gain a solid foothold during the boom years.

1.1 Data Mining

Since Data Mining is the superset of Text Mining, it is important to understand Data Mining first. Data is increasing at an unimaginable rate every year. The area of Data Mining has arisen over the last decade to address this problem. Progress in digital data acquisition and storage technology has resulted in the growth of huge databases. This has occurred in all areas starting from simple applications like supermarket transactions, railway reservations to the more complex and complicated ones like space research, molecular databases, images and astronomical bodies etc. Using this data to discover hidden knowledge, unexpected patterns and unknown information is Data Mining.

Data Mining research and practice is in a state similar to that of databases in the 1960s. At that time the concept of databases was new and in the development stage where programmers were still trying to come out of the third generation of languages. Slowly the concept of relational databases was being developed, implemented and improvised upon. Presently we can say that databases are fully implemented and working efficiently all over the world.

The evolution of data warehouses from databases is slowly taking shape. The evolution of Data Mining techniques may take a similar path over the next few decades, making Data Mining techniques easier to use and develop.

Data Mining can be defined as follows:

"Data Mining is the non-trivial extraction of implicit, previously unknown and potentially useful information from data."

Most organizations have large databases that contain a wealth of potentially accessible information. However, it is usually very difficult to access this information. This uncontrolled growth of data will inevitably lead to a situation in which it becomes extremely difficult to access the desired information. In fact it would be like looking for a needle in a haystack.

The sudden rise of interest in Data Mining could be because of the following reasons:

- Most of the organizations have stored gigabytes of data about their products, customers, suppliers, competitors, etc. This database forms a potential gold or rather a diamond mine that can be explored to find hidden and extremely useful information. This information can be traced using simple queries. Data Mining algorithms typically zoom in on interesting sub-parts of the database and dig out the information.
- Since networks have developed extensively, it becomes easy to connect databases situated at remote places. Thus connecting a client's file to a file with demographic data may lead to unexpected views on the spending patterns of certain population groups.
- In the past few years, machine-learning techniques have expanded enormously. Neural networks, genetic algorithms and other techniques often make it easier to find connections in databases.
- The client-server revolution gives the individual access to central information systems. Marketing specialists and policy makers also want to avail themselves of these newly acquired technical possibilities that would help them in making their strategies.

The terms ***Knowledge Discovery in Database (KDD)*** and Data Mining are often used interchangeably. In fact there are other names like knowledge extraction, information discovery, exploratory data analysis, etc. also given to Data Mining. However KDD is the most popular name.

KDD is a process that involves many different steps. The input to this process is the data and the output is the useful information desired by the users. To ensure the usefulness and accuracy of the results of the process, interaction throughout the process by domain experts and technical experts might be needed. Of the many steps in KDD, one of the steps is Data Mining. However, if Data Mining is considered separately, to perform Data Mining all these steps are required. So in a way both mean the same thing.

The different steps of KDD are as follows:

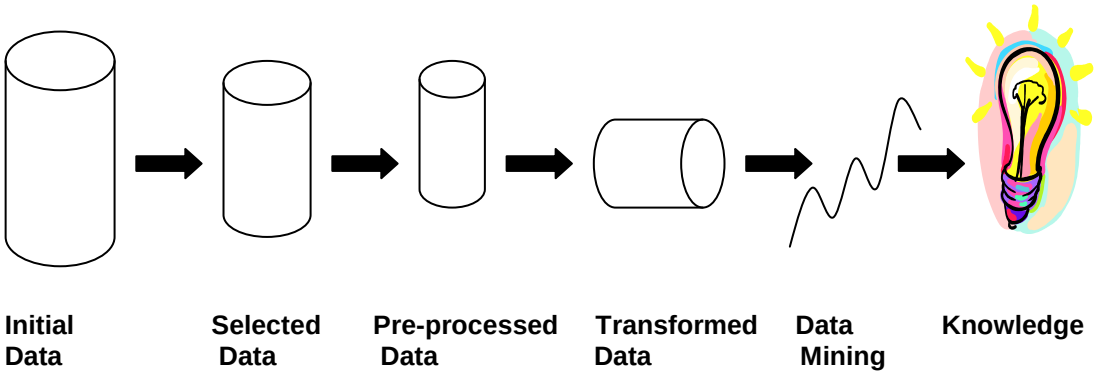

Figure 1.1 Steps of Knowledge Discovery in Databases

Brief description of the steps:

Selection:

- The data obtained from heterogeneous data sources
- The data selected depends on objective of Data Mining
- The data are of different types like active, supplementary, shelf life etc.
- This step is also called the identification and extraction stage

Preprocessing:

- Erroneous data is removed i.e. data that is skewed and invalid
- Missing data is supplied i.e. usually by predicting the values

Transformation:

- The data from different sources is converted to a common format
- If required data is encoded
- Some data conversion is also done i.e. from simple format to more complex one
- If statistics is to be used, several variables may be grouped into one
- If neural networks is used values are changed to 1s and 0s

Data Mining:

- Applying algorithms to the transformed data
- Selection of the correct set of algorithms
- Each set could result in a different type of output

Knowledge:

- This step consists of interpretation and evaluation of results obtained
- This is a heuristic i.e. a self-learning approach
- The result, which is in the form of graphs and charts, is analyzed by experts giving knowledge

The steps shown above are those of Data Mining. When applied to textual data there is a slight change in the steps and the kind of work to be done on the textual data.

1.2 Text Mining

Marti Hearst was one of the first researchers who talked about Text Mining and presented a paper on it in 1999[1]. According to him, Text Mining is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking together of the extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation.

Text Mining is different from what we're familiar with in web search. In search, the user is typically looking for something that is already known and has been written by someone else. The problem is pushing aside all the material that currently isn't relevant to your needs in order to find the relevant information. In Text Mining, the goal is to discover unknown but useful information from documents or rather unstructured data.

To the uninitiated, it may seem that Google and other Web search engines do something similar, since they also pore through reams of documents in split-second intervals. But, as experts note, search engines are merely retrieving information, displaying lists of documents that contain certain keywords.

[1] Hearst, M. *Untangling Text Data Mining* .In the Proceedings of ACL 1999

Text-mining programs go further, categorizing information, making links between otherwise unconnected documents and providing visual maps (some look like tree branches or spokes on a wheel) to lead users down new pathways that they might not have been aware of.

Thus, Text Mining can be defined as:

'The discovery by computer of new, previously unknown information, by automatically extracting information from a usually large amount of unstructured textual resources.'

Text Mining can be compared in a simple form to different concepts like Data Mining, web mining, Natural Language Processing (NLP) etc. as follows:

Data Mining

- In Data Mining the data is structured and generally located in databases and in Text Mining, patterns are extracted from unstructured data in documents and text files
- In Data Mining the information is implicit in the input – data i.e. unknown and not possible to extract without automatic techniques. In Text Mining the information is clearly stated in the input text but it not implied in a manner that is open to automatic processing. Text Mining strives to bring out the text in a form that is suitable for computer processing directly without human intervention

Web Mining

- The source of data is the Web – the largest source of data in the world where in Text Mining, the input is not necessarily the web – it could be textual data from any source (local or otherwise)
- Data on the web is dynamic and rich in features and patterns and the data is text, audio, video, graphics, hyperlinks, tags etc.

Information Retrieval (IR)

- No genuinely new information is found.
- The desired information merely coexists with other valid pieces of information.

Computation Linguistics (CPL) & Natural Language Processing

- An extrapolation from Data Mining on numerical data to Data Mining from textual collections

- CPL computes statistics over large text collections in order to discover useful patterns which are used to inform algorithms for various sub-problems within NLP, e.g. Parts Of Speech tagging, and Word Sense Disambiguation

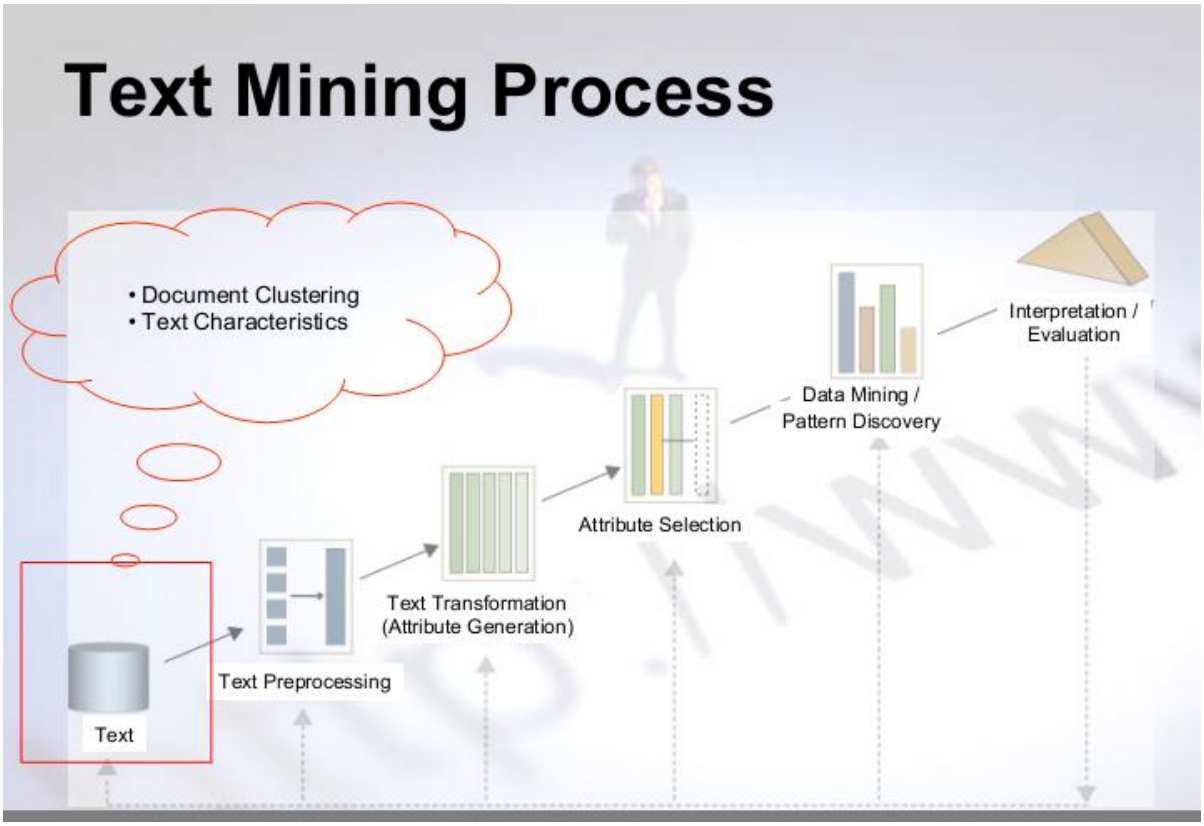

Figure 1.2 The Text Mining Process

1.3 Text Mining – A Research Domain

Although a lot of work has been done on Text Mining, it is still a field of pure delight for researchers like me. As observed from the material – books, papers, online articles, journals, periodicals etc. there is a lot of scope for text miners to compare and contrast the different Text Mining methods and put forth the comparatives in a well organized form. This type of work would be useful to researchers, students and people involved in the Decision Support System of their organizers to mine the large amount of textual information that is available with them.

Researchers and students would find this work very useful for the understanding and detailed study of Text Mining algorithms and methods.

Though a number of books are available, the topics covered are so vast and in some cases too detailed to grasp the real hub and core of Text Mining.

Along with the literature survey I searched and gathered datasets to study and implement the algorithms. There are a number of datasets available on the internet and I have used them for my research after downloading them on my system. Some of the datasets used are as mentioned below:

1. http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html
2. http://www.datawrangling.com/some-datasets-available-on-the-web
3. http://www.infochimps.com/tags/textmining
4. http://archive.ics.uci.edu/ml/
5. http://www.cs.umb.edu/~smimarog/textmining/datasets/index.html
6. http://repository.seasr.org/Datasets/UCI (UCI Machine Learning Repository)
7. TREC
8. REUTERS Collection
9. Times Magazine: http://www.ifs.tuwien.ac.at/~andi/somlib/experiments_time60.html
10. Tehelka Magazine: http://www.tehelka.com/archive_main.asp
11. http://www.cs.toronto.edu/~roweis/data.html
12. Citeseer Abstracts: http: //www.citeseerx.ist.psu.edu
13. The M. S. University of Baroda - Faculty of Tech. & Engg.- results of past years
14. Customer complaints dataset of Matrix Comsec Pvt. Ltd.

The datasets contain structured as well as unstructured data. I initially worked on some of the algorithms with structured data and then implemented them on unstructured data. It was important to understand the Data Mining concepts before actually working on the Text Mining methods. As part of the literature survey and preliminary study I published/presented a number of study and survey papers related to databases, Data Mining and Text Mining. The details are given in the publications chapter at the end. Some brief descriptions of the datasets are as follows:

- **The Reuters collection (Reuters-21578)**

  It is currently the most widely used test collection for text categorization research. The data was originally collected and labeled by Carnegie

Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system. This corpus was used by me in many of my algorithms. It is a standard text dataset used by researchers all over the world.

- **The SOMLib Digital Library**

  This site contains the Time Magazine Article Collection. It is a collection of 420 articles from the 1960's covering news from politics to social gossip.

- **Data for Matlab**

  This site contains text data which can be directly used in a Matlab program. The site address is given in the list above no. 11.

- **UCI Machine Learning Repository**

  This repository contains data in the .csv and .arff formats. Many datasets are available in this repository. The site address is as given in no. 6.

1.4 Layout of the Thesis

Broadly there are five steps involved in Text Data Mining. They are:

1. Text Gathering
2. Text Pre-processing
3. Data Analysis (Attribute generation & selection)
4. Visualization (Applying Text Mining algorithms)
5. Evaluation

The steps are quite similar to those of Data Mining. The most important issue over here was representing the textual data in order to apply the algorithms. The gathered data is pre-processed depending on whether the algorithm was statistical or some other. The text gathering was not a very difficult task as textual corpus for research is available on different sites with all the necessary metadata information regarding the layout of the data sets. The corpus details are mentioned above as well as along with the algorithms where they have been used.

Chapter 2 and Chapter 3 are related to the process of text pre-processing and text transformation. The steps related to pre-processing are discussed along

with the details of the models, measures and concepts used in pre-processing. The text transformation is related to transforming the text into a format which can be used for implementing the algorithms and creating the similarity matrices. The Vector Space Model, the Latent Semantic Analysis etc. is discussed in this chapter.

The different Text Mining methods and the study related to the existing work, their drawbacks or limitations and the proposed new algorithms with the implementation details and results has been covered in subsequent chapters.

Chapters 4, 5 and 6 are related to Text Clustering, Text Categorization and Text Summarization. In each chapter the perceptions and notions of the related area, different models as well as a comparative between the different methods available are given. The proposed algorithms and their implementation details are also shown and the related publication if any is mentioned at the appropriate places. The conclusions, results and future enhancements are mentioned towards the end of each chapter.

Chapter 7 briefly describes the future enhancements possible in this field of Text Mining.

The chapters are followed by the Summary of the work, the Appendices, Publication Details and Bibliography.

The Appendix – A contains the list of stop words and Appendix – B contains a list of Text Mining tools available.

Chapter 2: Text Pre-processing

2.1 Introduction

Though this is considered to be the preliminary step to be conducted before actually applying Text Mining algorithms/methods, it is a very important process and this routine itself is divided into a number of sub-methods which again have optional algorithms with their own set of advantages and disadvantages. The text data on which I have executed the algorithm have been first converted to text format if it was not so. In fact the majority of the datasets were already in text format.

Most of the Text Mining approaches are based on the idea that a text document can be described on the set of words contained in it i.e. bag-of-words representation. The pre-processing itself is made up of a sequence of steps. The steps are explained in detail.

2.2 Morphological Analysis

The first step in text-preprocessing is the morphological analyses. It is divided into three subcategories: tokenization, filtering and stemming. Morphology is a part of linguistics which is dealing with words. Therefore, it deals with the smallest, useful unit of a document. One could say that characters are the smallest unit. Nonetheless, characters do not carry any valuable information for information retrieval. Firstly, Text Mining requires the words and the endings of a document. Finding words and separating them is known as tokenization.

The next step is filtering of important and relevant words from our list of words which were the output of tokenization. This is also called stop words removal.

The third step is stemming. Stemming is very important and a lot of research work has already been done on it. Stemming reduces words variants to its root form. Stemming of words increases the recall and precision of the information retrieval in Text Mining. The term recall describes the proportion of all relevant documents in a data set that are retrieved by the information

retrieval system. The term precision describes the proportion of relevant documents in the data set returned to the user. Precision and recall are two very important measures for text categorization, clustering as well as summarization. The details are discussed further as and when they are applied.

2.3 Tokenization

Over here the input document is split into a set of words by removing all punctuation marks, tabs and other non-text characters and replacing them with white spaces. The part-of-speech (POS) tagging is also applied in some cases where words are tagged according to the grammatical context of the word in the sentence, hence dividing up the words into nouns, verbs, etc. This is important for the exact analysis of relations between words.

Another approach was to ignore the order in which the words occurred and instead focus on their statistical distributions (the bag-of-words approach). In this case it is necessary to index the text into data vectors. I have used the bag-of-words approach in implementing the algorithms. The POS becomes important if the research is related to NLP. In one algorithm as part of extension work POS has been implemented.

Tokenization has been done using Visual Basic (using strip () function) as well as Matlab (using strtok () function). The Matlab function was found to be much more efficient and fast.

2.4 Filtering

This step is related to removing words which are of no importance for our Text Mining process like articles, prepositions, conjunctions, etc. This is also known as ‘Stop Words Filtering’. It is controlled by human input and not automated. There is not one definite list of stop words which all tools use, if even used. The stop words list is available on the site of the Onix Text Retrievel Tookit and the site is:

http://www.lextek.com/manuals/onix/stopwords1.html.

This is a very popular list and as per the requirement the list can be modified. I have used this list to remove the stop words. Another popular list is available on the MIT site and can be downloaded from: http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop.

2.5 Stemming

2.5.1 Introduction

Word stemming is an important feature supported by present day indexing and search systems. Indexing and searching are in turn part of Text Mining applications, Natural Language Processing (NLP) systems and Information Retrieval (IR) systems. The main idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Recall in increased without compromising on the precision of the documents fetched. Stemming is usually done by removing any attached suffixes and prefixes (affixes) from index terms before the actual assignment of the term to the index. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents in an IR system. Text clustering, categorization and summarization also require this conversion as part of the pre-processing before actually applying any related algorithm.

A lot of research work has already been done on stemming[1] and a number of different algorithms have already been developed and implemented. In this section a brief description of the available stemmers and their comparatives is presented. A paper titled 'A Comparative Study of Stemming Algorithms' has been published by me as part of my research in the journal - '**International Journal of Computer Technology and Applications' (**IJCTA) - Volume 2 Issue 6/ November - December 2011/ pg. 1930-1938, ISSN:2229-6093. This journal has been indexed by Scirus, .docstoc, Scribd, Google Scholar, DOAJ, etc. Site: http://ijcta.com/vol2issue6.php

[1] The papers that I have referred to understand stemming methods and prepare a comparative based on them have been mentioned in the Bibliography – from [4] to [27].

Errors in Stemming

There are mainly two errors in stemming – over stemming and under stemming. Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive. Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Paice has proved that light-stemming reduces the over-stemming errors but increases the under-stemming errors. On the other hand, heavy stemmers reduce the under-stemming errors while increasing the over-stemming errors.

Classification of Stemming Algorithms

Broadly, stemming algorithms can be classified in three groups: truncating methods, statistical methods, and mixed methods. Each of these groups has a typical way of finding the stems of the word variants. These methods are shown in the Figure 2.1.

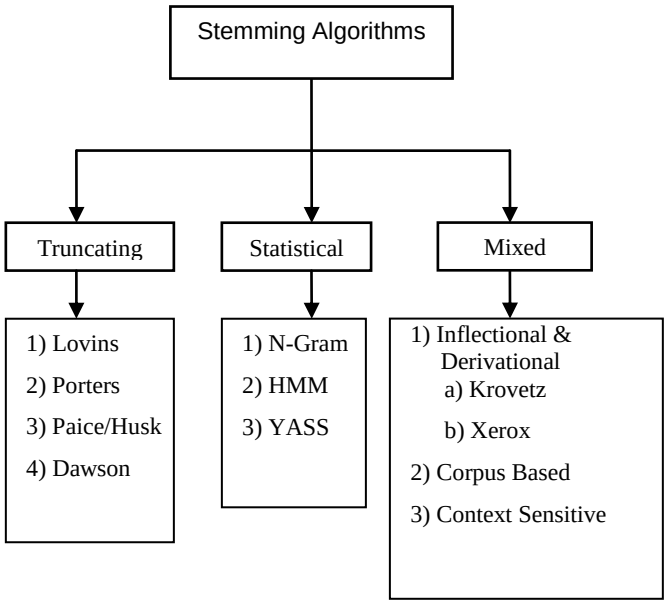

Figure 2.1 Types of Stemming Algorithms

2.5.2 Truncating Methods

As the name clearly suggests these methods are related to removing the suffixes or prefixes (commonly known as affixes) of a word. The most basic

stemmer was the Truncate (n) stemmer which truncated a word at the nth symbol i.e. keep n letters and remove the rest. In this method words shorter than n are kept as it is. The chances of over stemming increases when the word length is small.

Another simple approach was the S-stemmer – an algorithm conflating singular and plural forms of English nouns. This algorithm was proposed by Donna Harman. The algorithm has rules to remove suffixes in plurals so as to convert them to the singular forms.

Lovins Stemmer

This was the first popular and effective stemmer proposed by Lovins in 1968. It performs a lookup on a table of 294 endings, 29 conditions and 35 transformation rules, which have been arranged on a longest match principle. The Lovins stemmer removes the longest suffix from a word. Once the ending is removed, the word is recoded using a different table that makes various adjustments to convert these stems into valid words. It always removes a maximum of one suffix from a word, due to its nature as single pass algorithm. The advantages of this algorithm is it is very fast and can handle removal of double letters in words like ‘getting’ being transformed to ‘get’ and also handles many irregular plurals like – mouse and mice, index and indices etc.

Drawbacks of the Lovins approach are that it is time and data consuming. Furthermore, many suffixes are not available in the table of endings. It is sometimes highly unreliable and frequently fails to form words from the stems or to match the stems of like-meaning words. This is because of the technical vocabulary being used by the author.

Porters Stemmer

Porters stemming algorithm is as of now one of the most popular stemming methods proposed in 1980. Many modifications and enhancements have been done and suggested on the basic algorithm. It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. It has five steps, and within each step, rules are applied until one of them passes the conditions. If a rule

is accepted, the suffix is removed accordingly, and the next step is performed. The resultant stem at the end of the fifth step is returned.

The rule looks like the following:

<condition> <suffix> → <new suffix>

For example, a rule (m>0) EED → EE means “if the word has at least one vowel and consonant plus EED ending, change the ending to EE”. So “agreed” becomes “agree” while “feed” remains unchanged. This algorithm has about 60 rules and is very easy to comprehend.

Porter designed a detailed framework of stemming which is known as ‘Snowball’. The main purpose of the framework is to allow programmers to develop their own stemmers for other character sets or languages. Currently there are implementations for many Romance, Germanic, Uralic and Scandinavian languages as well as English, Russian and Turkish languages.

Based on the stemming errors, Paice reached to a conclusion that the Porter stemmer produces less error rate than the Lovins stemmer. However it was noted that Lovins stemmer is a heavier stemmer that produces a better data reduction. The Lovins algorithm is noticeably bigger than the Porter algorithm, because of its very extensive endings list. But in one way that is used to advantage: it is faster. It has effectively traded space for time, and with its large suffix set it needs just two major steps to remove a suffix, compared with the five of the Porter algorithm.

Paice/Husk Stemmer

The Paice/Husk stemmer is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix. On each iteration, it tries to find an applicable rule by the last character of the word. Each rule specifies either a deletion or replacement of an ending. If there is no such rule, it terminates. It also terminates if a word starts with a vowel and there are only two letters left or if a word starts with a consonant and there are only three characters left. Otherwise, the rule is applied and the process repeats.

The advantage is its simple form and each iteration taking care of both deletion and replacement as per the rule applied.

The disadvantage is it is a very heavy algorithm and over stemming may occur.

Dawson Stemmer

This stemmer is an extension of the Lovins approach except that it covers a much more comprehensive list of about 1200 suffixes. Like Lovins it too is a single pass stemmer and hence is pretty fast. The suffixes are stored in the reversed order indexed by their length and last letter. In fact they are organized as a set of branched character trees for rapid access.

The advantage is that it covers more suffixes than Lovins and is fast in execution.

The disadvantage is it is very complex and lacks a standard reusable implementation.

2.5.3 Statistical Methods

These are the stemmers who are based on statistical analysis and techniques. Most of the methods remove the affixes but after implementing some statistical procedure.

N-Gram Stemmer

This is a very interesting method and it is language independent. Over here string-similarity approach is used to convert word inflation to its stem. An n-gram is a string of n, usually adjacent, characters extracted from a section of continuous text. To be precise an n-gram is a set of n consecutive characters extracted from a word. The main idea behind this approach is that, similar words will have a high proportion of n-grams in common. For n equals to 2 or 3, the words extracted are called digrams or trigrams, respectively. For example, the word ‘INTRODUCTIONS’ results in the generation of the digrams:

I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S

and the trigrams:

I, *IN, INT, NTR, TRO, ROD, ODU, DUC, UCT, CTI, TIO, ION, ONS, NS*, S

Where '*' denotes a padding space. There are n+1 such digrams and n+2 such trigrams in a word containing n characters.

Most stemmers are language-specific. Generally a value of 4 or 5 is selected for n. After that a textual data or document is analyzed for all the n-grams. It is obvious that a word root generally occurs less frequently than its morphological form. This means a word generally has an affix associated with it. A typical statistical analysis based on the inverse document frequency (IDF) can be used to identify them.

This stemmer has an advantage that it is language independent and hence very useful in many applications.

The disadvantage is it requires a significant amount of memory and storage for creating and storing the n-grams and indexes and hence is not a very practical approach.

HMM Stemmer

This stemmer is based on the concept of the Hidden Markov Model (HMMs) which are finite-state automata where transitions between states are ruled by probability functions. At each transition, the new state emits a symbol with a given probability. This model was proposed by Melucci and Orio.

This method is based on unsupervised learning and does not need a prior linguistic knowledge of the dataset. In this method the probability of each path can be computed and the most probable path is found using the Viterbi coding in the automata graph.

In order to apply HMMs to stemming, a sequence of letters that forms a word can be considered the result of a concatenation of two subsequences: a prefix and a suffix. A way to model this process is through an HMM where the states are divided in two disjoint sets: initial can be the stems only and the later can be the stems or suffixes. Transitions between states define word building process. There are some assumptions that can be made in this method:

1. Initial states belong only to the stem-set - a word always starts with a stem

2. Transitions from states of the suffix-set to states of the stem-set always have a null probabiliy - a word can be only a concatenation of a stem and a suffix.
3. Final states belong to both sets - a stem can have a number of different derivations, but it may also have no suffix.

For any given word, the most probable path from initial to final states will produce the split point (a transition from roots to suffixes). Then the sequence of characters before this point can be considered as a stem.

The advantage of this method is it is unsupervised and hence knowledge of the language is not required.

The disadvantage is it is a little complex and may over stem the words sometimes.

YASS Stemmer

The name is an acronym for Yet Another Suffix Striper. This stemmer was proposed by Prasenjit Majumder, et al. According to the authors the performance of a stemmer generated by clustering a lexicon without any linguistic input is comparable to that obtained using standard, rule-based stemmers such as Porter's. This stemmer comes under the category of statistical as well as corpus based. It does not rely on linguistic expertise. Retrieval experiments by the authors on English, French, and Bengali datasets show that the proposed approach is effective for languages that are primarily suffixing in nature.

The clusters are created using hierarchical approach and distance measures. Then the resulting clusters are considered as equivalence classes and their centroids as the stems. As per the details given by Prasenjit, the edit distance and YASS distance calculations for two string comparisons is shown in Figure 2.2 and Figure 2.3. The YASS distance measures D_1, D_2, D_3 and D_4 are based on a Boolean function pi for penalty. It is defined as:

$$p_i = 0 \quad if\ x_i = y_i\ 0 \leq i \leq \min(n, n')$$
$$p_i = 1 \quad otherwise$$

Where X and Y are two strings, $X = x_0x_1x_2 \ldots x_n$ and $Y = y_0y_1y_2 \ldots y_n$. If the strings are of unequal lengths we pad the shorter string with null characters to make the strings lengths equal. Smaller the distance measure indicates greater similarity between the strings. The edit distance between two strings of characters is the number of operations required to transform one of them into the other.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
a	s	t	r	o	n	o	m	e	r	-	-	-	-
a	s	t	r	o	n	o	m	i	c	a	l	l	Y

$D_1 = 1/2^8 + 1/2^9 + \ldots + 1/2^{13} = 0.0077$

$D_2 = 1/8 \times (1/2^0 + \ldots + 1/2^{13-8}) = 0.2461$

$D_3 = 6/8 \times (1/2^0 + \ldots + 1/2^{13-8}) = 1.4766$

$D_4 = 6/14 \times (1/2^0 + \ldots + 1/2^{13-8}) = 0.8438$

Edit distance = 6

Figure 2.2 Calculation of distance measures – 1

0	1	2	3	4	5	6	7	8	9
a	s	t	r	o	n	o	m	e	r
a	s	t	o	n	i	s	h	-	-

$D_1 = 1/2^3 + 1/2^4 + \ldots + 1/2^9 = 0.2480$

$D_2 = 1/3 \times (1/2^0 + \ldots + 1/2^{9-3}) = 0.6615$

$D_3 = 7/3 \times (1/2^0 + \ldots + 1/2^{9-3}) = 4.6302$

$D_4 = 7/10 \times (1/2^0 + \ldots + 1/2^{9-3}) = 1.3891$

Edit distance = 5

Figure 2.3. Calculation of distance measures - 2

As per the distances D_1, D_2, D_3 and D_4 it can be seen that astronomer and astronomically are more similar than astronomer and astonish. The edit distance shows exactly opposite which means the new distance measures are more accurate.

2.5.4 Inflectional and Derivational Methods

This is another approach to stemming and it involves both the inflectional as well as the derivational morphology analysis. The corpus should be very large to develop these types of stemmers and hence they are part of corpus base stemmers too. In case of inflectional the word variants are related to the language specific syntactic variations like plural, gender, case, etc whereas in derivational the word variants are related to the part-of-speech (POS) of a sentence where the word occurs.

Krovetz Stemmer (KSTEM)

The Krovetz stemmer was presented in 1993 by Robert Krovetz and is a linguistic lexical validation stemmer. Since it is based on the inflectional property of words and the language syntax, it is very complicated in nature. It effectively and accurately removes inflectional suffixes in three steps:

1. Transforming the plurals of a word to its singular form
2. Converting the past tense of a word to its present tense
3. Removing the suffix ‘ing’

The conversion process first removes the suffix and then through the process of checking in a dictionary for any recoding, returns the stem to a word. The dictionary lookup also performs any transformations that are required due to spelling exception and also converts any stem produced into a real word, whose meaning can be understood.

The strength of derivational and inflectional analysis is in their ability to produce morphologically correct stems, cope with exceptions, processing prefixes as well as suffixes. Since this stemmer does not find the stems for all word variants, it can be used as a pre-stemmer before actually applying a stemming algorithm. This would increase the speed and effectiveness of the main stemmer. Compared to Porter and Paice / Husk, this is a very light stemmer. The Krovetz stemmer attempts to increase accuracy and robustness by treating spelling errors and meaningless stems.

If the input document size is large this stemmer becomes weak and does not perform very effectively. The major and obvious flaw in dictionary-based algorithms is their inability to cope with words, which are not in the lexicon.

Also, a lexicon must be manually created in advance, which requires significant efforts. This stemmer does not consistently produce a good recall and precision performance.

Xerox Inflectional and Derivational Analyzer

The linguistics groups at Xerox have developed a number of linguistic tools for English which can be used in information retrieval. In particular, they have produced English lexical database which provides a morphological analysis of any word in the lexicon and identifies the base form. Xerox linguists have developed a lexical database for English and some other languages also which can analyze and generate inflectional and derivational morphology. The inflectional database reduces each surface word to the form which can be found in the dictionary, as follows:

- nouns singular (e.g. children child)
- verbs infinitive (e.g. understood understand)
- adjectives positive form (e.g. best good)
- pronoun nominative (e.g. whom who)

The derivational database reduces surface forms to stems which are related to the original in both form and semantics. For example, ‘government’ stems to ‘govern’ while ‘department’ is not reduced to ‘depart’ since the two forms have different meanings. All stems are valid English terms, and irregular forms are handled correctly. The derivational process uses both suffix and prefix removal, unlike most conventional stemming algorithms which rely solely on suffix removal. A sample of the suffixes and prefixes that are removed is given below:

- Suffixes: ly, ness, ion, ize, ant, ent, ic, al, Ic, ical, able, ance, ary, ate, ce, y, dom, ee, eer, ence, ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ous, ty, ship, some, ure
- Prefixes: anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tn, ultra, un

The databases are constructed using finite state transducers, which promotes very efficient storage and access. This technology also allows the conflation

process to act in reverse, generating all conceivable surface forms from a single base form. The database starts with a lexicon of about 77 thousand base forms from which it can generate roughly half a million surface forms.

The advantages of this stemmer are that it works well with a large document also and removes the prefixes also where ever applicable. All stems are valid words since a lexical database which provides a morphological analysis of any word in the lexicon is available for stemming. It has proved to work better than the Krovetz stemmer for a large corpus.

The disadvantage is that the output depends on the lexical database which may not be exhaustive. Since this method is based on a lexicon, it cannot correctly stem words which are not part of the lexicon. This stemmer has not been implemented successfully on many other languages. Dependence on the lexicon makes it a language dependent stemmer.

2.5.5 Corpus Based Method

This method of stemming was proposed by Xu and Croft. They have suggested an approach which tries to overcome some of the drawbacks of Porter stemmer. For example, the words ‘policy’ and ‘police’ are conflated though they have a different meaning but the words ‘index’ and ‘indices’ are not conflated though they have the same root. Porter stemmer also generates stems which are not real words like ‘iteration’ becomes ‘iter’ and ‘general’ becomes ‘gener’. Another problem is that while some stemming algorithms may be suitable for one corpus, they will produce too many errors on another.

Corpus based stemming refers to automatic modification of conflation classes – words that have resulted in a common stem, to suit the characteristics of a given text corpus using statistical methods. The basic hypothesis is that word forms that should be conflated for a given corpus will co-occur in documents from that corpus. Using this concept some of the over stemming or under stemming drawbacks are resolved e.g. ‘policy’ and ‘police’ will no longer be conflated.

To determine the significance of word form co-occurrence, the statistical measure used is,

$$Em(a, b) = nab / (na + nb)$$

Where, a and b are a pair of words, na and nb are the number of occurrences of a and b in the corpus, nab is the number of times both a and b fall in a text window of size win in the corpus (they co-occur).

The way this stemmer works is to first use the Porter stemmer to identify the stems of conflated words and then the next step is to use the corpus statistics to redefine the conflation. Sometimes the Krovetz stemmer (KSTEM) along with Porter stemmer is used in the initial stem to make sure that word conflations are not missed out.

The advantage of this method is it can potentially avoid making conflations that are not appropriate for a given corpus and the result is an actual word and not an incomplete stem.

The disadvantage is that you need to develop the statistical measure for every corpus separately and the processing time increases as in the first step two stemming algorithms are first used before using this method.

2.5.6 Context Sensitive Method

This is a very interesting method of stemming unlike the usual method where stemming is done before indexing a document, over here for a Web Search, context sensitive analysis is done using statistical modeling on the query side. This method was proposed by Funchun Peng et al.

Basically for the words of the input query, the morphological variants which would be useful for the search are predicted before the query is submitted to the search engine. This dramatically reduces the number of bad expansions, which in turn reduces the cost of additional computation and improves the precision at the same time.

After the predicted word variants from the query have been derived, a context sensitive document matching is done for these variants. This conservative strategy serves as a safeguard against spurious stemming, and it turns out to be very important for improving precision.

This stemming process is divided into four steps after the query is fired:

1. Candidate generation:

Over here the Porter stemmer is used generate the stems from the query words. This has absolutely no relation to the semantics of the words. For a

better output the corpus-based analysis based on distributional similarity is used. The rationale of using distributional word similarity is that true variants tend to be used in similar contexts. In the distributional word similarity calculation, each word is represented with a vector of features derived from the context of the word. We use the bigrams to the left and right of the word as its context features, by mining a huge Web corpus. The similarity between two words is the cosine similarity between the two corresponding feature vectors.

2. Query Segmentation and head word detection:

When the queries are long, it is important to detect the main concept of the query. The query is broken into segments which are generally the noun phrases. For each noun phrase the most important word is detected which is the head word. Sometimes a word is split to know the context. The mutual information of two adjacent words is found and if it passes a threshold value, they are kept in the same segment. Finding the headword is by using a syntactical parser.

3. Context sensitive word expansion:

Now that the head words are obtained, using probability measures it is decided which word variants would be most useful – generally they are the plural forms of the words. This is done using the simplest and most successful approach to language modeling which is the one based on the *n*-gram model which uses the chain rule of probability. In this step all the important head word variants are obtained. The traditional way of using stemming for Web search, is referred as the naive model. This is to treat every word variant equivalent for all possible words in the query. The query “book store” will be transformed into “*(book OR books)(store OR stores)*” when limiting stemming to pluralization handling only, where *OR* is an operator that denotes the equivalence of the left and right arguments.

4. Context sensitive document matching:

Now that we have the word variants, in this step a variant match is considered valid only if the variant occurs in the same context in the document as the original word does in the query. The context is the left or the right non-stop segments of the original word. Considering the fact that queries and documents may not represent the intent in exactly the same way, this

proximity constraint is to allow variant occurrences within a window of some fixed size. The smaller the window size is, the more restrictive the matching.
The advantage of this stemmer is it improves selective word expansion on the query side and conservative word occurrence matching on the document side. The disadvantage is the processing time and the complex nature of the stemmer. There can be errors in finding the noun phrases in the query and the proximity words.

2.6 Stemming and Lemmatizing

The basic function of both the methods – stemming and lemmatizing is similar. Both of them reduce a word variant to its 'stem' in stemming and 'lemma' in lemmatizing. There is a very subtle difference between both the concepts. In stemming the 'stem' is obtaining after applying a set of rules but without bothering about the part of speech (POS) or the context of the word occurrence. In contrast, lemmatizing deals with obtaining the 'lemma' of a word which involves reducing the word forms to its root form after understanding the POS and the context of the word in the given sentence.
In stemming, conversion of morphological forms of a word to its stem is done assuming each one is semantically related. The stem need not be an existing word in the dictionary but all its variants should map to this form after the stemming has been completed. There are two points to be considered while using a stemmer:

- Morphological forms of a word are assumed to have the same base meaning and hence should be mapped to the same stem
- Words that do not have the same meaning should be kept separate

These two rules are good enough as long as the resultant stems are useful for our Text Mining or language processing applications. Stemming is generally considered as a recall-enhancing device. For languages with relatively simple morphology, the influence of stemming is less than for those with a more complex morphology. Most of the stemming experiments done so far are for English and other west European languages.
Lemmatizing deals with the complex process of first understanding the context, then determining the POS of a word in a sentence and then finally

finding the ‘lemma’. In fact an algorithm that converts a word to its linguistically correct root is called a lemmatizer. A lemma in morphology is the canonical form of a lexeme. Lexeme, in this context, refers to the set of all the forms that have the same meaning, and lemma refers to the particular form that is chosen by convention to represent the lexeme.

In computational linguistics, a stem is the part of the word that never changes even when morphologically inflected, whilst a lemma is the base form of the verb. Stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications. Lemmatizers are difficult to implement because they are related to the semantics and the POS of a sentence. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. The results are not always morphologically right forms of words. Nevertheless, since document index and queries are stemmed "invisibly" for a user, this peculiarity should not be considered as a flaw, but rather as a feature distinguishing stemming from lemmatization. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the lemma.

For example, the word inflations like gone, goes, going will map to the stem ‘go’. The word ‘went’ will not map to the same stem. However a lemmatizer will map even the word ‘went’ to the lemma ‘go’.

Stemming:

introduction, introducing, introduces – introduc

gone, going, goes – go

Lemmatizing:

introduction, introducing, introduces – introduce

gone, going, goes, went – go

2.7 Comparison between stemming methods

As per all the methods and the related stemming algorithms discussed so far, a comparative of them related to their advantages and limitations is shown in Table 4, Table 5 and Table 6. It is clearly deduced that none of the stemmers

are totally exhaustive but more or less the purpose of stemming is resolved. As of now the Porter's Stemmer is the most popular and researchers make their own changes in the basic algorithm to cater to their requirements.

Table 2.1 Truncating (Affix Removal) Methods

Stemmer	Advantages	Limitations
Lovins	1) Fast – single pass algorithm. 2) Handles removal of double letters in words like 'getting' being transformed to 'get'. 3) Handles many irregular plurals like – mouse and mice etc.	1) Time consuming. 2) Not all suffixes available. 3) Not very reliable and frequently fails to form words from the stems . 4) Dependent on the technical vocabulary being used by the author.
Porters	1) Produces the best output as compared to other stemmers. 2) Less error rate. 3) Compared to Lovins it's a light stemmer. 4) The Snowball stemmer framework designed by Porter is language independent approach to stemming.	1) The stems produced are not always real words. 2) It has at least five steps and sixty rules and hence is time consuming.
Paice / Husk	1) Simple form. 2) Each iteration takes care of deletion and replacement.	1) Heavy algorithm. 2) Over stemming may occur.
Dawson	1) Covers more suffixes than Lovins. 2) Fast in execution.	1) Very complex. 2) Lacks a standard implementation.

Table 2.2 Statistical Methods

Stemmer	Advantages	Limitations
N-Gram	1) Based on the concept of n-grams and string comparisons. 2) Language independent.	1) Not time efficient. 2) Requires significant amount of space for creating and indexing the n-grams. 3) Not a very practical method.
HMM	1) Based on the concept of Hidden Markov Model. 2) Unsupervised method and so is language independent.	1) A complex method for implementation. 2) Over stemming may occur in this method.
YASS	1) Based on hierarchical clustering approach and distance measures. 2) It is also a corpus based method. 3) Can be used for any language without knowing its morphology.	1) Difficult to decide a threshold for creating clusters. 2) Requires significant computing power.

Table 2.3 Inflectional & Derivational Methods

Stemmer	Advantages	Limitations
Krovetz	1) It is a light stemmer. 2) Can be used as a pre-stemmer for other stemmers.	1) For large documents, this stemmer is not efficient. 2) Inability to cope with words outside the lexicon. 3) Does not consistently produce a good recall and precision. 4) Lexicon to be created in advance.
Xerox	1) Works well for a large document also. 2) Removes the prefixes where ever applicable. 3) All stems are valid words.	1) Inability to cope with words outside the lexicon. 2) Not implemented successfully on language other than English. Over stemming may occur in this method. 3) Dependence on the lexicon makes it language dependent.

2.8 Syntactical and Semantical Analysis

2.8.1 Syntactical Analysis

This analysis deals with the syntax of a sentence in natural language and is useful in Information Retrieval systems. It can be divided in three parts: part-of-speech tagging, phrase recognition and parsing.

1. Part-of-speech tagging: The recognition of the elements of a sentence like nouns, verbs, adjectives, prepositions, etc. is realized through part of speech tagging (POS tagging).

   The part-of-speech (POS) tagging is also applied in some cases where words are tagged according to the grammatical context of the word in the sentence, hence dividing up the words into nouns, verbs, etc. This is important for the exact analysis of relations between words.
2. Phrase Recognition (PR): This is also very similar to POS. It is required to locate group of words or phrases. PR finds phrases like those given below:
   - Preposition phrase (e.g. in love)
   - Noun Phrase(e.g. the magician of Mecca)
   - Verb Phrase (e.g. do business)
   - Adjectival Phrase (e.g. small house)
   - Adverbial Phrase (e.g. very quickly)
3. Parsing: This process is also part of POS as well as phrase recognition. The sentences are fractionalized into grammatical units. The Stanford parser is very popular for parsing. It generates a tree which is useful for information extraction.

2.8.2 Semantical Analysis

This part of pre-processing deals with the meaning of the textual data i.e. the semantics. It is more or less related to Natural Language Processing.

2.9 Conclusion

As can be seen from all the algorithms that have been discussed so far, there is a lot of similarity between the stemming algorithms and if one algorithm scores better in one area, the other does better in some other area. In fact,

none of them give 100% output but are good enough to be applied to the Text Mining, NLP or IR applications.

The main difference lies in using either a rule-based approach or a linguistic one. A rule based approach may not always give correct output and the stems generated may not always be correct words. As far as the linguistic approach is concerned, since these methods are based on a lexicon, words outside the lexicon are not stemmed properly. It is of utmost importance that the lexicon being used is totally exhaustive which is a matter of language study. A statistical stemmer may be language independent but does not always give a reliable and correct stem.

The problem of over stemming and under stemming can be reduced only if the syntax as well as the semantics of the words and their POS is taken into consideration. This in conjunction with a dictionary look-up can help in reducing the errors and converting stems to words. However no perfect stemmer has been designed so far to match all the requirements.

For the purpose of implementation tokenizing has been implemented using both Visual Basic and Matlab, filtering by Matlab and stemming by Visual Basic. The stemming method implemented is the Porters Stemming as it has been found to be the most appropriate by most of the researchers.

2.10 Future Enhancements

Although a lot of research work has already been done in developing stemmers there still remains a lot to be done to improve recall as well as precision.

There is a need for a method and a system for efficient stemming that reduces the heavy tradeoff between false positives and false negatives. A stemmer that uses the syntactical as well as the semantical knowledge to reduce stemming errors should be developed. Perhaps developing a good lemmatizer could help in achieving the goal.

Chapter 3: Text Transformation

3.1 Introduction

This is one of the most important stages as this is the process where the data is modeled as per the Text Mining technique which is going to be selected and used. So depending on whether we are planning for clustering, summarization or categorization, the data transformation technique/model is selected. Data Transformation is also called the Dimension Reduction Technique. Since I have used the bad-of-words approach, the transformation techniques would be related to it.

Broadly I am covering the most common and most efficient techniques which I have focused on. Other related techniques would be explained as and when required along with the algorithm whenever it is discussed.

3.2 The Vector Space Model (VSM)

3.2.1 Introduction to VSM

This model was proposed by Salton and it incorporates the local as well as global information about terms in a document and corpus.

It is an algebraic model for representing text documents as vectors of identifiers. The vector space model procedure can be divided in to three stages. The first stage is the document indexing where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance retrieval of document relevant to the user. The last stage ranks the document with respect to the query according to a similarity measure. The term ‘query’ is used because this model is used in Information Retrieval also.

The similarity between documents or a query and a document is determined through calculations of the cosine similarity, Dice’s coefficient, the Jaccard’s coefficient and in some cases the Eucliean distance. The vector space model has been shown diagrammatically as in Figure 3.1. In the figure, d1 and d2

are document vectors and q1 is the query vector. We call them vectors because they are made up of different terms.

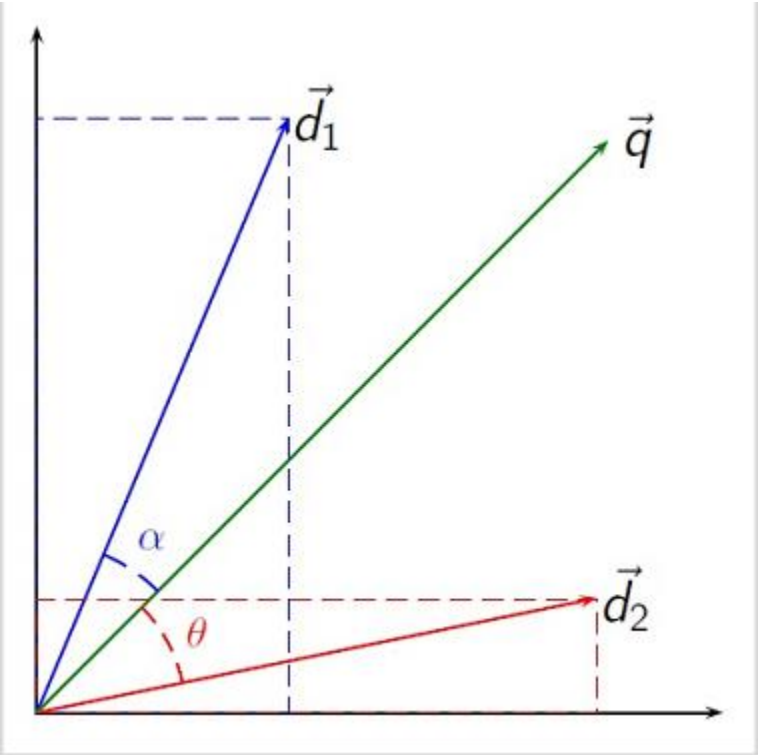

Figure 3.1 The Vector Space Model

The angle between the documents or the query and documents determines the similarity between them. The similarity measures are discussed in section 3.2.3.

3.2.2 The tf-idf score

The document indexing is done using the tf-idf method. It stands for term-frequency (tf) and inverse document frequency (idf). It is weight based on statistics which is assigned to a word to evaluate its importance in a single document or a collection of documents. This weight is also used to generate ranking in documents. It is used in almost all Text Mining algorithms. Over here the assumption is that the first three steps of data pre-processing – tokenization, removing stop words and stemming is already complete.

In the VSM, each document d is considered to be a vector in the term-space i.e. terms that make the document. A document d can be represented as,

dtf = (tf1, tf2, …, tfn),

where tfi is the frequency of the ith term in document d. In this way each term in a document can be represented by the tf vector. Since all documents are not of the same size, we normalize the term frequency by dividing it by the total number of unique terms in the document.

The inverse document frequency (idf) is a measure of the general importance of the term in the corpus. It is obtained by dividing the total number of documents by the number of documents containing the term and taking the logarithm of that quotient.

$$idf(t) = log\ \frac{|D|}{|d{:}t \epsilon d|} \qquad (3.1)$$

Where,

|D| - total number of documents in the corpus

$|d{:}t\epsilon d|$ - number of documents where term t appears

If a term is not in the corpus this will lead to division by zero and so we adjust (1) by adding 1 to the denominator. i.e. 1 + $|d{:}t\epsilon d|$.

So now the tf-idf score for a term in a document becomes,

$$\text{tf-idf}\,(t, d) = \text{tf} \times \text{idf} \qquad (3.2)$$

A high weight in tf-idf is reached by a high term frequency in a document and a low document frequency of the term in the whole collection of documents. This will filter out the common terms across the corpus. For terms of more importance in certain algorithms, weights are also assigned i.e. tf score of important terms is multiplied by some integer to increase its weightage.

The tf-idf scoring is very effectively shown in Figure 3.2[1]. For each term of each document in the corpus, in this way the tf-idf score is obtained. A matrix is created to store these scores and then in the Text Mining algorithms these scores are applied. The matrix looks like the example shown in Table 3.1[2].

[1] The diagram has been taken from a very informative article by Dr. E. Garcia, "The Vector Space Model", http://www.miislita.com/term-vector/term-vector-3.html

[2] Ibid.

The actual scores are stored in text files and the Matlab or Visual Basic programs first creates the file and then reads and uses the scores in the programs for the different Text Mining algorithms implementation.

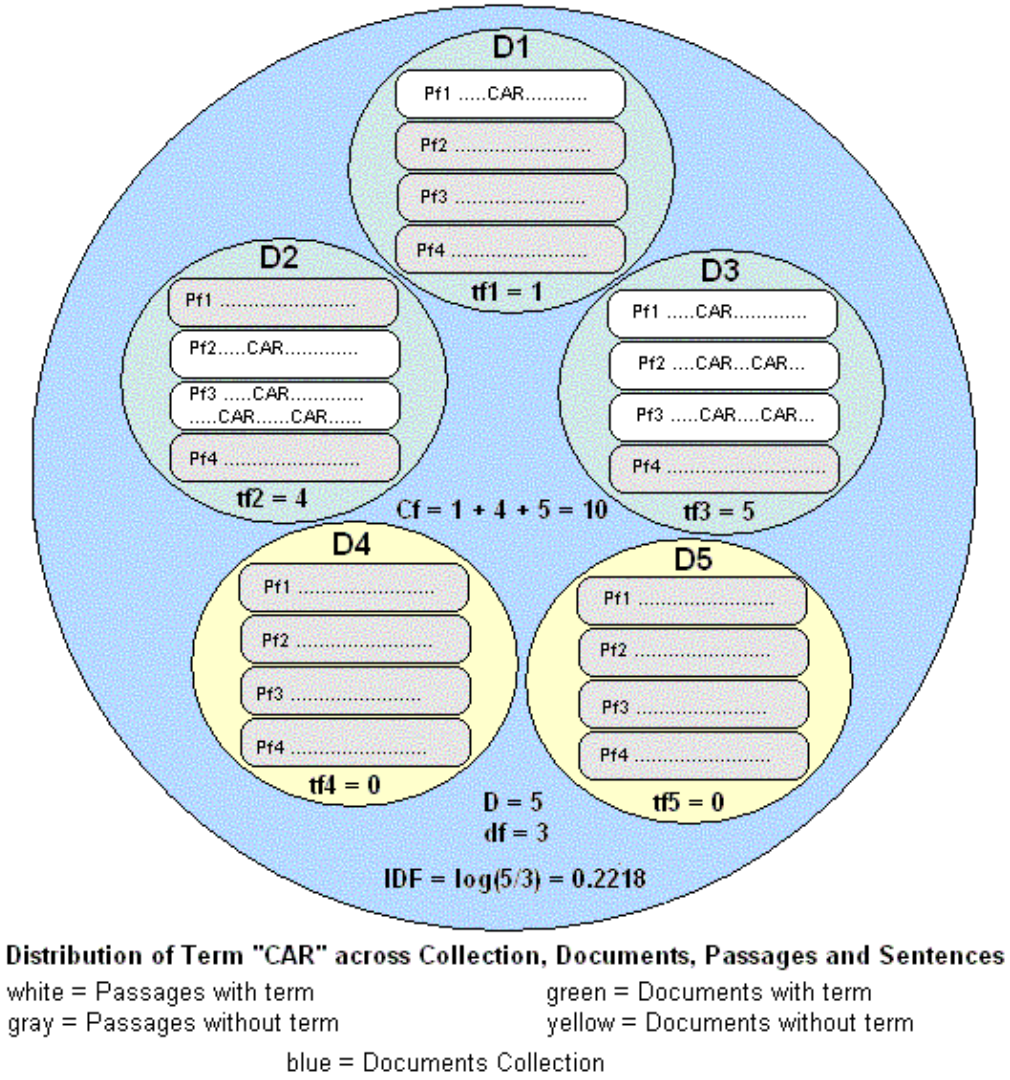

Figure 3.2 The term and document frequencies

As shown in the figure, the corpus is a collection of documents, documents consist of passages and passages consist of sentences. Thus, for a term i in a document j we can talk in terms of collection frequencies (Cf), term frequencies (tf), passage frequencies (Pf) and sentence frequencies (Sf).

Table 3.1 The tf-idf matrix example

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$											
Query, Q: "gold silver truck" D_1: "Shipment of gold damaged in a fire" D_2: "Delivery of silver arrived in a silver truck" D_3: "Shipment of gold arrived in a truck" D = 3; IDF = log(D/df_i)											
		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	**Q**	**D_1**	**D_2**	**D_3**	**df_i**	**D/df_i**	**IDF_i**	**Q**	**D_1**	**D_2**	**D_3**
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

3.2.3 Similarity Measures

As shown in Figure 3.1, to find the similarity between documents for text classification or text clustering it is necessary to find the distance between the documents. In the vector space model the most popular distance measure is the cosine similarity.

The cosine similarity

$$sim(d_1, d_2) = \frac{\overrightarrow{V}(d_1).\overrightarrow{V}(d_2)}{|\overrightarrow{V}(d_1)||\overrightarrow{V}(d_2)|} \quad (3.3)$$

Over here the numerator is the dot product and the denominator is the product of their Euclidean lengths. Dividing by the lengths will normalize the lengths of the documents to the unit length and this would give an accurate comparison now.

d1 and d2 are the documents that we are comparing. If the document vector for document d1 is:

$$\overrightarrow{V}(d_1)$$

Which has a dictionary size of M, the Euclidean length of this vector is given by:

$$\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(d)}$$

This will normalize the length of the vector i.e. the effect of the denominator of (3.3) is to length normalize the vectors. So the unit vector,

$$\overrightarrow{v}(d_1)$$

Will get the value,

$$\overrightarrow{V}(d_1) \; / \; \left|\overrightarrow{V}(d_1)\right|$$

Similarly for $\overrightarrow{v}(d_2)$.

We can rewrite (3.3) as,

$$sim(d_1, d_2) = \overrightarrow{v}(d_1).\overrightarrow{v}(d_2) \qquad (3.4)$$

Where the RHS is the dot product of two unit vectors now. Higher the value of similarity the more the documents are similar to each other. Based on this concept a term-document matrix as shown in Table 3.1 is created.

A query can also be considered as a short document with a few terms. The cosine similarity can be calculated between the query terms and the document terms and the top ranking documents can be selected as output. This is however very expensive and a document can have a high cosine score even if it does not have all the terms of the query – (if some of the terms the query occur a no. of times in the document the cosine similarity will increase)

The Jaccard's Co-efficient

This is another method of finding the similarity between two documents. The formula in (3.5) is the Jaccard's co-efficient for two documents di and dj.

$$\text{Similarity (doci, docj)} = \frac{|\,doci \cap docj\,|}{|\,doci\ U\ docj\,|} \quad (3.5)$$

Where,

$|\,doci\ U\ docj\,|$ = total number of distinct words in doci or docj

$|\,doci \cap docj\,|$ = total number of common words in doci and docj

The Dice's Co-efficient

Dice's coefficient, named after Lee Raymond Dice is given as follows:

$$sim(d_1, d_2) = \frac{2|d_1 . d_2|}{|d_1|^2 + |d_2|^2} \quad (3.6)$$

Where,

d1 and d2 are documents to be compared.

The Euclidean distance

$$Dist(d_1, d_2) = \left(\sum_{t=1}^{m} | w_{t,d1} - w_{t,d2} |^2\right)^{1/2} \quad (3.7)$$

Where,

T = {t1, … ,tm} is the term set (vocabulary)

As mentioned before, the term weights are the tf-idf scores,

$W_{t,d1}$ – tf-idf score of the t^{th} term in document d1.

Lower the value of the distance, closer are the documents.

3.2.4 Analysis of the Vector Space Model

The advantages of this model are:

- It is a simple model based on linear algebra
- The term weights not binary and dependent on term occurrence
- Allows computing a continuous degree of similarity between queries and documents
- Allows ranking documents according to their possible relevance

- Easy to implement and understand

The limitations of this model are:

- Long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality)
- In case of information retrieval, the search keywords must precisely match document terms; word substrings might result in a "false positive match"
- Semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match".
- The order in which the terms appear in the document is lost in the vector space representation
- Assumes terms are statistically independent
- Weighting is intuitive but not very formal

3.3 Latent Semantic Analysis (LSA)

3.3.1 Introduction to LSA

Latent Semantic Analysis is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is not a traditional natural language processing or artificial intelligence program; it uses no humanly constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, etc., and takes as its input only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs.

The first step is to represent the text as a matrix in which each row stands for a unique word and each column stands for a text passage or other context. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column. Next, the cell entries are subjected to a preliminary transformation in which each cell frequency is weighted by a function that expresses both the word's importance in the particular passage and the degree to which the word type carries information in the domain of discourse in general.

LSA is also known as Latent Semantic Indexing (LSI). This method uses a semantic approach for information retrieval. The results of the search will include terms which were not part of the query but similar to the meaning or close to the terms in the query.

3.3.2 Singular Value Decomposition

LSA applies singular value decomposition (SVD) to the matrix. This is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. In SVD a rectangular matrix is decomposed into the product of three other matrices. One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed. There is a mathematical proof that any matrix can be so decomposed perfectly, using no more factors than the smallest dimension of the original matrix. When fewer than the necessary number of factors are used, the reconstructed matrix is a least-squares best fit. One can reduce the dimensionality of the solution simply by deleting coefficients in the diagonal matrix, ordinarily starting with the smallest. (In practice, for computational reasons, for very large corpora only a limited number of dimensions can be constructed.)

3.3.3 Working of LSA

Singular value decomposition can be used in topic identification of documents. Using SVD, an m × n matrix, say X, is factored as: $X = U\sum V^T$ where U is m× t matrix, V^T is t × n matrix, and $\sum$ is a diagonal matrix of t × t. Here, we define matrix X as [doci] with one row per document, where X is n×d where n is the number of documents and d is the vocabulary size. This term-document matrix decomposes into: topic-document (U), topic-topic similarity ($\sum$) and term-topic (V^T). The topic-document matrix (U) is of importance to us since it represents the association between a document and a topic using which we identify the most prevalent topic in the document. Documents with same topic will lie in the same cluster. Hence the documents, which are highly

associated with the same topic, are clustered together. A document can have more than one topic.

3.4 Principal Components Analysis (PCA)

This method is also related to the SVD. It has wide applications like dimension reduction in information retrieval, image processing, pattern matching etc. The main aim of PCA is the reduction of high dimensional data set into a very low dimensional subspace.

It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data.

The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, i.e. by reducing the number of dimensions, without much loss of information. It is necessary to understand the statistical concepts of standard deviation and covariance as well mathematical concepts of eigenvectors and eigenvalues.

3.5 Attribute Selection

3.5.1 Introduction

Attribute selection, more popularly known as feature selection is the technique of selecting a subset of relevant features for building robust learning models in machine learning using statistical methods. It is also called variable selection, feature reduction or variable subset selection.

Many attribute / feature selection methods have been developed and extensive research work has already been done in this field. A brief summary of the methods available is given in section 3.5.2.

Feature selection is a process commonly used in machine learning, wherein a subset of the features available from the data is selected for application of a learning algorithm. The best subset contains the least number of dimensions that most contribute to accuracy; we discard the remaining, unimportant dimensions. This is an important stage of preprocessing and is one of two

ways of avoiding the curse of dimensionality – the other is feature extraction. It decreases the size of the effective vocabulary and increases accuracy of Text Mining by decreasing noise.

3.5.2 Comparison of Attribute Selection Methods

The most popular attribute selection methods are the frequency distribution, Mutual Information (MI), the chi-square test, correlation coefficient and relevancy score. The methods are all statistical based on probability distributions. The formulas of these methods are given in Table 3.2. The details about these methods are in the references section as per the reference numbers in the last column of the table.

Table 3.2 Main methods of feature reduction / selection

Function	Denoted by	Mathematical Form
Document Frequency	$\#(t_k, c_i)$	$P(t_k, c_i)$
Information gain (Expected Mutual Information)	$IG = (t_k, c_i)$	$P(t_k, c_i) \cdot \log\frac{P(t_k, c_i)}{P(c_i) \cdot P(t_k)} + P(\overline{t_k}, c_i \cdot \log\frac{P(\overline{t_k}, c_i)}{P(c_i) \cdot P(\overline{t_k})}$
Chi-square	$x^2(t_k, c_i)$	$\frac{g \cdot [P(t_k, c_i) \cdot P(\overline{t_k}, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t_k}, c_i)]^2}{P(t_k) \cdot P(\overline{t_k}) \cdot P(c_i) \cdot P(\overline{c_i})}$
Correlation coefficient	$CC(t_k, c_i)$	$\frac{\sqrt{g} \cdot [P(t_k, c_i) \cdot P(\overline{t_k}, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t_k}, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t_k}) \cdot P(c_i) \cdot P(\overline{c_i})}}$
Relevancy score	$RS(t_k, c_i)$	$\log\frac{P(t_k \mid c_i) + d}{P(\overline{t_k} \mid \overline{c_i}) + d}$

As per the research done by Martin Sewell[3], the different feature selection methods are as follows.

- Kira and Rendell (1992) described a statistical feature selection algorithm called RELIEF that uses instance based learning to assign a relevance weight to each feature.
- John, Kohavi and Pfleger (1994) addressed the problem of irrelevant features and the subset selection problem. They presented definitions for irrelevance and for two degrees of relevance (weak and strong). They also state that features selected should depend not only on the features and the target concept, but also on the induction algorithm. Further, they claim that the filter model approach to subset selection should be replaced with the wrapper model.
- Pudil, Novovi˘cov´a and Kittler (1994) presented "floating" search methods in feature selection. These are sequential search methods characterized by a dynamically changing number of features included or eliminated at each step. They were shown to give very good results and to be computationally more effective than the branch and bound method.
- Koller and Sahami (1996) examined a method for feature subset selection based on Information Theory: they presented a theoretically justified model for optimal feature selection based on using cross-entropy to minimize the amount of predictive information lost during feature elimination.
- Jain and Zongker (1997) considered various feature subset selection algorithms and found that the sequential forward floating selection algorithm, proposed by Pudil, Novovi˘cov´a and Kittler (1994), dominated the other algorithms tested.
- Dash and Liu (1997) gave a survey of feature selection methods for classification. In a comparative study of feature selection methods in statistical learning of text categorization (with a focus is on aggressive dimensionality reduction).

[3] The different feature selection methods as discussed by Martin Sewell, http://www.machine-learning.martinsewell.com/feature-selection

- Yang and Pedersen (1997) evaluated document frequency (DF), information gain (IG), mutual information (MI), a CHI-square test and term strength (TS); and found IG and CHI to be the most effective.
- Blum and Langley (1997) focused on two key issues: the problem of selecting relevant features and the problem of selecting relevant examples.
- Kohavi and John (1997) introduced wrappers for feature subset selection. Their approach searches for an optimal feature subset tailored to a particular learning algorithm and a particular training set.
- Yang and Honavar (1998) used a genetic algorithm for feature subset selection.
- Liu and Motoda (1998) wrote their book on feature selection which offers an overview of the methods developed since the 1970s and provides a general framework in order to examine these methods and categorize them.
- Weston, et al. (2001) introduced a method of feature selection for SVMs which is based upon finding those features which minimize bounds on the leave-one-out error.
- Xing, Jordan and Karp (2001) successfully applied feature selection methods (using a hybrid of filter and wrapper approaches) to a classification problem in molecular biology involving only 72 data points in a 7130 dimensional space.
- Forman (2003) presented an empirical comparison of twelve feature selection methods. Results revealed the surprising performance of a new feature selection metric, ‘Bi-Normal Separation’ (BNS).
- Guyon and Elisseeff (2003) gave an introduction to variable and feature selection. They recommend using a linear predictor of your choice (e.g. a 2 linear SVM) and select variables in two alternate ways: (1) with a variable ranking method using correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates.

Chapter 4: Text Clustering

4.1 Introduction to Text Clustering

Clustering is an unsupervised method of grouping texts / documents in such a way that in spite of having little knowledge about the content of the documents, we can group together similar documents into independent clusters based on some input parameters. In fact, given a training data set of documents, the goal of a clustering algorithm is to group similar documents in the same cluster while putting dissimilar documents in different clusters. Clustering is used in a wide variety of fields: biology, statistics, pattern recognition, information retrieval, machine learning, psychology, and Data Mining. For example, it is used to group related documents for browsing, to find genes and proteins that have similar functionality, to find the similarity in medical image database, or as a means of data compression.

Document clustering has been studied for quite a while and has wide applications like topic extraction, content filtering and also as a pre-processing step for text categorization. The indexing methods that I have used to rank documents have already been discussed in the previous sections. The measures that have been used in implementing the clustering algorithms are the tf-idf scores, singular value decomposition (svd), etc. and distance measure like the cosine similarity, Jaccard's co-efficient, etc.

4.2 Evaluation of Cluster Quality

There are different measures available to evaluate the correctness of clusters after an algorithm has been implemented. There are two types of measures – internal and external. Internal measures check the correctness within the clusters and across clusters i.e. how similar documents within a single cluster are and how different documents are across clusters.

The other more popular measures are the external measures. They are called external measures because we test the clusters on documents which have

been already classified (training sets) or those classified by human experts/judges which are called the gold standard classes.

There are many other external measures. Some are explained as follows[1]:

- Purity: This is a very simple method. To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N – total number of documents.

$$purity(\Omega, \mathrm{C}) = \frac{1}{N}\sum_k max_j \mid w_k \cap c_j \mid \qquad (4.1)$$

Where,

Ω = {w1, w2,…,wk} is the set of clusters

C = {c1, c2, …, cj} is the set of classes

w_k - the set of documents in w_k

c_j- the set of documents in c_j

Purity is close to zero for bad clusters and close to one for good clusters.

- The Rand Index (RI): The Rand index penalizes both false positive and false negative decisions during clustering. A true positive (TP) decision assigns two similar documents to the same cluster; a true negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A false positive (FP) decision assigns two dissimilar documents to the same cluster. A false negative (FN) decision assigns two similar documents to different clusters. The Rand index measures the percentage of decisions that are correct. That is, it is simply accuracy.

$$RI = \frac{TP+TN}{TP+FP+FN+TN} \qquad (4.2)$$

[1] These measures are taken from the book "An Introduction to Information Retrieval" by Christopher et al, online edition

- F-measure: This measure is based on recall and precision. The formula to calculate the F-measure for a cluster j and class i is:

  Recall(i, j) = nij / ni (4.3)

  Precision (i, j) = nij / nj (4.4)

  Where,

  nij – number of members of class i in cluster j

  ni – number of members of class i

  nj – number of members of class j

  F(i, j) = (2 * Recall(i, j) * Precision (i, j)) / (Recall(i, j) + Precision (i, j)) (4.5)

  After finding the F measure for all clusters in this way the overall F measure is computed by taking the weighted average of all values for F:

  $$F = \sum_i \frac{n_i}{n} \{F(i,j)\} \quad (4.6)$$

  Where,

  n – total number of documents

  Another way of representing the above is:

  $$Precision\ (P) = \frac{TP}{TP+FP} \quad (4.7)$$

  $$Recall\ (R) = \frac{TP}{TP+FN} \quad (4.8)$$

  $$F = \frac{2PR}{P+R} \quad (4.9)$$

Other than these measures there are others like the Mutual Information (MI) and the Entropy – both of which are based on probabilities.

The clustering algorithms studied are:

- The K-Means Algorithm
- The DBSCAN Algorithm
- The SNN
- Have developed SNNAE

4.3 The K-Means Algorithm

4.3.1 The simple K-Means

The K-Means is a partitioning method of clustering where n documents are partitioned into k partitions / clusters in such a way that each cluster has at least one document and each document belongs to only one cluster. The second condition is sometimes relaxed if we know that a document can belong to more than one topic or subject.

This is one of the simplest methods and creates clusters which are spherical in shape. This algorithm works well for a small corpus. The time complexity of this algorithm is linear in the number of documents. K-means is based on the concept that a center point can represent a cluster. In particular, for K-means we use the notion of a centroid, which is the mean or median point of a group of points (in this case documents). Note that a centroid almost never corresponds to an actual data point.

In the K-Means algorithm the input is the number of clusters k, the corpus containing the documents to be clustered and k initial arbitrary documents. The output contains k clusters of documents.

The algorithm works as follows:

1. Select arbitrarily K documents as the initial centroids.
2. Assign each document to the closest centroid using some similarity function.
3. Re-compute the centroid (mean) of each cluster.
4. Repeat steps 2 and 3 until the centroids do not change.

4.3.2 The Bisecting K-Means Algorithm

This algorithm is combination of K-Means and agglomerative hierarchical algorithm (uses the divisive method). This algorithm also has a complexity which is linear in the number of documents.

In this method we start with a single cluster which contains all the documents. The algorithm then splits the main cluster in different clusters as per the following algorithm:

1. Select a cluster to split
2. Find two sub-clusters from the cluster using the K-Means algorithm

3. Repeat step 2, the bisecting step, by selecting the largest cluster with least overall similarity.
4. Repeat steps 1, 2 and three until the desired number of clusters is reached.

4.3.3 The similarity measures

To find the distances between the documents and the centroids, I first calculated the tf-idf scores for each term in the documents. After that I created a term X document matrix for the tf-idf scores.

As per the details given in Table 2, for different values of k and initial cluster centroids, the clusters created are shown. The similarity measure used to find the similarity between the documents and the centroids was the cosine similarity.

$$(\text{doci}, \text{docj}) = \frac{\sum_1^{\mathbf{k}} (\mathbf{doci}^{(\mathbf{k})} \times \mathbf{docj}^{(\mathbf{k})})}{\|\mathbf{doci}\| \times \|\mathbf{docj}\|} \quad (4.10)$$

Where,

$$\|\text{doci}\| = \sqrt{\sum_1^k (\text{doci}^{(k)})^2}$$ (used to normalize the vectors)

doci & docj – are the tf-idf scores of the two documents

For our algorithm, we need to find the centroid c once iteration is complete. The centroid for a set of S documents with their vector representations can be found by,

$$c = \frac{1}{|S|} \sum_{d \, \epsilon \, S} d \quad (4.11)$$

This would be a vector obtained by averaging individual scores of all documents belonging to the set S. Finally the sets become the clusters.

4.4 The DBSCAN Algorithm

It stands for Density Based Spatial Clustering Algorithm with Noise. Though this is a density based algorithm it has been found to give very good results in text clustering also.

DBSCAN requires two parameters: epsilon (eps) and minimum points (minPts). In the textual data eps would be the value of cosine distance (between 0 and 1) and minPts generally works well for 4 (at least four documents are similar to the document under consideration). For the following algorithm, a point is a document.

1. It starts with an arbitrary starting point that has not been visited. It then finds all the neighbor points within distance eps of the starting point.
2. If the number of neighbors is greater than or equal to minPts, a cluster is formed. The starting point and its neighbors are added to this cluster and the starting point is marked as visited.
3. The algorithm then repeats the evaluation process for all the neighbors recursively.
4. If the number of neighbors is less than minPts, the point is marked as noise.
5. If a cluster is fully expanded (all points within reach are visited) then the algorithm proceeds to iterate through the remaining unvisited points in the dataset.

This algorithm has an advantage that it does not require to know the number of clusters in the data a priori and it can also detect noise. Moreover the clusters unlike K-Means are of arbitrary shape.

The disadvantage is it depends on the distance function. For structured low dimensional data the Euclidean distance is good enough but for the textual data we sometimes consider other similarity measures also. I have used the cosine similarity to implement DBSCAN. If documents of a particular class are few in number they would get classified as noisy documents.

4.5 The Shared Nearest Neighbor Algorithm (SNN)

The main difference between this algorithm and DBSCAN is that it defines the similarity between points by looking at the number of nearest neighbors that

two points share. Using this similarity measure in the SNN algorithm, the density is defined as the sum of the similarities of the nearest neighbors of a point. Points with high density become core points, while points with low density represent noise points. All remainder points that are strongly similar to a specific core points will represent a new clusters.

The steps to implement SNN are:

1. Identify the k nearest neighbors for each point (the k points most similar to a given point, using a distance function to calculate the similarity).
2. Calculate the SNN similarity between pairs of points as the number of nearest neighbors that the two points share. The SNN similarity is zero if the second point in not in its list of k nearest neighbors, and vice-versa.
3. Calculate the SNN density of each point: number of nearest neighbors that share Eps or greater neighbors.
4. Detect the core points. If the SNN density of the point is equal or greater than MinPts then classify the point as core.
5. Form the cluster from the core points. Classify core points into the same cluster if
   they share Eps or greater neighbours.
6. Identify the noise points. All non-core points that are not within a radius of Eps of a core point are classified as noise.
7. Assign the remainder points to the cluster that contains the most similar core point.

The SNN similarity takes the sum of the similarity of the point's nearest neighbors as a measure of density. It works well for the data in low, medium and high dimensionality. The SNN similarity measure reflects the local configuration of the points in the data space. It is insensitive to the variation of density and dimensionality. According to SNN density, the higher the density, it is likely to represent core or representative point and lower the density, it is likely to represent noise points. The key feature of SNN density measure is that it is able to find clusters of different shapes and sizes.

After studying the K-Means, DBSCAN and SNN algorithms I have designed a new algorithm which I call the SNNAE (Shared Nearest Neighbor Algorithm with Enclosures). This proposed algorithm has created better clusters and given a better output.

This algorithm has been published by the **World Research Congress and IEEE** and is available on IEEE Computer Society Portal, ACM Digital Library, Google Scholar, Bibsonomy, etc. and is available on the site: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5171034

4.6 The Shared Nearest Neighbor Algorithm with Enclosures (SNNAE)

The proposed algorithm, SNNAE, is based on the 'enclosure' approach, which uses an inexpensive distance measure to approximately divide the data into overlapping subsets and then applies expensive distance measure to calculate similarity only between the points in the same enclosure. The proposed algorithm is efficient and scalable because with this approach significant computation is reduced by eliminating all of the distance comparisons among points that do not fall within a common enclosure.

In the proposed algorithm, the density of the points is defined in terms of number of neighbors with which it shares a total number – Eps(a parameter) or greater neighbors. If this number is greater or equal to the MinPts(another parameter), then a point is consider to have high density otherwise it represents low density points. Also, the parameter Eps is calculated automatically from the enclosures and MinPts can be user specified or fixed to 3, which is shown as good value from many experimental results. The size of nearest neighborhood is also provided as input. The steps of the algorithm are as follows:

- **Creating Enclosures:** In the first stage, data points are divided into overlapping subset or enclosures where enclosure is simply a subset of the data points. Every data point must appear in at least one enclosure and any data point may appear in more than one enclosure as shown in

Figure 4.1[2]. Enclosures are created with the intention that points not appearing in any common enclosure are far enough apart that they could not possibly be in the same cluster.

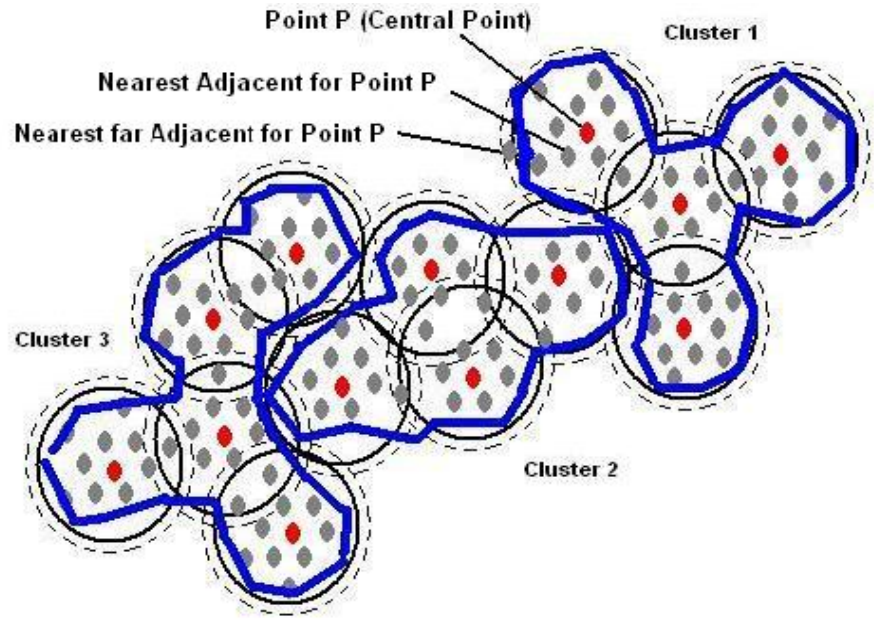

Figure 4.1 Three data clusters and enclosures

In the above figure, the solid circles show the example of overlapping enclosures that cover all the data points. Consider an arbitrary point p as shown in the figure. All the points inside the solid circle are the nearest adjacent points for the central point p. All the points between the dashed circle and solid circle are the nearest far adjacent points to the central point p. The dashed circle is used to ensure that points in the same clusters will not be split into different clusters. For e.g. there are no common points in two overlapping enclosures of cluster1. If dashed circles were not used then natural cluster1 would split into two small clusters.

In the second stage, the proposed algorithm finds the optimal value of Eps, the radius of neighborhood to define the density from the overlapped enclosures created in first stage and then apply the SNN clustering algorithm.

[2] A.M.Fahim et al. "Density Clustering Algorithm Based on Radius of Data (DCBRD)", Georgian Electronic Scientific Journal: Computer Science and Telecommunications, vol. 11, No.4, 2006.

To create overlapping subset or enclosure we first need to calculate radius of the enclosure. First of all, all the data points are stored in a single cluster called cluster feature (CF). This is a data structure which contains summary information about all points.

$$\overrightarrow{CF} = \left(n, \overrightarrow{LS}\right) \quad (4.12)$$

Where, LS is the linear sum of the n data points.

$$\overrightarrow{LS} = \sum_{i=1}^{n} \overrightarrow{x_i} \quad (4.13)$$

Where, xi is d-dimensional data points.

Then to calculate the radius of the data space which covers all the data point, we first find the centre of all the data point using the formula:

$$\overrightarrow{x_0} = \overrightarrow{LS}/n = \sum_{i=1}^{n} \overrightarrow{x_i}/n \quad (4.14)$$

Then radius of entire data set is calculated as:

$$R = \left(\sum_{i=1}^{n} (\overrightarrow{x_i} - \overrightarrow{x_0})^2/n\right)^{1/2} \quad (4.15)$$

From this radius, area of the circle is found as:

$$circular\ area = 3.14 * R^d \quad (4.16)$$

The circular area for more than two dimensions requires the 4/3 coefficient to be included in the formula. Then we calculate area from another point of view, called rectangular area, based on minimum bounding rectangle, which also covers all data point.

$$rectangular\ area = \prod_{i=1}^{d} L_i \quad (4.17)$$

Where, L_i is the difference between maximum and minimum value for the dimension i, which is also called as length of dimension i. In the Figure 4.2 blue points indicate length of x dimension and red points indicate length of y dimension.

In the multi-dimensional data set, as the dimension increase the data becomes more uniform and sparser. Therefore, while calculating radius of the overlapping enclosures, we have to consider dimension of the

data set. Also, radius depends on the area of the data space. This ratio should vary between 0 and 1.

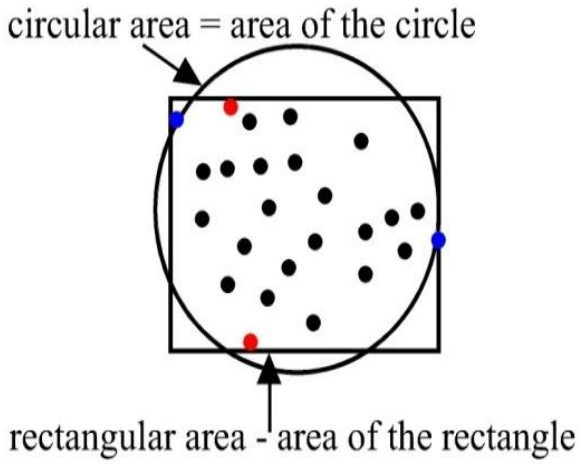

Figure 4.2 Circular and rectangular area of data space

By considering ratio area and dimension of the data space, the radius r of the overlapped enclosures is calculated as:

$$r = d * \text{ratio area} + \text{ratio area} / 2 \quad (4.18)$$

Where, d is the dimension of data space and ratio area is ratio of rectangular and circular area or inverse. Always, the radius of the overlapped enclosures is greater than expected Eps.

Overlapping enclosures are created using the radius r. To create enclosures, the first point of the data set is taken as the centre of the first enclosure. Then consider distance between every other point with that centre point. All the points whose distance to the centre point is less than or equal to radius r, are considered to be the nearest adjacent point to the centre point and are put in the list1 of that enclosure. If the distance to the centre point is less than or equal to r * 1.5 then points are considered to be as nearest far adjacent points and are put in list2 of that enclosure. All the points whose distance is greater than r * 1.5 and less than r * 2 are considered to be as centre of the next enclosure to ensure overlapping of enclosures.

For each point in the list1, the algorithm keeps the distance to the nearest enclosure and identification of that enclosure, since point may be covered by more than one enclosure.

- **Finding Optimal Value of Eps:** To find the optimal value of Eps, we consider only the points in the solid threshold. We find distance for each point with every other point within the same enclosure. These distances are calculated by using the following equation:

$$d(P_i, P_j) = (\sum_{k=1}^{d}(P_{i,k} - P_{j,k})^2)^{1/2} \quad (4.19)$$

  Where, d is the dimension, and Pi, k and P j, k are the k th component of the ith and jth object. From this distance, we find the maximum distance between the nearest pair. This process is repeated for all the enclosures. Then optimal value of Eps is found by taking average of all of these maximum distances, i.e. Eps = maxi / k, where k is the number of overlapped enclosures and maxi is the maximum distance for enclosure i. This Eps value is used to measure the density of the point in the proposed algorithm.

- **Finding Nearest Neighbor:** When the algorithm retrieves the nearest neighbors of a point, it directly goes to the best enclosure (i.e. nearest enclosure which covers that point.). It then computes the distance with every other point in enclosure (within solid threshold) and considers only those points that are having distance less than or equal to Eps as the nearest neighbors of that point. If size k, of the nearest neighbor list is given as an input then only k nearest neighbors are considered, otherwise all neighbors having distance less than or equal to Eps are considered as nearest neighbors of that point.

  If the distance between the centre point and point considered is greater than Eps then the algorithm computes the distance between that point and solid edge. If this distance is less than Eps value then we compute distance between the point considered and all the points in the dashed edge to find its nearest neighbors with respect to Eps. In this way the final clusters are created.

4.6.1 Complexity of the SNNAE Algorithm

The complexity of the algorithm can be found as follows:

Let,

n= number of records/instances/data points in data file

d= number of dimensions/ attributes

k= size of nearest neighbor list

e= number of enclosures

s= n/e i.e. average number of points in each enclosure

The proposed algorithm is divided into two stages.

1. In the first stage, n data points are divided into e enclosures. So its Complexity is O (ne)
2. The Eps value is calculated using only the points that are amongst the same enclosure. Assuming each enclosure covers s points on an average, calculation of Eps will have time complexity of (s2e). So its Complexity is O (s2e).

So, the time complexity of the first stage is:

O (ne +s2e)

In the second stage, getting the nearest neighbor, complexity:

O (ns),

Since distance of each point with every other point in the same enclosure is only calculated. Therefore, the total time complexity of the proposed algorithm is:

O (ne + s2e + ns)

4.6.2 The Dataset Description

The datasets used those already mentioned in the first chapter of Text Mining Overview. In the implementation of the algorithms, the datasets are db1, fish, abalone, and cpu. These datasets vary in size and number of attributes (dimensions). The site from where they can be downloaded is http://archive.ics.uci.edu/ml/ and the WEKA datasets. The brief description of each dataset used in the evaluation of this algorithm and the SNN algorithm is shown in Table 4.1.

Table 4.1 Details of datasets used

Dataset	Instances	Attributes	Data Type
Cpu	209	7	Real
Fish	1100	2	Synthetic
Abalone	4177	7	Real
Db1	10000	2	Synthetic

4.6.3 Implementation and result

The proposed SNNAE algorithm has been evaluated on several different real and synthetic datasets. The result of this algorithm is compared with that of SNN clustering algorithm in terms of scalability, efficiency and quality of clusters. Both algorithms produce the same result most of the time, but compared to SNN, SNNAE is more scalable and efficient.

The implementation shown is on the Abalone dataset. Abalones are small to very large-sized edible sea snails. The shells of abalones have a low and open spiral structure, and are characterized by several open respiratory pores in a row near the shell's outer edge. The flesh of abalones is widely considered to be a desirable food, and is consumed raw or cooked in a variety of different dishes. The attribute details are given in Table 4.2. The algorithms were implemented by varying the values of the input parameters and finding the output for each case. Some sample inputs and outputs are given further.

The parameter values calculated using the SNNAE algorithm for creating first the enclosures and then finding the Eps is depicted in Table 4.3. In Table 4.4 a comparison in terms of execution time in seconds between the SNN and the SNNAE algorithm at each step of execution is shown.

There are seven attributes and a total of 4177 record sets. The details of the attributes are:

Table 4.2 Attribute details of Abalone dataset

Name	Data Type	Description
Length	continuous	Longest shell measurement
Diameter	continuous	perpendicular to length
Height	continuous	with meat in shell
Whole weight	continuous	whole abalone
Shucked weight	continuous	weight of meat
Viscera weight	continuous	gut weight (after bleeding)
Shell weight	continuous	after being dried

Input: Dataset, Minimum points (Minpts), nearest neighbors (nnls)

Output: Clusters, noise points

Table 4.3 Value of different parameters

Radius of all data	0.489
Radius of enclosure	0.099
Number of enclosures	288
Eps	0.0538

Case No. 1: Considered Minpts = 3, and all possible nearest neighbours.

Table 4.4 Implementation Result for minpts-3, nnls-all

No.	Functionality	SNN time (sec)	SNNAE time(sec)
1	Find nearest neighbors	51.262	4.227
2	Find shared nearest neighbors	92.305	76.675
3	Get initial clusters	3.011	2.917
4	Get border and noise points	68.531	75.973
5	Compose cluster	0.375	0.764
Total		215.484	160.556

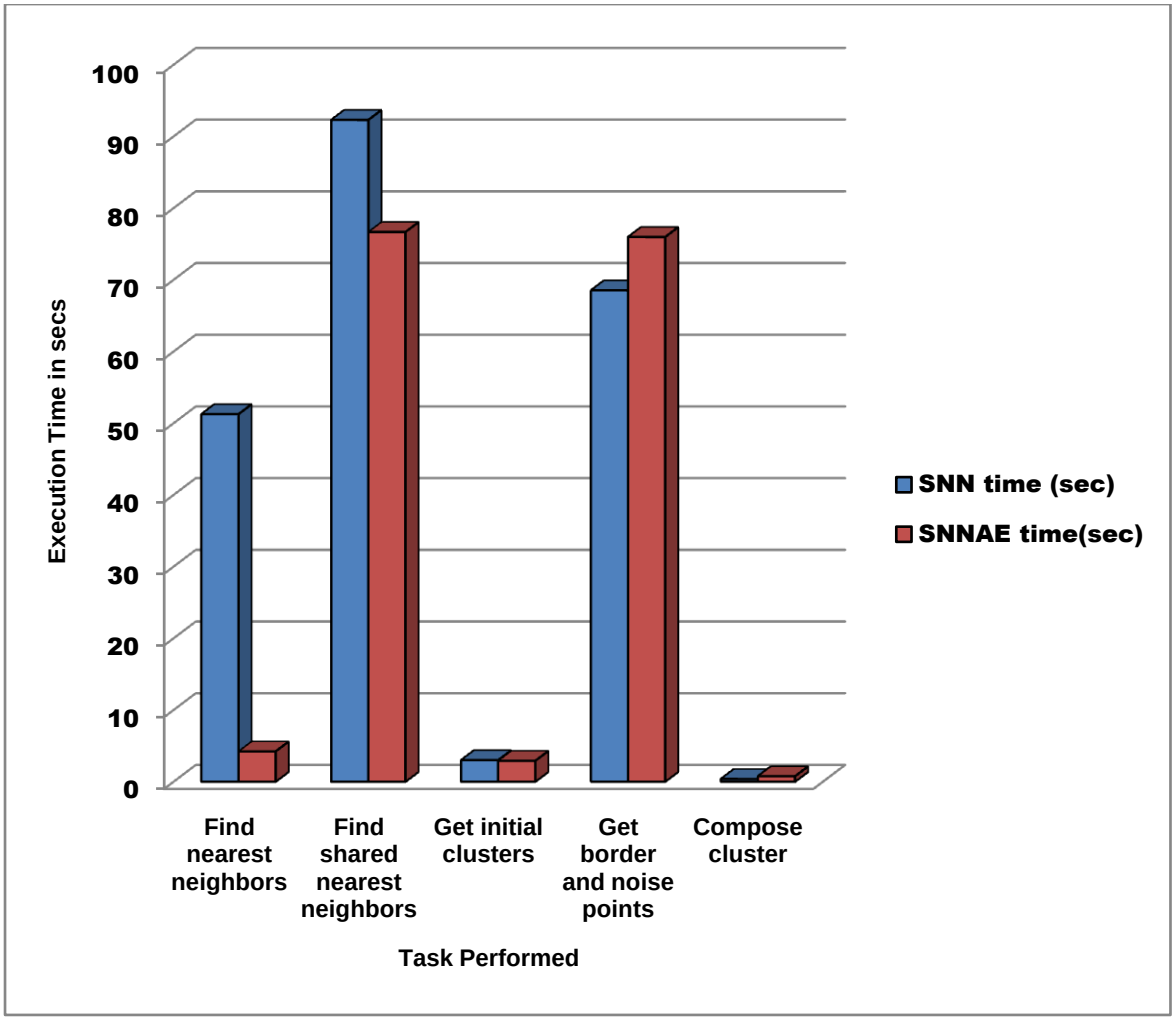

Figure 4.3 Implementation graph for minpts-3 and nnls-all

Table 4.5 Points distribution in clusters with minpts-3, nnls-all

Cluster No.	% of points with SNN	% of points with SNNAE
0 (noise)	14.29	15.21
1	85.08	84.17
2	0.31	0.31
3	0.19	0.19
4	0.12	0.12
Total	100	100

Case No. 2: Considered Minpts = 3, and nnls = 20

Table 4.6 Implementation Results for minpts-3, nnls-20

No.	Functionality	SNN time (sec)	SNNAE time(sec)
1	Find nearest neighbors	49.561	4.259
2	Find shared nearest neighbors	5.257	0.842
3	Get initial clusters	1.139	1.077
4	Get border and noise points	69.514	74.943
5	Compose cluster	0.468	0.811
Total		125.939	81.932

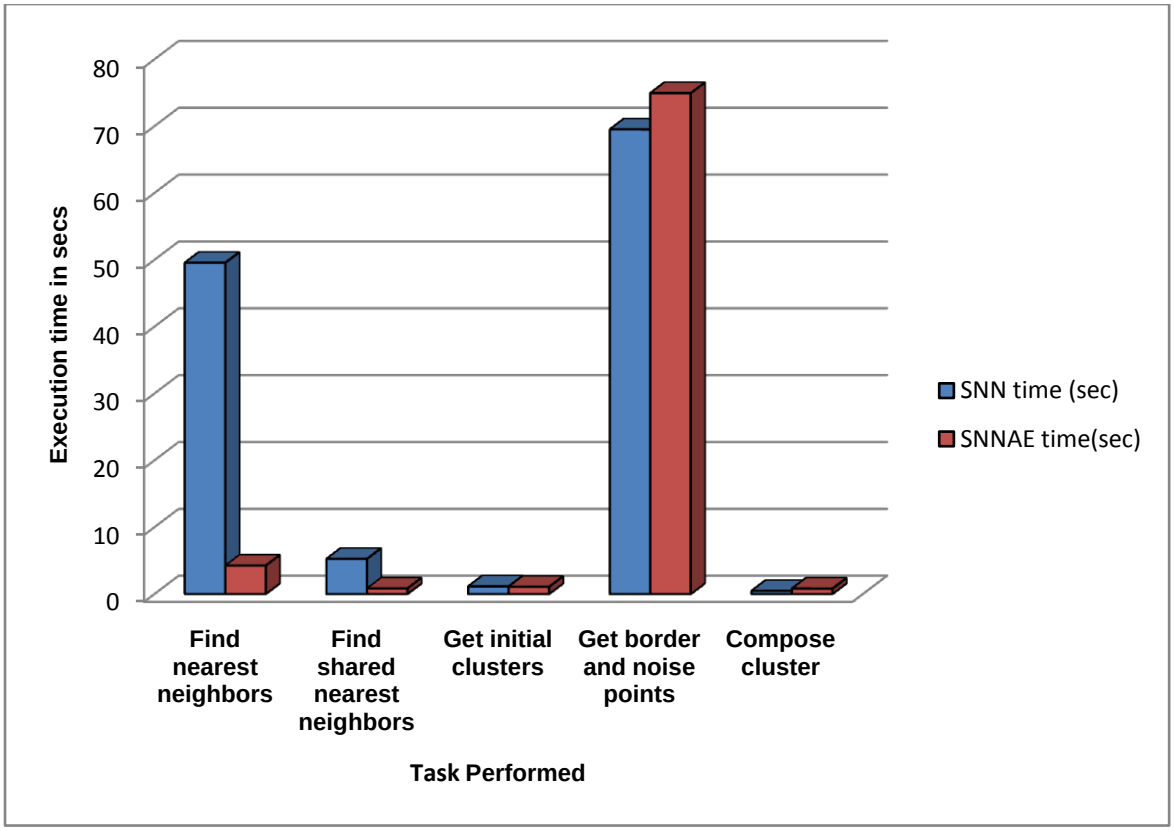

Figure 4.4 Implementation graph for minpts-3 and nnls-20.

Table 4.7 Points distribution in clusters with minpts-3, nnls-20

Cluster No.	% of points with SNN	% of points with SNNAE
0 (noise)	14.01	15.87
1	84.68	83
2	0.34	0.22
3	0.22	0.12
4	0.12	1
5	0.1	0.12
6	0.12	0.07
7	0.1	0.1
8	0.1	0.14
9	0.14	0.1
10	0.1	0.17
Total	100	100

Case No. 3: Considered Minpts = 3, and nnls = 25

Table 4.8 Implementation Results for minpts-3, nnls-25

No.	Functionality	SNN time (sec)	SNNAE time(sec)
1	Find nearest neighbors	49.546	4.29
2	Find shared nearest neighbors	7.129	1.326
3	Get initial clusters	1.498	1.404
4	Get border and noise points	65.957	69.031
5	Compose cluster	0.437	0.796
Total		124.582	76.847

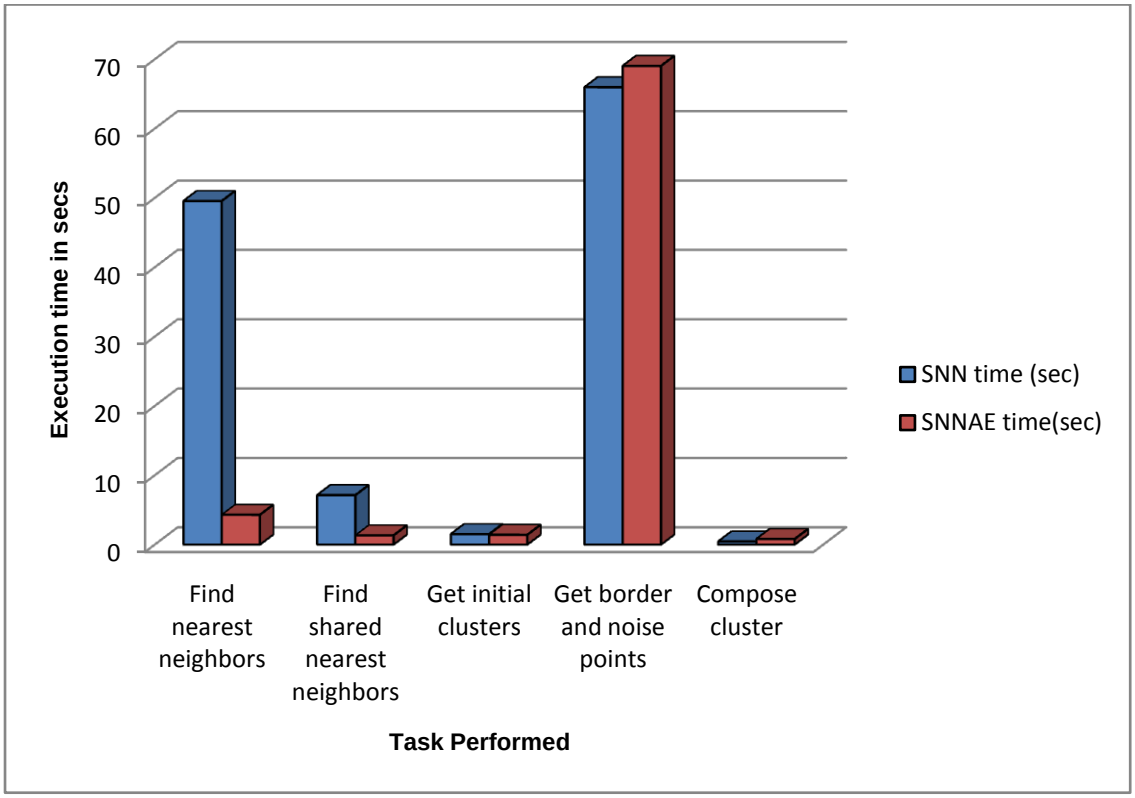

Figure 4.5 Implementation graph for minpts-3 and nnls-25

Table 4.9 Points distribution in clusters with minpts-3, nnls-25

Cluster No.	% of points with SNN	% of points with SNNAE
0 (noise)	13.48	15.44
1	85.2	83.43
2	0.34	0.22
3	0.22	0.12
4	0.12	1
5	0.1	0.12
6	0.12	0.07
7	0.1	0.1
8	0.1	0.14
9	0.14	0.1
10	0.1	0.17
Total	100	100

In this way two more cases were taken with minpts =4 and nnls = 20 and then 25. The consolidated result is as shown in Table 4.10.

Table 4.10 Combined Results of all cases

Experiment No.	Minpts	nnls	SNN Total time(sec)	SNNAE Total time(sec)
1	3	all	215.484	160.556
2	3	20	125.939	81.932
3	3	25	124.582	76.847
4	4	20	127.016	80.793
5	4	25	128.81	79.076

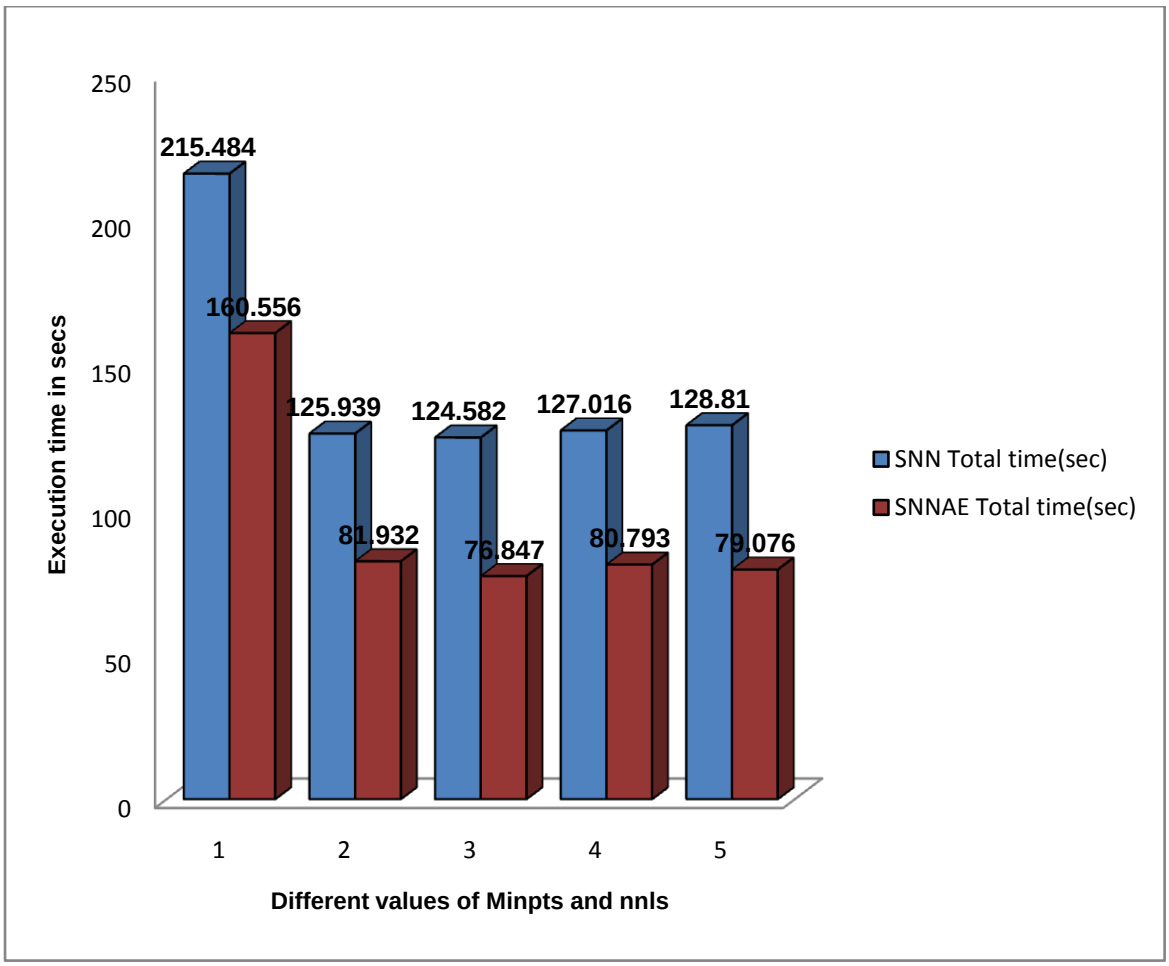

Figure 4.6 Implementation Graph for all cases

Similar results have been obtained for the rest of the datasets with structured as well as unstructured data. It is clearly evident that the time taken in finding the nearest neighbors and shared nearest neighbors is very less in SNNAE as compared to SNN. In finding the initial clusters and composing the clusters both take almost the same time. In finding the border and noise points however, SNN is slightly better than SNNAE. The overall time taken is by SNNAE is quite less as compared to SNN (Figure 4.6).

4.7 Conclusion and comparison of algorithms

The proposed algorithm, SNNAE is based on density, and k-nearest neighbor approach. The basic idea of this algorithm is to find the way of computing the nearest neighbors of points by restricting the number of points considered. This algorithm uses the enclosure approach to divide the data into overlapping region which greatly reduce the number of distance calculations required for clustering. This reduces the computational complexity of the SNN clustering algorithm which is O (n2) to O (ne + s2e + ns). The experimental results demonstrated the scalability and efficiency of the proposed algorithm. The algorithm has been tested against structured as well as unstructured data. The datasets can be downloaded from the sited mentioned in chapter 1.

The proposed algorithm provides a robust alternative to the other considered clustering approaches that are more limited in the types of data and clusters that they can handle.

Table 4.11 shows the comparison of the clustering algorithms like the K-Means, DBSCAN, SNN and SNNAE. The comparison is in terms of complexity, handling multidimensional data etc.

There are many clustering algorithms but the partitioning and hierarchical methods are more popular. Variants of the basic k-Means are also very popular. This chapter gives details of the popular algorithms being used in the field of Text Mining.

Table 4.11 Comparison of clustering algorithms

Clustering Criteria	K-means	DBSCAN	SNN	SNNAE
Complexity	O(nkt) where n = no. of data points k = no. of clusters t = no. of iterations	O(n2) where n is the number of data points	O(n2) where n is the number of data points	O (ne + s2e + ns) where n=no. of data points s= avg. number of points in enclosure e= number of enclosures
Handle multidimensional data	No	No	Yes	Yes
Handle large dataset	Yes	Yes	Yes	Yes
Handle Noise	No	Yes	Yes	Yes
Shape of Clusters	Spherical only	Any arbitrary shape	Any arbitrary shape	Any arbitrary shape
Types of data handled	Any	Any	Any	Any
Scalable	Yes	Not without enhancement because of its complexity	Not without enhancement because of its complexity	Yes
Input parameters	k – no. of clusters k initial cluster centroid	Eps – radius Minpts – no. of minimum points	k - size of nearest neighbor list MinPts Eps	Minpts – no. of minimum points Nnls – size of nearest neighbor list

4.8 Future Enhancement

There is a lot of scope in this field of Text Clustering. Methods like the fuzzy clustering are becoming popular as these methods apply the concept of fuzziness i.e. a three valued logic like true, false and maybe for a document to belong to a cluster. There are other hierarchical methods also where still there is scope for research. There are clustering methods related to neural networks also.

Document clustering is still not a very popular method in Information Retrieval. The reason being clustering is slow for large corpora. This can act as a domain of research too.

Chapter 5: Text Categorization

5.1 Introduction to Text Categorization

Text categorization (also known as text classification or topic spotting) is the task of automatically sorting a set of documents into categories from a predefined set. This task has several applications, including automated indexing of scientific articles according to predefined thesauri of technical terms, filing patents into patent directories, selective dissemination of information to information consumers, identification of document genre, authorship attribution, survey coding, and even automated essay grading.

Automatic text categorization can play an important role in a wide variety of more flexible, dynamic and personalized information management tasks as well: real-time sorting of email or files into folder hierarchies; topic identification to support topic-specific processing operations; structured search and/or browsing; or finding documents that match long-term standing interests or more dynamic task-based interests. Classification technologies should be able to support category structures that are very general, consistent across individuals, and relatively static.

There are two approaches that you can take:

- rule-based approach
    - write a set of rules that classify documents
- machine learning-based approach
    - using a set of sample documents that are classified into the classes (training data), automatically create classifiers based on the training data

The research on text categorization can be cast back to the work of M. E. Maron. From that time, the technique has being used to apply to information retrieval, document organization, and text filtering and so on. The schematic of the learning process for categorization is shown in Figure 5.1.

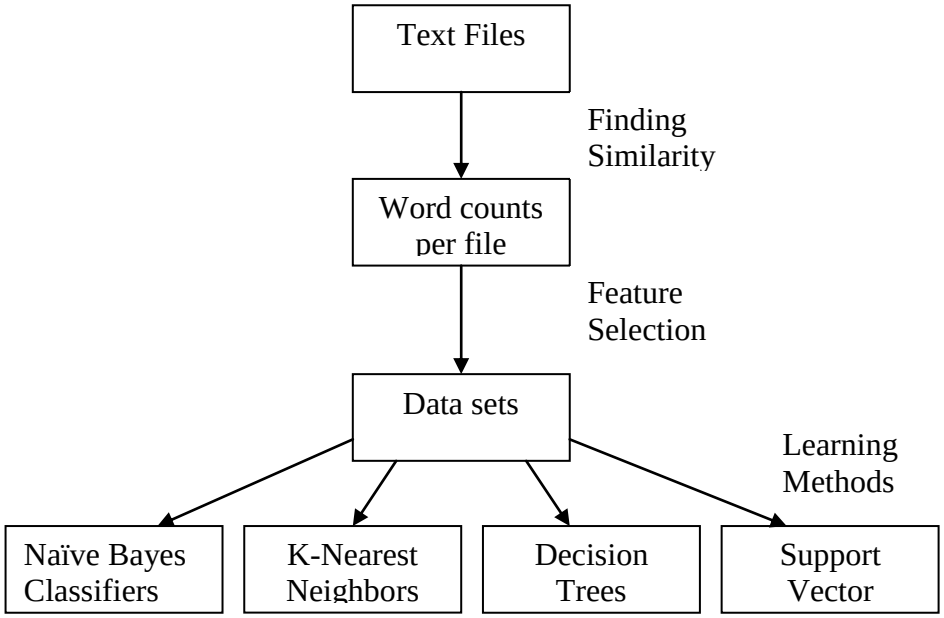

Figure 5.1 Schematic of learning process

There are broadly two steps in classification: developing the classifiers using the training dataset and then implementing them on the testing dataset. Sometimes the training set is itself divided into two parts where the first half is used to generate rules and the second half is used to check the validity of the rules. Since classification itself first needs the training datasets with the classes predefined, this part has to be done manually by a domain expert or can be done automatically by first using some text clustering method which does not require the domain knowledge, and then use the clusters that have been created as the training datasets with classes as defined by clustering.

Thus each class / label requires a set of rules based on which the test dataset can be classified. In machine learning approach these set of rules or text classifiers are automatically created from the training dataset. If the method used is a statistical one it is called statistical text classification. The test data can belong to one class or multiple classes. It depends on the assumptions used while developing the algorithm. I have implemented and studied four methods– Naïve Bayes, K-Nearest Neighbors and Decision Trees which are statistical methods. The fourth method support vector machine is a combination of statistics and mathematics.

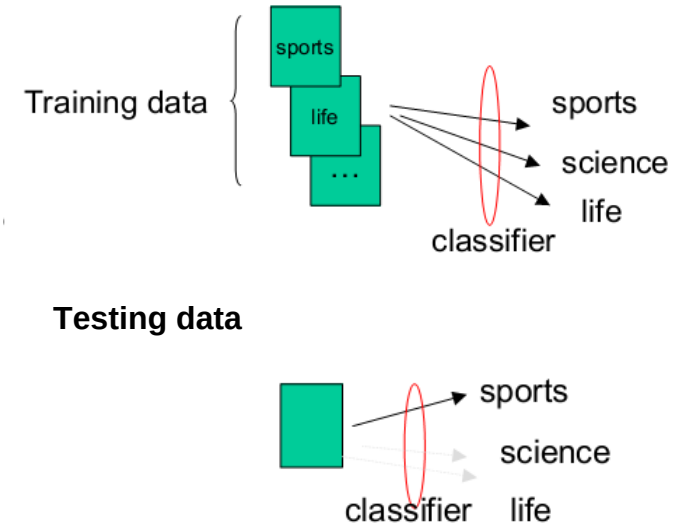

Figure 5.2 The training and testing datasets

5.2 The Evaluation Measures

Text classification rules are typically evaluated using performance measures from information retrieval. Common metrics for text categorization evaluation include recall, precision, accuracy and error rate and F1. Given a test set of N documents, a two-by-two contingency table with four cells can be constructed for each binary classification problem.

The cells contain the counts for true positive (TP), false positive (FP), true negative (TN) and false negative (FN), respectively. Clearly, N = TP + FP + TN + FN. I have used these parameters for my study. The metrics for binary-decisions are defined as:

- Precision(P) = TP / (TP + FP)
- Recall(R) = TP / (TP + FN)
- Accuracy = (TP + TN)/N
- Error = (FP + FN)/N
- F1 = 2(P*R) / P + R

F-measure (F1)

- harmonic mean of recall and precision
- sometimes instead of multiplying by the numerator by 2, other parameters like $\propto$ or β are also used where each one of them has some integer value.

Micro-average F1

- global calculation of F1 regardless of topics

Macro-average F1

- average on F1 scores of all the topics

The formulas to find the different F measures is given in Table 5.1.

Table 5.1 The F measures for Microaveraging and Macroaveraging

	Microaveraging	**Macroaveraging**
Precision(π)	$\pi = \frac{\sum_{i=1}^{\lvert c\rvert} TP_i}{\sum_{i=1}^{\lvert c\rvert} TP_i + FP_i}$	$\pi = \frac{\sum_{i=1}^{\lvert c\rvert} \pi_i}{\lvert c\rvert} = \frac{\sum_{i=1}^{\lvert c\rvert} \frac{TP_i}{TP_i + FP_i}}{\lvert c\rvert}$
Recall(ρ)	$\rho = \frac{\sum_{i=1}^{\lvert c\rvert} TP_i}{\sum_{i=1}^{\lvert c\rvert} TP_i + FN_i}$	$\rho = \frac{\sum_{i=1}^{\lvert c\rvert} \rho_i}{\lvert c\rvert} = \frac{\sum_{i=1}^{\lvert c\rvert} \frac{TP_i}{TP_i + FN_i}}{\lvert c\rvert}$

Where c represents the class.

These scores can be computed for the binary decisions on each individual category first and then be averaged over categories (macro-averaging). They can also be computed globally over all the n x m binary decisions where n is the total number of test documents and m is the number of categories under consideration (micro-averaging).

The micro-averaged F1 tend to be dominated by the classifier's performance on common categories whereas the macro-averaged F1 are more influenced by the performance of rare categories.

Due to the often highly unbalance number of positive vs. negative examples, note that TN often dominates the accuracy and error of a system, leading to miss-interpretation of the results. For example, when the positive examples of a category constitute only 1% of the entire test set, a trivial classifier that makes negative predictions for all documents has an accuracy of 99%, or an error of 1%. However, such a system is useless. For this reason, recall,

precision and F1 are more commonly used instead of accuracy and error in text categorization evaluations.

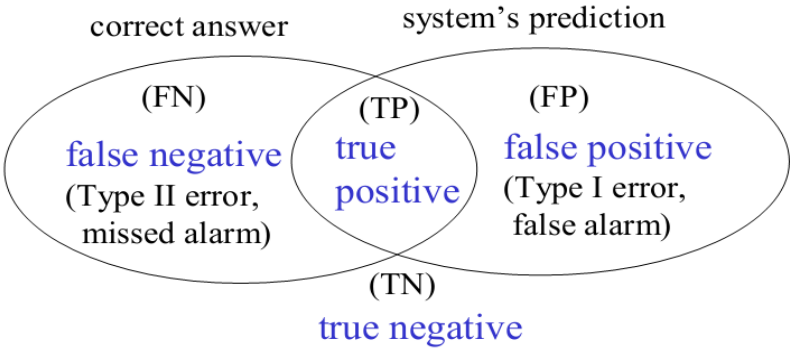

Figure 5.3 Common evaluation metrics

In multi-label classification, the simplest method for computing an aggregate score across categories is to average the scores of all binary task. The resulted scores are called **macro-averaged** recall, precision, F1, etc. Another way of averaging is to sum over TP, FP, TN, FN and N over all the categories first, and then compute each of the above metrics. The resulted scores are called **micro-averaged**. Macro-averaging gives an equal weight to each category, and is often dominated by the system's performance on rare categories (the majority) in a power-law like distribution. Micro-averaging gives an equal weight to each document, and is often dominated by the system's performance on most common categories. The two ways of measuring performance are complementary to each other, and both are informative.

A free-text document is typically represented as a feature vector x=(x(1),…,x(p)) , where feature values x(i) typically encode the presence of words, word n-grams, syntactically or semantically tagged phrases, Named Entities (e.g., people or organization names), etc. in the document. A standard method for computing the feature values x(i) for a particular document d is called the bag of words approach as discussed before.

It is useful to differentiate text classification problems by the number of classes a document can belong to. If there are exactly two classes (e.g. spam / non-spam), this is called a 'binary' text classification problem. If there are more than two classes (e.g. positive / neutral / negative) and each document falls into exactly one class, this is a 'multi-class' problem. In many cases,

however, a document may have more than one associated category in a classification scheme, e.g., a journal article could belong to computational biology, machine learning and some sub-domains in both categories. This type of text classification task is called a ‘multi-label’ categorization problem. Multi-label and multi-class tasks are often handled by reducing them to k binary classification tasks, one for each category. For each such binary classification tasks, members of the respective category are designated as positive examples, while all others are designated as negative examples. We will therefore focus more on binary classification.

The classification algorithms studied are:

- Naïve Bayes Classifiers
- k Nearest Neighbor
- Decision Trees
- Support Vector Machines

5.3 Feature Selection

The feature selection methods have already been discussed in the Chapter 3 on Text Transformation. In Text Classification applications, it is customary to run a dimensionality reduction pass before starting to build the internal representations of the documents. This means identifying a new vector space in which to represent the documents in such a way that the new vectors have a much smaller number of dimensions than the original ones. Several techniques for dimensionality reduction have been devised within Text Classification. An important class of such techniques is feature extraction methods (e.g., term clustering methods, latent semantic indexing). Feature extraction methods define a new vector space in which each dimension is a combination of some or all of the original dimensions; their effect is usually a reduction of both the dimensionality of the vectors and the overall stochastic dependence among dimensions.

An even more important class of dimensionality reduction techniques is that of feature selection methods, which do not attempt to generate new terms, but try to select the best ones from the original set. The measure of quality for a term is its expected impact on the accuracy of the resulting classifier. To

measure this, feature selection functions are employed for scoring each term according to this expected impact so that the highest scoring terms can be retained for the new vector space.

These functions mostly come from statistics (e.g., chi-square), information theory (e.g., Mutual Information), or machine learning (e.g., Information Gain), and tend to encode each in their own way the intuition that the best terms for classification purposes are the ones that are distributed most differently across the different categories.

5.4 Naïve Bayes Classification (NB)

NB algorithm has been widely used for document classification, and shown to produce very good performance. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. NB algorithm computes the posterior probability that the document belongs to different classes and assigns it to the class with the highest posterior probability. The posterior probability of class is computed using Bayes rule and the testing sample is assigned to the class with the highest posterior probability. The naive part of NB algorithm is the assumption of word independence that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category.

There are two versions of NB algorithm. One is the Bernoulli model that only takes into account the presence or absence of a particular term, so it doesn't capture the number of occurrence of each word. The other model is the multinomial model that captures the word frequency information in documents. Both the models have been implemented using the dataset described above. The detailed study and comparisons are given below. Both are based on the Bayes theorem, where a document d is placed in the class c as per the probability given below:

$$P(c/d) = \frac{\mathrm{P(d/c)P(c)}}{\mathrm{P(d)}} \qquad (5.1)$$

For classes the probability P(d) will remain same as it does not depend on any class and hence can be ignored. So, the posterior probability now becomes,

$$P(c/d) = \mathrm{P(c)P(d/c)} \quad (5.2)$$

Since we are dealing with terms within a document the posterior probability P(d/c), is actually P((t1,t2, …tn)/c) where each ti is a term or a feature of the document. Similarly for P(c/d). In the Naïve Bayes approach we assume each term occurrence is conditionally independent and that is why the name Naïve. It's a simple assumption because of which of posterior probability P((t1,t2, …tn)/c) now becomes,

$$P(d/c) = \mathrm{P((t1, t2, \dots, tnd)/c)} = \mathrm{P(t1/c)P(t2/c) \dots P(tnd/c)} \quad (5.3)$$

$$P(d/c) = \prod_{1 \le k \le n} \mathrm{P(tk/c)} \quad (5.4)$$

In the above formula,

$\mathrm{P(tk/c)}$ - is the conditional probability of term tk occurring in class c,

P(c) - is the prior probability of a document occurring in class c

t1,t2, … ,tnd - are the terms(tokens) of document d that are part of the vocabulary

nd – total number of tokens in the document

$\prod_{1 \le k \le n} \mathrm{P(tk/c)}$ - is the multiplication of conditional probabilities of all terms in the document

5.4.1 The Multinomial Model:

This model depends on the term counts in a document i.e. the number of times a term occurs in a document. The position of the term is not considered. As per (5.2) and (5.4), the posterior probability becomes,

$$P(c/d) = \mathrm{P(c)} \prod_{1 \le k \le n} \mathrm{P(tk/c)} \quad (5.5)$$

$\prod_{1 \le k \le n} \mathrm{P(tk/c)}$ can have many terms and the multiplication of the probabilities can result in floating point underflow. So a better option is to use logarithms. After applying logs, (5.5) becomes,

$$P(c/d) = \log \mathrm{P(c)} + \sum_{1 \le k \le nd} \log \mathrm{P(tk/c)} \qquad (5.6)$$

In the multinomial model to find the probabilities on the RHS of (5.5) we use the maximum likelihood estimate i.e. the relative frequencies of occurrences. As per this estimate, the probability P(c/d) can be calculated by using the following frequencies:

$$P(c) = \frac{\mathrm{Nc}}{\mathrm{N}} \qquad (5.7)$$

Where,

Nc – total number of documents in class c

N – total number of documents

$$P(t/c) = \frac{\mathrm{Tct}}{\sum_{t' \epsilon V} \mathrm{Tct'}} \qquad (5.8)$$

Where,

Tct – total number of occurrences of term t in documents of class c

$\sum_{t' \epsilon V} \mathrm{Tct'}$ - total number of terms in all documents of class c

In (5.8) the probability becomes zero if a term does not occur in a class and if it is applied to (5.5) it will become multiplication by zero. To eliminate this problem, Laplace's smoothing is used as follows:

$$P(t/c) = \frac{\mathrm{Tct+1}}{\sum_{t' \epsilon V} (\mathrm{Tct'+1})} = \frac{\mathrm{Tct+1}}{\sum_{t' \epsilon V} \mathrm{Tct'} + \mathrm{V}} \qquad (5.9)$$

$\sum_{t' \epsilon V} (1)$ - this is equal to V the size of the vocabulary.

The formula in (5.5) using (5.7) and (5.8) becomes,

$$P(c/d) \propto \frac{\mathrm{Nc}}{\mathrm{N}} . \prod \frac{\mathrm{Tct+1}}{\sum_{t' \epsilon V} \mathrm{Tct'} + \mathrm{V}} \qquad (5.10)$$

We can now find $P(c/d)$ the probabilities for the document for each class and the document will belong to the class where this probability is maximum. So our aim is to find max. $P(c/d)$.

5.4.2 The Bernoulli Model

In this model, unlike the multinomial model instead of counting the number of times terms/tokens occur in a document, the presence or absence of a term (0 or 1) is noted. The P(t/c) estimates the fraction of documents of class c that contain term t.

The implementation and analysis shows that the Bernoulli model works well for short documents only. The main drawback being that since just the presence of a term is noted, a document which contains a certain term only once may get classified in a class to which it actually does not depend.

This model is also based on the formula mentioned in (5.6) and (5.10). The difference lies in the estimation strategies. Unlike multinomial in the Bernoulli model the absence of a term is also modeled while computing the probabilities. The estimates for priors P(c) are similar to (5.7), whereas there is a little difference in estimates for (5.9) which is given below:

Tct – total number of documents of class c where term t occurs

$\sum_{t' \epsilon V} \mathrm{Tct}'$ - total number of documents of class c

V – two cases to be considered for each term i.e. occurrence or non-occurrence

The final equation of (10) now becomes,

$$P\left(\frac{c}{d}\right) \propto \frac{\mathrm{Nc}}{\mathrm{N}}.\ \left[P\left(\frac{t1}{c}\right).P\left(\frac{t2}{c}\right)\dots P\left(\frac{tx}{c}\right)\right].\left[1-P\left(\frac{ty}{c}\right).1-P\left(\frac{ty+1}{c}\right)\dots 1-P\left(\frac{tm}{c}\right)\right] \quad (5.11)$$

In (5.11), the terms t1… tx occur in the document whereas the terms ty … tm do not occur in the document. The document d is placed in the class which has the highest probability using (5.11).

5.4.3 Comparison of the Multinomial and Bernoulli Models

The comparison[1] between both the models considering different aspects in shown Table 5.2.

Table 5.2 Comparison between Multinomial model and Bernoulli model

Parameters	Multinomial Model	Bernoulli Model
Variable used	t – number of times a term occurs in a document	t – 1 if term occurs, 0 if it does not in a document
Document representation	d = {t1, t2, … , tn} where t1, t2,…tn terms are occurring in document d and are also part of V(vocabulary) – terms in d but not part of V are not considered	d = {t1, t2, … , tn} where each tk ϵ {0,1} – indicates the presence or absence of a term in the document d
Multiple term occurrences consideration	Yes	No
Efficiency (Document size)	Both short and long	Only short
No. of features handled	Efficient with more	Works best with few
Estimate for the term 'the'	P(X = the / c) 0.5	$P(U_{the} = 1 / c) = 1.0$

5.5 k-Nearest Neighbor (kNN)

kNN classifier is an instance-based learning algorithm that is based on a distance function for pairs of observations, such as the Euclidean, Cosine or Jaccard distance. In this classification paradigm, k nearest neighbors of a training data is computed first. Then the similarities of one sample from testing data to the k nearest neighbors are aggregated according to the class of the neighbors, and the testing sample is assigned to the most similar class. One of advantages of kNN is that it is well suited for multi-modal classes as its classification decision is based on a small neighborhood of similar objects (i.e., the major class). So, even if the target class is multi-modal (i.e., consists of objects whose independent variables have different characteristics for different subsets), it can still lead to good accuracy. A major drawback of the similarity measure used in kNN is that it uses all features equally in computing

[1] Based on the details in the book by Christopher Manning et al.,"An Introduction to Information Retrieval", Cambridge UP, 2009.

similarities. This can lead to poor similarity measures and classification errors, when only a small subset of the features is useful for classification.

The main concept behind kNN is that a test document is expected to have the same class as that of the training documents located in the local region surrounding it. The test document is assigned to the majority class of its k closest neighbors. It is important to decide the value of k because the accuracy of the classification is dependent on it. It is desirable to keep an odd value for k so that ties may not occur. One alternative way is to set the value of k in such a way that it gives the best result on the held out portion of the training set.

To decide whether a document di should be classified under class ck, first of all the k most similar documents to di are taken. These are the k-nearest neighbors of di. If a large proportion of them are classified under ck then di is classified under ck otherwise not.

The formula used for classification is:

$$\mathrm{y}(\mathrm{d_i}) = \arg \ max_k \ \sum_{x_j \epsilon kNN} y(x_j, c_k) \qquad (5.12)$$

Where,

di – document to be classified

xj – one of the neighbors of di

y(xj, ck) ϵ {0, 1} indicates whether document xj belongs to class ck

Unlike other classifiers, kNN does not classify documents in just two subspaces. This is an advantage over other methods.

There is another measure to classify documents using kNN. The system first finds the k nearest neighbors from amongst the training documents. The similarity score of each of these k documents to the test document is used as the weight of the categories of the neighbor document. The weights of the documents of the same category are added together and are considered as the likelihood score of the test document belonging to that category. After finding the likelihood scores for all categories falling in the k-neighborhood, the scores are sorted in ascending order and are ranked accordingly. By thresholding on these scores, the binary category assignments are obtained. The decision rule in kNN is:

$$\mathrm{y(d_i)} = \arg\ max_k \sum_{x_j \epsilon kNN} sim(d_i, x_j) y(x_j, c_k) \quad (5.13)$$

Where,

$sim(d_i, x_j)$ is the similarity measure between the test document di and neighboring document xj. This is generally the cosine similarity. The rest of the variables are as defined in (12).

As per this method the class with the maximum similarity is chosen as the class to which the test document will now belong. This definitely gives a better approximation as shown in the results in the next section. In the simple method discussed before the class of the top ranking category is assigned to the document but in that method the document belongs to only one category. Actually documents can belong to more than one category so depending on the similarity measures it can belong to more than one category.

5.6 The Novel kNN

Both the above methods work well when we assume that the number of documents in each class is equally distributed in the training sets. This may not be always true. If the number of training documents for some classes is much more than the rest then there are chances that these documents may get selected in the k nearest neighbors and the test document would automatically belong to the majority class instead of the actual class it belongs to.

To overcome this drawback I have designed an algorithm which I will now call 'The Novel kNN Algorithm'. In this proposed algorithm the selection of k nearest documents has been normalized and the algorithm will follow the following steps:

1. First select the n nearest neighbors from each class for the document di. The value of n should not be greater than the size of the smallest class. For e.g. the smallest training class contains 10 documents then n <= 10.

2. Sort these n nearest neighbors in descending order of the similarity - $sim(d_i, x_j)$.
3. Select now the top k documents from the list prepared in step 2. These are the final k nearest neighbors of document di now.
4. Using (5.13), now find the class to which di is most similar to and would belong.

Selecting the value of n is very important because we would not like to misclassify a document. This has been done by first splitting the training set into two – one to select the classes and apply the formula and other split to be experimented upon to check the validity of the algorithm. The implementation and results are displayed in section 5.11.
This algorithm has been submitted for publication in an International Journal.

5.7 Decision Trees

This form of classification uses a decision tree algorithm for creating rules. Generally speaking, a decision tree is a method of deciding between two (or more, but usually two) choices. In document classification, the choices are "the document matches the training set" or "the document does not match the training set."
A decision tree has a set of attributes (features) that can be tested. These can include: words from the document stems of words from the document (as an example, the stem of running is run) themes from the document (if themes are supported for the language in use) Decision trees are produced by algorithms that identify various ways of splitting a data set into branch-like segments. These segments form an inverted decision tree that originates with a root node at the top of the tree.
The object of analysis is reflected in this root node as a simple, one-dimensional display in the decision tree interface. The name of the field of data that is the object of analysis is usually displayed, along with the spread or distribution of the values that are contained in that field.
For each leaf of the decision tree, the decision rule provides a unique path for data to enter the class that is defined as the leaf. All nodes, including the

bottom leaf nodes, have mutually exclusive assignment rules; as a result, records or observations from the parent data set can be found in one node only. Once the decision rules have been determined, it is possible to use the rules to predict new node values based on new or unseen data. In predictive modeling, the decision rule yields the predicted value.

A decision looks like the one shown in Figure 5.4[2] where the documents are to be classified for the category 'Coffee'. The rules are binary presence (P) or absence (A) of an attribute.

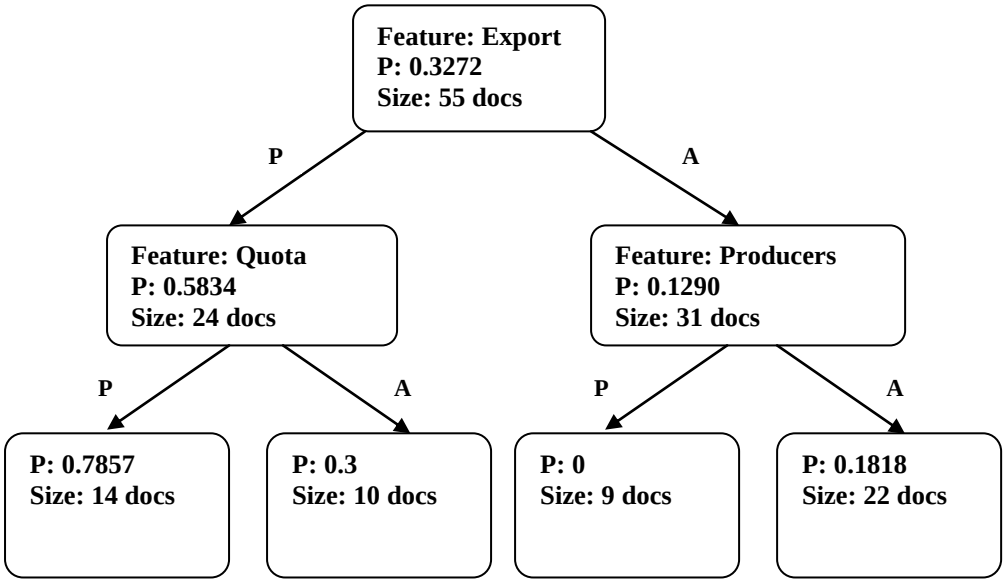

Figure 5.4 Decision tree

The test at each node in the Figure 5.4 above is binary – which is not necessarily the case always. It is possible that each node i.e. feature value falls in different ranges and takes more than two paths. The feature at each node determines the order of the rules.

To build a decision tree it is necessary to take care of the following criteria:

- The most discriminating feature is at the root of the tree – this can be found by selecting the feature that can best discriminate and classify a document (any of the feature selection methods described before can be selected)

[2] Have referred to the book 'Text Mining Application Programming' by Manu Kochady for decision trees – algorithm and figure.

- Find the information gain and entropy for the selected feature – higher the entropy lower is the information gain. Features with low information gain would not generate a good tree
- Select features which generally divide the documents in equal sizes
- Once all the features have been evaluated for their information gain a decision tree can be constructed as per the algorithm given below.

The following is a recursive algorithm (based on C4.5 algorithm) to generate a decision tree once all the training documents and the features involved are decided:

1. If all documents in the set of passed documents belong to one category, then return that category.
2. If there are no features with sufficient information gain for a branch decision, then return the most popular category among the documents
3. Choose the feature with the highest information gain.
    a. Build a sub tree of documents with the positive values of the feature and remove that feature from the current list of features.
    b. Build a sub tree of documents with the negative values of the feature and remove that feature from the current list of features.
    c. Recursively, call this algorithm with the sub trees and new feature list from steps a and b. Do these till no more features remain.
    d. Add two branches to the current tree with positive and negative values of the feature.
4. Return the tree

The above algorithm is for a binary tree but it can be modified for a general tree. Generating a decision tree is complex and sometimes over fitting may occur. To avoid this, the tree is pruned i.e. the leaf nodes with the least probability of classification are removed.

The measures used to decide the features are the I-Measure, Entropy and Information Gain.

5.8 Support Vector Machine (SVM)

A support vector machine (SVM) is a concept in computer science for a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Vapnik proposed SVMs in 1979, but they have only recently been gaining popularity in the learning community. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin – see Figure 5.5.

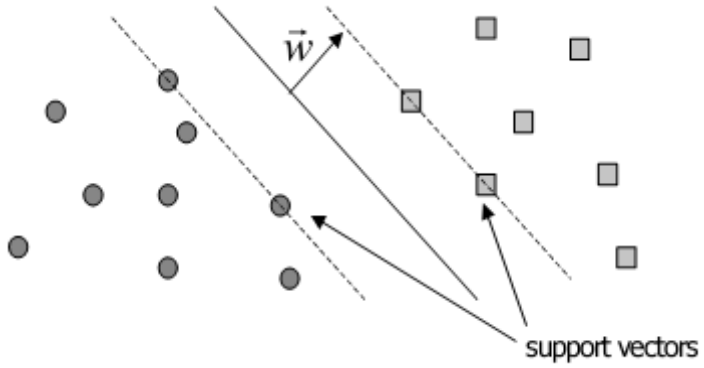

Figure 5.5 The linear SVM

The simplest linear version of the SVM gives good classification accuracy, is fast to learn and fast for classifying new instances[3].

[3] Detailed information about SVM is available from Vapnik, V., The Nature of Statistical Learning Theory, Springer-Verlag, 1995.

5.9 Dataset Description

The Reuters-21578 collection has been used for the above methods and it is a very popular one for text categorization research and is publicly available at: http://www.research.att.com/~lewis/reuters21578.html.

There are 12,902 stories that have been classified into 118 categories (e.g., corporate acquisitions, earnings, money market, grain, and interest). The stories average about 200 words in length. There are different splits available but I have used the ModApte[4] split in which 75% of the stories (9603 stories) are used to build classifiers and the remaining 25% (3299 stories) to test the accuracy of the resulting models in reproducing the manual category assignments. The stories are split temporally, so the training items all occur before the test items. The mean number of categories assigned to a story is 1.2, but many stories are not assigned to any of the 118 categories, and some stories are assigned to 12 categories. The number of stories in each category varied widely as well, ranging from "earnings" which contains 3964 documents to "castor-oil" which contains only one test document. Table 5.3 shows the ten most frequent categories along with the number of training and test examples in each. These 10 categories account for 75% of the training instances, with the remainder distributed among the other 108 categories. Table 5.4[5] gives the experimental results.

[4] Susan Dumais et al., "Inductive Learning Algorithms and Representations for Text Categorization ",Proceedings of the seventh international conference on Information and knowledge management, ISBN:1-58113-061-9 doi>10.1145/288627.288651

[5] Ibid

Table 5.3 Dataset Description – Training and Testing

Category	Training Set (total no. of docs)	Testing Set (total no. of docs)
Earn	2877	1087
Acquisitions	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	118
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	182	56

5.10 Implementation and result

The first step was pre-processing the documents (training as well as testing datasets). This has been done using MatLab 7 and Visual Basic 6:

1. Tokenizing
2. Removing stop words
3. Stemming (Porters Stemming)

For the training datasets:

1. Creating the vocabulary
2. Finding the term counts – document-wise, whole collection-wise
3. Creating a term by document matrix (one with term counts for multinomial, one with 0s and 1s for Bernoulli). The tf-idf matrix is also created.

The pre-processed data is now used to implement the algorithms. The results are shown in the Table 5.4 and graphical representation in Figure 5.6 given below. The breakeven point is the value at which precision and recall are equal or the ratio of precision to recall. The Naïve Bayes in the Table 5.4 is the multinomial model.

The comparison graph[6] of multinomial and Bernoulli for the same dataset but two categories interest and ship is shown in Figure 5.8. It is clearly visible that as the vocabulary size increases, the performance of Bernoulli model deteriorates.

Table 5.4 The breakeven performance for all categories

	Naïve Bayes (%)	Decision Trees (%)	Linear SVM (%)
Earn	95.9	97.8	98.0
Acquisitions	87.8	89.7	93.6
Money-fx	56.6	66.2	74.5
Grain	78.8	85.0	94.6
Crude	79.5	85.0	88.9
Trade	63.9	72.5	75.9
Interest	64.9	67.1	77.7
Ship	85.4	74.2	85.6
Wheat	69.7	92.5	91.8
Corn	65.3	91.8	90.3

To compare these methods in the algorithms discussed, the comparative between the implementation by three researchers[7] – Yang, Weiss and Joachims is given in Table 5.5 and graphically in Figure 5.7. The dataset is the same and values of the breakeven points are for all sets together. As seen from the table in most cases, support vector machine and k nearest neighbor (kNN) have better output.

[6] Andrew Mc Callum et al., "A comparison of event models for Naïve Bayes Classification"

[7] Shi Yong et al., "Comparison of Text Categorization Algorithm", Wuhan University Journal of Natural Sciences
Vol. 9 No. 5, 2004, pg. 798-804

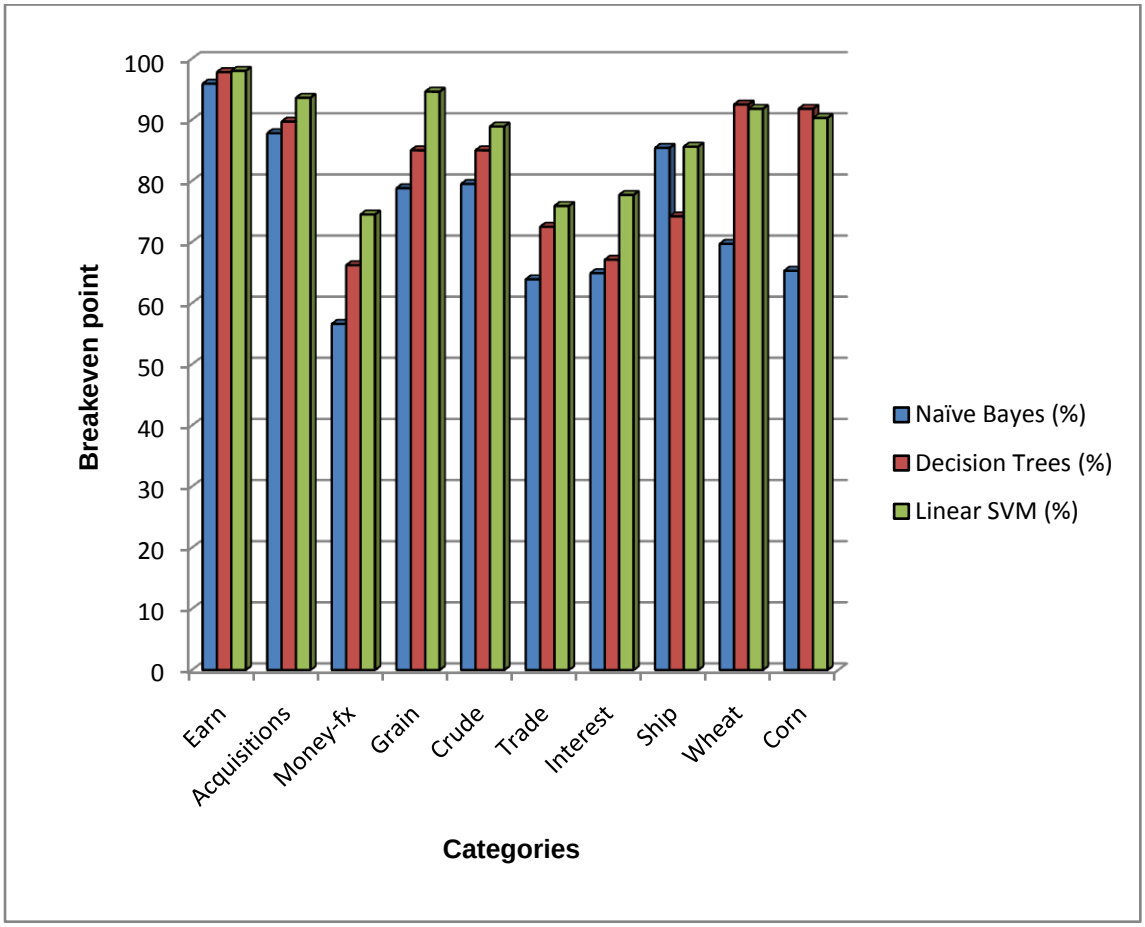

Figure 5.6 Comparison of the methods for all categories

Table 5.5 Comparative details for the algorithms (Breakeven points)

Researcher	Naïve Bayes	K Nearest Neighbors	SVM
Yang	71.5	85.0	85.9
Weiss	73.4	86.3	86.3
Joachims	72.0	82.3	86.0

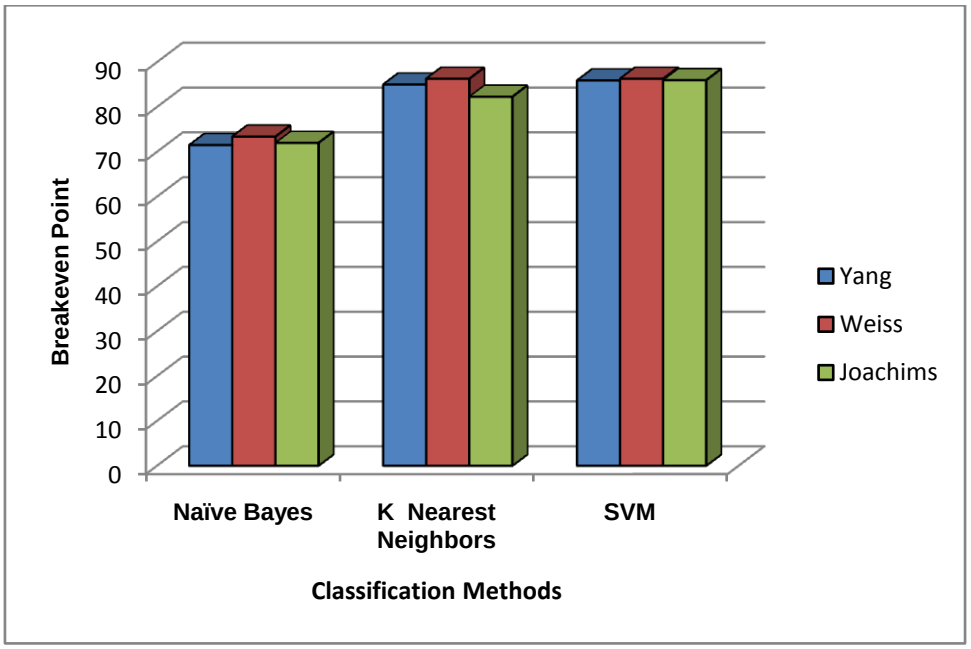

Figure 5.7 Comparison of the methods for three categories combined

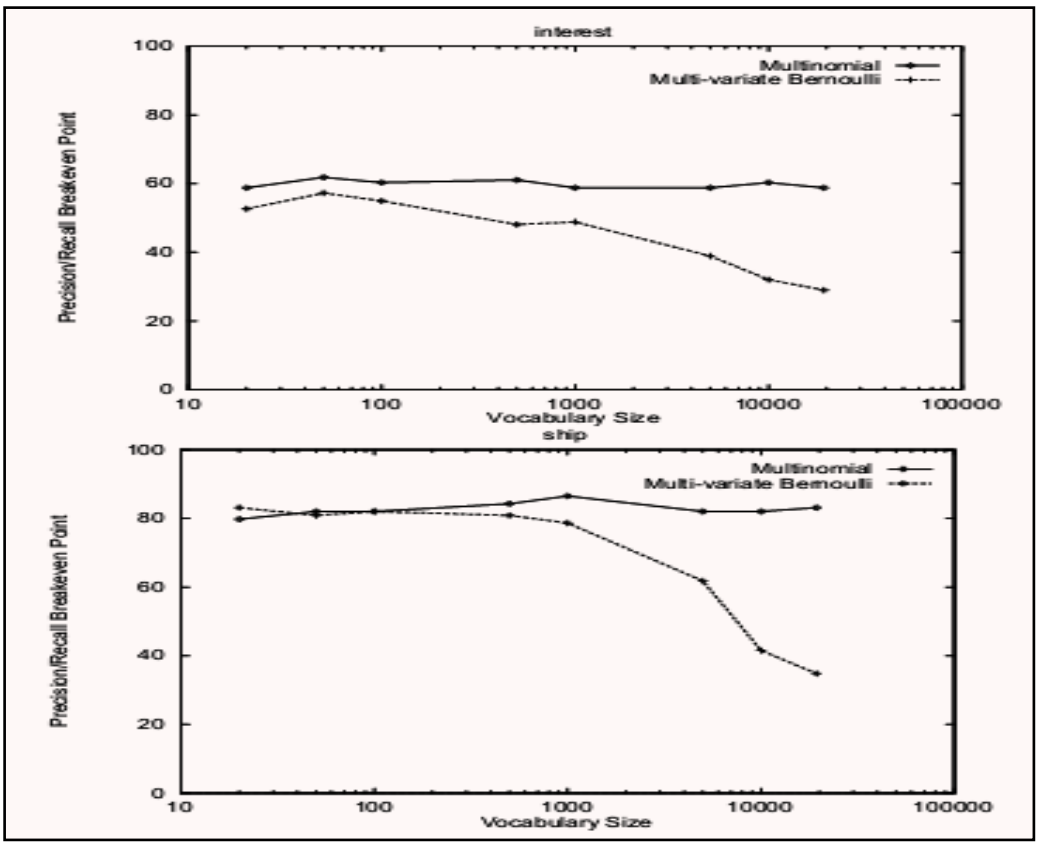

Figure 5.8 Comparison of Multinomial and Bernoulli models

5.11 The comparison between k-NN and Novel k-NN

Using the same datasets but first by taking a subset of it and using the holdback method (some part of training datasets being used as testing datasets) and varying the value of k, the output and the breakeven point for both the methods are given in Table 5.6. The categories considered are only the top 5 categories.

In case of the proposed algorithm – the Novel k-NN as already explained before, first the value of n is taken as input from the user which is value less than or equal to the size of the smallest class. After selecting n number of nearest neighbors from each class, they are sorted in descending order of their similarity and then the top k are selected. Though this method increases the number of iterations, the output is more accurate. As can be seen from the values, as k increases, the performance of Novel k-NN also improves as compared to the k-NN.

Table 5.6 Comparison of k-NN and Novel k-NN

	Breakeven Points	
Value of k	k - Nearest Neighbors	Novel k - Nearest Neighbors
5	67.95	66.50
10	68.50	67.45
20	65.60	66.75
30	63.54	66.90
40	60.80	65.50
50	59.25	64.20

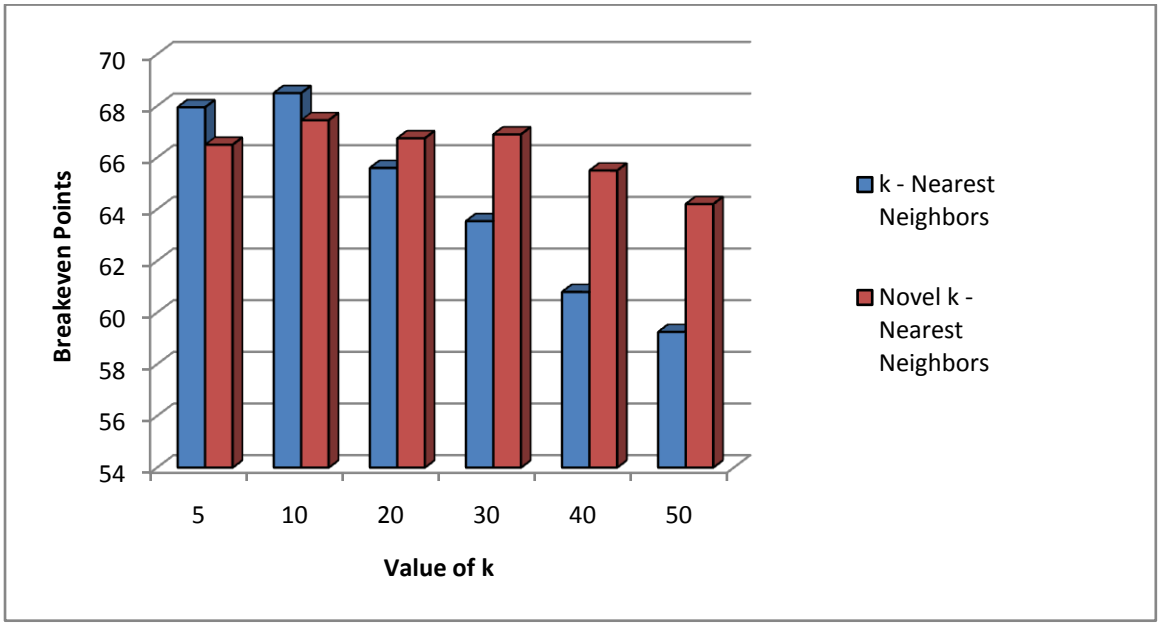

Figure 5.9 Comparison of k-NN and Novel k-NN

5.12 Future Enhancement

There are basically two frontiers for further research and development in this field. The first and foremost challenge is delivering high accuracy in all applicative contexts. While highly effective classifiers have been produced for applicative domains such as the thematic classification of professionally authored texts (such as newswires), in other domains reported accuracies are far from satisfying. Such applicative contexts include the classification of Web pages, where the use of text is more varied and obeys rules different from those of linear verbal communication; spam filtering, a task that has an adversarial nature in that spammers adapt their spamming strategies to circumvent the latest spam filtering technologies; and authorship attribution, in which current technology is not yet able to tackle the inherent stylistic variability among texts written by the same author. Though these areas have their own methods of classification it is still not fully developed and exploited – the methods used are a combination of NLP, image processing, AI, etc. and not just text processing.

A second important challenge is to bypass the document labeling bottleneck (i.e., labeling, or manually classifying, documents for use in the training phase is costly). To this end, semi-supervised methods have been proposed that allow building classifiers from a small sample of labeled documents and a usually larger sample of unlabeled documents (Nigam, McCallum, Thrun, & Mitchell, 2000). However, the problem of learning text classifiers mainly from unlabeled data is still open.

Another area of research is Bayes Network. It differs from the Naïve Bayes in the basic naïve assumption of word independence that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category. When this assumption is removed, comes the concept of Bayes Nets. BNs became extremely popular models in the last decade. They have been used for applications in various areas, such as machine learning, Text Mining, natural language processing, speech recognition, signal processing, bioinformatics, error-control codes, medical diagnosis, weather forecasting, and cellular networks.

Chapter 6: Text Summarization

6.1 Introduction to Text Summarization

Text Summarization is condensing the source text into a shorter version preserving its information content and overall meaning. It is very difficult for human beings to manually summarize large documents of text. The Internet normally provides more information than is needed. Therefore, a twofold problem is encountered: searching for relevant documents through an overwhelming number of documents available, and absorbing a large quantity of relevant information.

The goal of automatic text summarization is condensing the source text into a shorter version. Summaries may be classified by any of the following criteria:

- Detail: Indicative/informative
- Granularity: specific events/overview
- Technique: Extraction/Abstraction
- Content: Generalized/Query-based

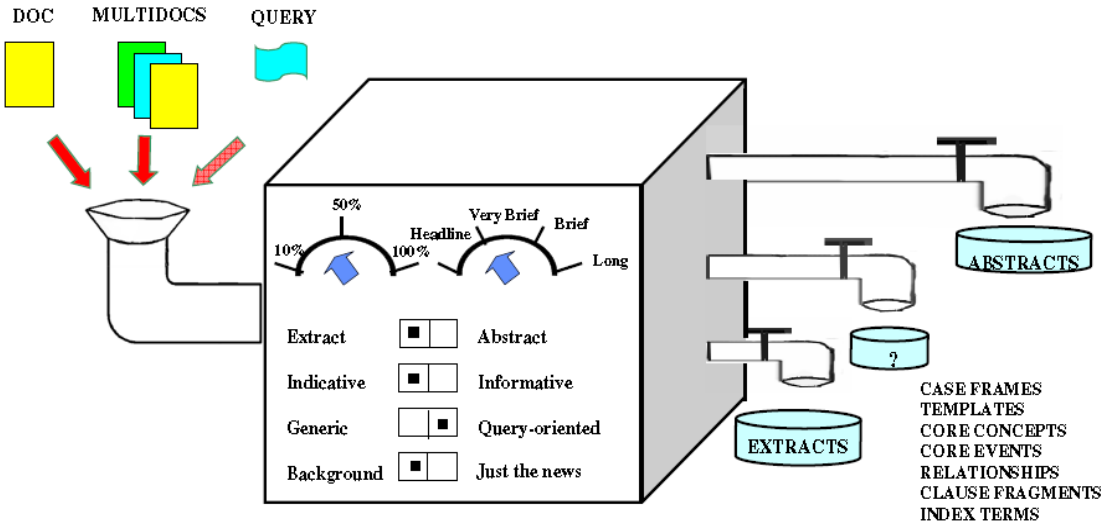

Figure 6.1 A Summarization Machine

An ideal summarization machine would look like the one shown in Figure 6.1. An indicative summary gives the main focus of the document and contains only a few lines whereas an informative summary is generally long and can be

read in place of the main document. The granularity decides the extent to which we want the summary to be broken into i.e. short, medium, detailed (specific event related or an overview) etc.

When the summary is the result of a query asked it becomes a query related otherwise it is a general summary. Topic-oriented summaries focus on a user's topic of interest, and extract the information in the text that is related to the specified topic. On the other hand, generic summaries try to cover as much of the information content as possible, preserving the general topical organization of the original text.

Text Summarization methods are more popularly classified into extractive and abstractive summarization. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences. An Abstractive summarization attempts to develop an understanding of the main concepts in a document and then express those concepts in clear natural language.

Text Summarization can be divided into the following areas:

- Selection based (tf-idf, ranking, etc.)
- Understanding based (syntactic analysis, semantic analysis)
- Information Extraction / Information Retrievel

The selection methods are more popular than the understanding based as the latter are connected to the Natural Language Processing (NLP). The extractive methods are based on tf-idf (term frequency-inverse document frequency), cluster based methods, the Latent Semantic Analysis (LSA) which is based on singular value decomposition or concept based summarization which is based on the vector space model. There are other methods also which are based on graphs**,** neural networks, fuzzy logic, regression, etc. Some of the above mentioned topics have been discussed earlier in the classification and clustering section.

The extraction based summarization methods studied are:

- Single document summarization
- Multi-document summarization
- Topic models

As part of further research in the field of abstraction, I have developed an algorithm based on the semantics i.e. a part-of-speech (POS) tagger. It is called the 'Multi-Liaison Algorithm' and it is explained in section 6.8.
For all types of summarization techniques the pre-processing steps generally remain same as discussed before. However there is a slight difference in text summarization. The details are given below.

6.2 Tokenization

Though tokenization is required to find the term frequencies, we store the sentences of the document separately and the weights assigned are to the sentences also. In some cases the position of the sentences is very important because the term weights depend on the position of the sentences in which the terms occur i.e. the title, the first paragraph, the last paragraph etc. These positions are given more weightage.
Removal of stop words and stemming remain the same.

6.2.1 Sentence Scoring

In extractive summarization it is important that from the document or set of documents we find out first which sentences are more important for the summary than the rest. This is possible only if some ranking / scoring is associated with them. There are four types of words which generally affect the sentence scores:

1. Cue words: These are the indicative words of the document which give some hint or analysis of the content like "summary", "reflects", "conclusion", "purpose" etc. These types of words are to be given more weightage.
2. Content Words (keywords): These are generally the nouns in sentences. Generally sentences containing proper nouns are considered important. These can also be the words which are acronyms, capitalized or italicized.
3. Title words: If a document has a title, generally the words in the title represent the main concept on which the document is based, so these words are important and are given extra weightage.
4. Location: the location of the sentence is very important. The first line and the last paragraph are more or less very important for the summary.

The sentence scoring has been done as follows:

$$Si = w1 * Ci + w2 * Ki + w3 * Ti + w4 * Li \quad (6.1)$$

Where,

Si – score of sentence i

Ci – score of sentence i based on cue words

Ki – score of sentence i based on keywords

Ti – score of sentence i based on title words

Li – score of sentence i based on its location

w1, w2, w3, w4 – are the weights assigned

In short, for document summary, score of a sentence is dependent on the frequency of the words in that sentence, their related weightage as per the details given above and the sum of it.

6.3 Single document summarization

Whenever summarization is be done it is necessary to know to what length the main document should be summarized (size of summary as compared to size of the document). This is also known as the compression rate. For example a 10 sentence document when compressed by 10% results in a one line summary.

Once each sentence is scored those sentences are ranked based on the descending order of their scores. Then depending on the compression rate the top sentences are selected as part of the summary.

6.4 Multi-document summarization

When summary is required from multiple documents, it is necessary that the documents are related to each other as far as the main content topics are concerned. In case we need to summarize multiple documents which are of mixed types, the first step is to applying text clustering on them so as to form clusters of same types of documents. Once these clusters are formed, for each cluster a separate summary can be generated.

Since the individual summary is generated from multiple documents belonging to a cluster, there is always a possibility that similar sentences from different

documents selected and repeated in the final summary. To make sure that the inter-sentence similarity is low, the following formula can be applied:

$$Cosine\ (ti, tj) = \sum_{h=1}^{k} t_{ih} t_{jh} \Big/ \sqrt{\sum_{h=1}^{k} t_{ih}^2 \sum_{h=1}^{k} t_{jh}^2} \quad (6.2)$$

Where,

i, j – the ith and jth sentences

ti, tj – term frequencies of ith and jth sentences

Depending on the similarity measures and compression ratio, top ranking but non-overlapping sentences are selected from multiple documents. The limitation in this method is the sequence in which the sentences from different documents would be displayed. This can be handled by noting the location of the selected sentences in its respective document (starting, middle, and end) and try to output each sentence as per its location.

Purely extractive summaries often give better results compared to automatic abstractive summaries. This is due to the fact that the problems in abstractive summarization, such as semantic representation, inference and natural language generation, are relatively harder compared to a data-driven approach such as sentence extraction. In fact, truly abstractive summarization has not reached to a mature stage today. Existing abstractive summarizers often depend on an extractive preprocessing component. The output of the extractor is cut and pasted, or compressed to produce the abstract of the text.

Limitations of Extractive Methods are:

- Extracted sentences usually tend to be longer than average. Due to this, part of the segments that are not essential for summary also get included, consuming space.
- Important or relevant information is usually spread across sentences, and extractive summaries cannot capture this (unless the summary is long enough to hold all those sentences).
- Conflicting information may not be presented accurately.

6.5 Comparison of Text Summarization methods

There are a number of different methods that have been developed for Text summarization and the base of these methods is either related to statistics, mathematics or NLP. A comparative is given in Table 6.1.

Table 6.1 Comparison between Text Summarization methods

Main concept of the method	Working	Method type
Tf-idf based summary	Based on simple heuristic features of the sentences: ➢ Position in the text ➢ The overall frequency of the words they contain ➢ Key phrases indicating the importance of the sentences ➢ A commonly used measure to assess the importance of the words in a sentence is the inverse document frequency	Extractive Method
Centroid-based summarization, a well-known method for judging sentence centrality and then selecting the sentences	The measures used are: ➢ Degree ➢ LexRank with threshold ➢ Continuous LexRank inspired from the prestige concept in social networks.	Extractive method
Lexical chains	➢ Basically lexical chains exploit the cohesion among an arbitrary number of related words ➢ Lexical chains can be computed in a source document by grouping (chaining) sets of words that are semantically related ➢ Identities, synonyms, and hypernyms / hyponyms (which together define a tree of “is a” relations between words) are the relations among words that might cause them to be grouped into the same lexical chain.	Abstractive method
A graph based representation	➢ A document cluster where vertices represent the sentences and edges are defined in terms of the similarity relation between pairs of sentences ➢ This representation enables us to make use of several centrality heuristics defined on graphs	A combination of Extractive and Abstractive methods

Maximum Marginal Relevance Multi Document (MMR-MD) summarization	➢ (MMR-MD) summarization is a purely extractive summarization method that is based on Maximal Marginal Relevance concept proposed for information retrieval ➢ It aims at having high relevance of the summary to the query or the document topic, while keeping redundancy in the summary low ➢ It can accommodate a number of criteria for sentence selection such as content words, chronological order, query/topic similarity, anti-redundancy and pronoun penalty	Extractive Method
Cluster based methods	➢ Documents are usually written such that they address different topics one after the other in an organized manner ➢ They are normally broken up explicitly or implicitly into sections i.e. themes ➢ If the document collection for which summary is being produced is of totally different topics, document clustering becomes almost essential to generate a meaningful summary.	Extractive Method
Latent Semantic Indexing	➢ This method uses the concept of the Singular Value Decomposition (SVD) ➢ The process starts with the creation of a terms by sentences matrix ➢ After applying the SVD as discussed before, the sentences with the highest index i.e. best sentences describing the salient topics of the text are selected	Extractive Method

The above is not an exhaustive list of methods but covers the most popular and commonly used ones. Variants of the above methods are also available. Some very good Text Summarization tools have been developed. They are:

MEAD

MEAD is a publicly available toolkit for multi-lingual summarization and evaluation. The toolkit implements multiple summarization algorithms (at arbitrary compression rates) such as position-based, Centroid, TF*IDF, and query-based methods. Methods for evaluating the quality of the summaries

include co-selection (precision/recall, kappa, and relative utility) and content-based measures (cosine, word overlap, bigram overlap).

MEAD v1.0 and v2.0 were developed at the University of Michigan in 2000 and early 2001. MEAD v3.01 – v3.06 were written in the summer of 2001, an eight-week summer workshop on Text Summarization was held at Johns Hopkins University. More details are available at:

http://www.clsp.jhu.edu/ws2001/groups/asmd.

SUMMARIST

This tool provides the abstracts and the extracts for English, Indonesian, Arabic, Spanish, Japanese etc. documents. It combines the symbolic world knowledge i.e. dictionaries like the WordNet and other lexicons as well as robust NLP processing techniques to generate the summaries. SUMMARIST is based on the following equation:

Summarization = topic identification + interpretation + generation

This tool is developed by the Natural Language Group at the University of Southern California.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation. Lin and Hovy's designed this package. For the inception of ROUGE, please refer Lin & Hovy's HLT-NAACL 2003 (Lin and Hovy 2003) paper.

SUMMONS

McKeown and Radev (1995) presented a system called SUMMONS which summarizes related news articles. The SUMMONS is a genre specific system which operates in the terrorist domain. The goal of the system is to generate fluent, variable–length summaries. SUMMONS is based on traditional language generation architecture and has two main modules for doing content planning and linguistic operations. The content planner consists of paragraph

planner and combiner. The linguistic component is made up of a lexical chooser, ontologizer and a sentence generator.

6.6 Sample Output of Text Summarizer

A sample output of the MEAD summarizer is given below. The input are three text files, the summary generated is compressed by first 10% and then 25%. File1 is of 1319, File2 of 1307 and File3 of1067 characters. Though the system is for Dutch, its gives a good output for English language.
Input to the summarizer were three files:

TEXT 1
I have assessed Ami in the lab assignments where I found that she has the potential of a very good programmer. She was also effectively involved in organizing university level technical event “Dwianki” where I was mentor for the same. I observed that she had the quality to work independently as well as in a group with equal ease. Her dedication to work for the best is substantiated by her excellent grades in all the courses I have handled. Considering her overall academic distinctions and achievements, I place her among top 5 % of the students associated with me in recent years. I am happy to see that she has decided to take her education to a higher level by pursuing a Masters degree at your Graduate School. She is a person with pleasing demeanor and has good communication skills. She always had the passion to learn new things and I am sure that she will continue to explore new horizons with the same zeal. I am confident that she will not only continue to be a promising and competitive student but would also be capable of efficiently discharging her roles as research / graduate assistant. I strongly recommend her for higher studies with deserving financial assistance. I feel that her academic proficiency and potential for research make her one of the truly outstanding candidates I have come across.
TEXT 2
In my course of interaction with him I have come to know Deepal as an exceptionally sincere and assiduous student. He has good understanding of theoretical aspects on one hand and its application to practical problems on

other hand. His lab work is consistent and he has performed exceedingly well in all his university lab examinations. This confirmed his capability of grasping the core concepts of the subjects and clear understanding of the basic principles. Deepal has mature personality and his attitude towards peer is co-operative and congenial. I have seen him produce very good results on complex projects that required great attention to details without compromising on the quality. ' Sparsh - Multi-touch Interaction System ' , final year project consisted of real time video processing and developing application to take the advantage of multi-touch sensing , which awarded Best project in two National level competitions . His keen analytic mind , systematic work habits , determination to pursue any chosen assignment to a successful conclusion provide an excellent blend of qualities required for successful pursuit of a graduate program . I am confident that given an opportunity, he will excel in his field of study. I therefore strongly recommend him for admission in your esteemed institution.

TEXT 3

As an Associate Professor of the Computer Science Department of The M S University, I can describe he has very good logical ability and intuitive thinking which makes him a very talented student. Moreover, he focuses more on conceptual learning and has the habit of thinking out of the box. He has excellent communication skills and always solves the problems assigned to him with a systematic and analytic approach. His diligence and dedication complement his good qualities. The quality I like the most about Nishant is that he is extremely humble and down to earth. He is an innovative thinker and is really good at writing. He has a good grasping power and his approach towards his work is always positive. He always has the attitude of learning from his mistakes. Apart from his studies, he actively participated in extracurricular activities and was the Training and Placement Coordinator of his batch. He was responsible for the campus recruitment of the Computer Science Department. Moreover, he always displayed good team spirit and was very supportive.

Output summary: (25%)

[1] I have assessed Ami in the lab assignments where I found that she has the potential of a very good programmer.

[2] I am happy to see that she has decided to take her education to a higher level by pursuing a Masters degree at your Graduate School.

[3] She always had the passion to learn new things and I am sure that she will continue to explore new horizons with the same zeal.

[4] In my course of interaction with him I have come to know Deepal as an exceptionally sincere and assiduous student.

[5] He has good understanding of theoretical aspects on one hand and its application to practical problems on other hand.

[6] As an Associate Professor of the Computer Science Department of The M S University , I can describe he has very good logical ability and intuitive thinking which makes him a very talented student .

[7] Moreover, he focuses more on conceptual learning and has the habit of thinking out of the box.

[8] He has excellent communication skills and always solves the problems assigned to him with a systematic and analytic approach.

Output summary: (10%)

[1] I have assessed Ami in the lab assignments where I found that she has the potential of a very good programmer.

[2] In my course of interaction with him I have come to know Deepal as an exceptionally sincere and assiduous student.

[3] As an Associate Professor of the Computer Science Department of The M S University , I can describe he has very good logical ability and intuitive thinking which makes him a very talented student .

[4] He has excellent communication skills and always solves the problems assigned to him with a systematic and analytic approach.

The screen shots are given below:

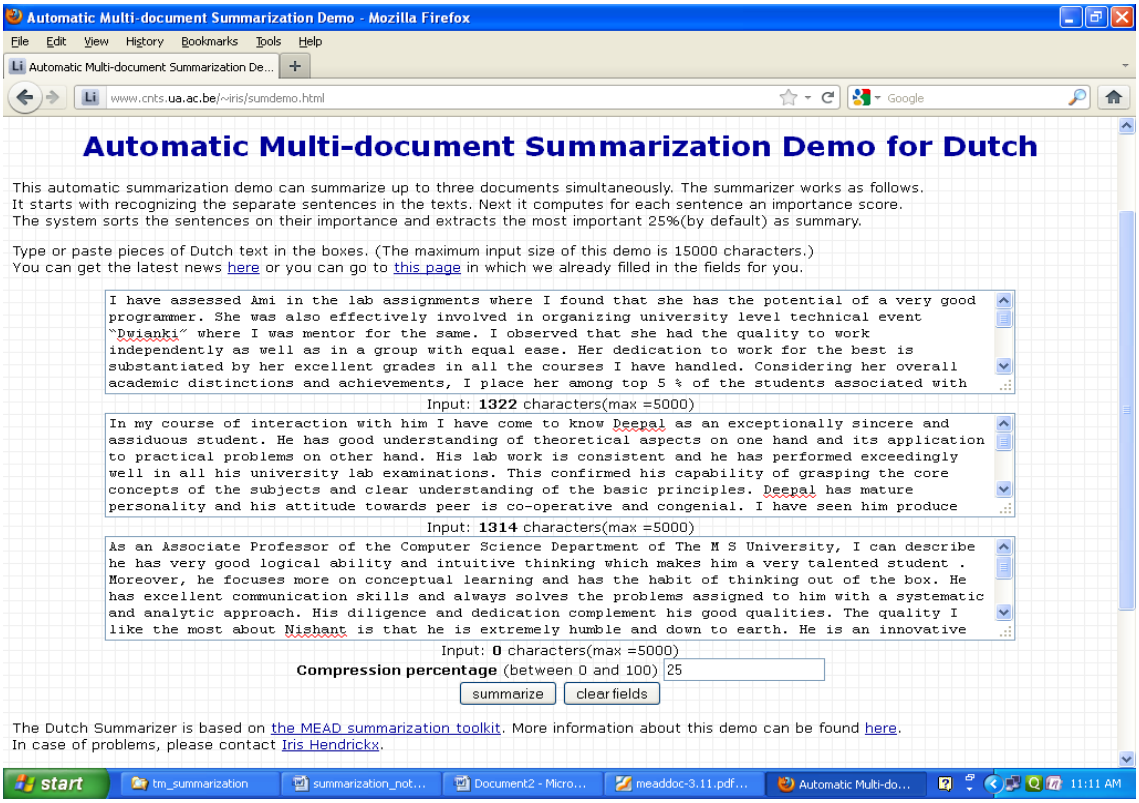

Figure 6.2 Screen shots of MEAD Summarizer

6.7 Topic Model

6.7.1 Introduction to Topic Model

A topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. An early topic model was probabilistic latent semantic indexing (PLSI), created by Thomas Hofmann in 1999. Latent Dirichlet allocation (LDA) is perhaps the most common topic model currently in use. The topic model is a statistical language model that relates words and documents through topics. It is based on idea that documents are made up of a mixture of topics, where topics are distributions over words. The Table 6.2 contains the conceptual comparison of various topic models.

With the increasing availability of other large, heterogeneous data collections, topic models have been adapted to model data from fields as diverse as computer vision, finance, bioinformatics, cognitive science, music, and the social sciences. While the underlying models are often extremely similar, these communities use topic models in different ways in order to achieve different goals.

Table 6.2 Conceptual comparison of various topic models

Model Name	Advantages	Disadvantages
Latent Semantic Analysis (LSA)	+ significant compression over simple tf-idf representation + Original high-dimensional vectors are sparse but the corresponding low-dimensional latent vectors will not be sparse, makes it possible to compute meaningful association between pairs of doc even no common term + can capture some aspects of basic linguistic notion such as polysemy and synonymy.	- Not capable of handling dynamic document collection - No generative model - No statistical standard methods - Output is not interpretable
k-means (cluster-model/mixture of unigrams)	+posses fully generative semantics	- document is considered to fall in a single cluster i.e. topic

Probabilistic Latent Semantic Analysis (pLSA)	+ consider the document to be made up of more than one topic	-No probabilistic model at the doc level (i.e. no generative model-how the document can be generated) which leads to very serious problem of : number of parameters grows linearly with the size of corpus. - How to assign the probability to document outside the training set is not defined. - No assumption about how the mixture weight θ is generated.
Latent Dirichlet Allocation (LDA)	+ provides a proper generative model + robust and versatile + domain knowledge is not required + as an unsupervised learning technique, human-intensive task of finding labeled examples for training set is completely eliminated.	-although, time and space complexity grows linearly with the number of documents, computations are only practical for modest-sized collections of up to hundreds of thousands of documents.

I have studied the topic model which is also a part of Text Mining in general and text summarization in particular. An approach called the Gibbs Sampling, a Markov Chain Monte Carlo method, is highly attractive because it is simple, fast and has very few adjustable parameters.

As part of the research, I have tried to derive a scalable algorithm which leads to reduction in the space complexity of the original Gibbs sampling for topic model. The concept used to reduce the space complexity is partitioning the dataset into smaller sets and then executing the algorithm for each partition. This reduces the space requirement without any impact on the time complexity. The enhanced Gibbs sampling algorithm has been implemented and experimented on four different datasets.

This work has been published in **'International Journal of Computer Information Systems' by Silicon Valley Publishers (UK)**, ISSN: 2229-5208, October 2011 issue and is available at:

http://www.svpublishers.co.uk/#/ijcis-oct-2011/4557969965. Before actually implementing the algorithm it was necessary to understand the LDA, the Gibbs Sampling and then propose a new approach. The step-wise and precise study and implementation is as given in the next section.

6.7.2 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures

over latent topics, where each topic is characterized by a distribution over words. The graphical model representation of LDA is shown in Figure 6.3.

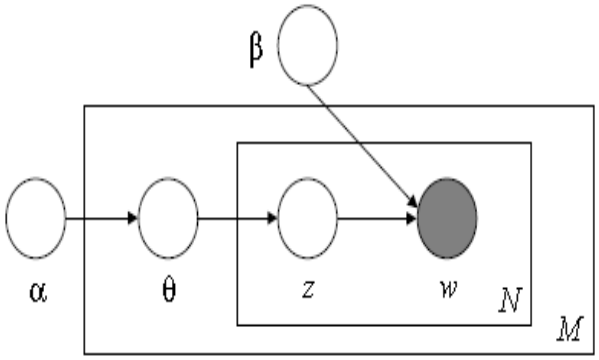

Figure 6.3 Graphical model representation of LDA

LDA assumes the following generative process for each document w in a corpus *D*:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the *N* words w_n:
   (a) Choose a topic $z_n \sim$ Multinomial(θ).
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

There are three levels to the LDA representation. The parameters and their significance are:

1. α and β are corpus level and are sampled once in the process of generating a corpus.
2. θ is document-level variable, sampled once per document.
3. *z* and *w* are word-level variables and are sampled once for each word in each document.

To compute the posterior distribution of the hidden nodes for a given document i.e. the inference to implement the LDA is given by:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta,z,w|\alpha,\beta)}{p(w|\alpha,\beta)} \quad (6.3)$$

The distribution is difficult to be estimated because of the denominator which is a normalizing constant.

The key idea behind the LDA model for text data is to assume that the words in each document were generated by a mixture of topics, where a topic is represented as a multinomial probability distribution over words.
The mixing coefficients for each document and the word topic distributions are unobserved (hidden) and are learned from data using unsupervised learning methods. Blei et al. introduced the LDA model within a general Bayesian framework and developed a variational algorithm for learning the model from data. Griffiths and Steyvers subsequently proposed a learning algorithm based on collapsed Gibbs sampling. Both the variational and Gibbs sampling approaches have their advantages: the variational approach is arguably faster computationally, but the Gibbs sampling approach is in principal more accurate since it asymptotically approaches the correct distribution.

6.7.3 Gibbs Sampling

Introduction

Gibbs sampling is an example of a Markov chain Monte Carlo algorithm. The algorithm is named after the physicist J. W. Gibbs, in reference to an analogy between the sampling algorithm and statistical physics. The algorithm was described by brothers Stuart and Donald Geman in 1984, some eight decades after the passing of Gibbs. As mentioned before Griffiths and Steyvers proposed the collapsed Gibbs sampling.

The Smoothed LDA

Before discussing Gibbs sampling it is necessary to understand how the LDA is smoothed[1] because of the problem with the original one. One problem that might arise with the original LDA model as shown in Figure 6.3 is that, the new document outside of training set is likely to contain words that did not appear in any of the documents in a training corpus, and zero probability would be assigned such words. To cope with the situation, the ‘smoothed’ model is shown in Figure 6.4. The strategy used is, not to estimate the model parameters explicitly, but instead considering the posterior distribution over the assignments of words to topics, P (z|w). The estimates of θ and Φ are then

[1] The detailed explanation of the smoothed LDA and the equations in given in the bibliography – [43] to [54].

obtained by examining this posterior distribution. Evaluating P (z|w) requires solving a problem that has been studied in detail in Bayesian statistics and statistical physics, computing a probability distribution over a large discrete space.

Here, α and β are hyper parameters, specifying the nature of the priors on θ and Φ. Although these hyper parameters could be vector-valued, for the purposes of this model we assume symmetric Dirichlet priors, with α and β each having a single value. These priors are conjugate to the multinomial distributions θ and Φ, allowing us to compute the joint distribution P (w, z) by integrating out θ and Φ.

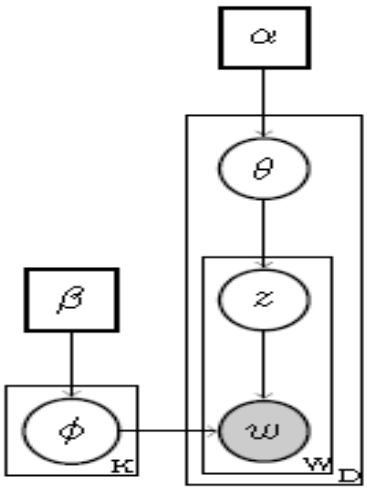

Figure 6.4 Graphical model representation of smoothed LDA

6.7.4 The Gibbs Algorithm for LDA

After applying a number of steps to the equation 6.3, the conditional distribution as mentioned by Griffiths et al.[2] is:

$$p(z_n = t|z_n, w) \;\alpha\; \frac{(\beta+C_{wt}^{-n})\;\;(\alpha+C_{td}^{-n})}{(C_t^{-n}+W\beta)\,(C_d^{-n}+T\alpha)} \qquad (6.4)$$

The different terminology used in the equation 6.4 is given in Tab. 6.3.

[2] The detailed derivation of the formula is in the work by Griffiths, T.L., and Steyvers, M., "*Finding Scientific Topics*", National Academy of Sciences, 101 (suppl. 1) 5228–5235, 2004.([45], [46]).

Table 6.3 Terms and their meanings for equation 6.4

Term	Meaning
C_{wt}^{-n}	Number of instances of word w assigned to topic t, not including current one
C_{t}^{-n}	Total number of words assigned to topic t, not including current one
C_{td}^{-n}	Number of words assigned to topic t in document d, not including current one
C_{d}^{-n}	Total number of words in document d not including current one

Having obtained the full conditional distribution, the Gibbs Sampling algorithm is then straightforward. The z_n variables are initialized to values in {1, 2 . . . T}, determining the initial state of the Markov chain. The chain is then run for a number of iterations, each time finding a new state by sampling each z_n from the distribution specified by the equation 6.4. After enough iterations for the chain to approach the target distribution, the samples are taken after an appropriate lag to ensure that their autocorrelation is low. The algorithm is presented in Figure 6.5.

With a set of samples from the posterior distribution P(z | w), statistics that are independent of the content of individual topics can be computed by integrating across the full set of samples. For any single sample we can estimate Φ and θ from the value z by:

$$\Phi'^{w}_{t} = \frac{(\beta + \mathrm{C}^{-\mathrm{n}}_{\mathrm{wt}})}{(\mathrm{C}^{-\mathrm{n}}_{\mathrm{t}} + \mathrm{W}\beta)} \tag{6.5}$$

$$\theta'^{d}_{t} = \frac{(\alpha + \mathrm{C}^{-\mathrm{n}}_{\mathrm{td}})}{(\mathrm{C}^{-\mathrm{n}}_{\mathrm{d}} + \mathrm{T}\alpha)} \tag{6.6}$$

These values correspond to the predictive distributions over new words w and new topics z conditioned on w and z. The algorithm for Gibbs sampling LDA is

shown in Figure 6.5. The dimensions required in this algorithm are shown in Tab. 6.4 and the details of the arrays required are shown in Tab. 6.5.

Table 6.4 Dimensions required in Gibbs Algorithm

Parameter	Description
D	Number of documents in corpus
W	Number of words in vocabulary
N	Total number of words in corpus
T	Number of topics
ITER	Number of iterations of Gibbs sampler

Table 6.5 Arrays used in Gibbs Algorithm

Array	Description
wid(N)	Word ID of n^{th} word
did(N)	Document ID of n^{th} word
z(N)	Topic assignment to n^{th} word
Cwt(W,T)	Count of word w in topic t
Ctd(T,D)	Count of topic t in document d
Ct(T)	Count of topic t

```
Input:  document-word index, vocabulary-word index, vocabulary, parameters value.
Output: topic wise word distribution
Procedure: as described below.
  //initialization of Markov chain initial state
 for all words of the corpus n Є [1, N] do
          sample topic index z (n) ~ Mult (1/T)
          // increment the count variables
          Cwt(wid(n),t) ++ , Ctd(t,did(n))++, Ct(t) ++ ;
 end for
 // run the chain over the burn-in period, and check for the
 // convergence. Generally for fixed number of iterations and then
 // take the samples at appropriate lag.
 for iteration i Є [1, ITER] do
          for all words of the corpus n Є [1, N] do
                   topic = z(n)
                   // decrement all the count variables, as not to
                   // include the current assignment
                   Cwt(wid(n),t) -- , Ctd(t,did(n))--, Ct(t) -- ;
                   for each topic t Є [1, T] do
                        P(t) = (Cwt(wid(n),t)+β)(Ctd(t,did(n))+α ) /
                              (Ct(t)+W β))
                   end for
                   sample topic t from P(t)
                   z(i) = t
                   // increment all the count variables to consider
                      //this new topic  assignment
                   Cwt(wid(n),t) ++ , Ctd(t,did(n))++, Ct(t) ++ ;
          end for
 end for
```

Figure 6.5 Gibbs Sampling Algorithm for LDA

6.7.5 Analysis of Gibbs Algorithm

The time and space complexity of the Gibbs sampling algorithm as shown in Figure 6.5 is:

Time Complexity ~ O (ITER * N* T)

Space Complexity ~ O (3 N + (D + W) T)

To understand the limitations of the existing algorithm, consider a million-document corpus with the following size parameters:

$D = 10^6$

$W = 10^4$

$N = 10^9$

For this corpus, it would be reasonable to run with T = 10^3 topics and ITER = 10^3 iterations. Using the space complexity equation as given above, the required memory would be,

(3* 10^9+ (10^6 +10^4) 10^3) = 4 Giga Bytes

This memory requirement is beyond most desktop computers and this makes Gibbs sampled topic model computation impractical for many purposes. As observed from the space complexity equation, the memory requirement increases because of N – the total number of words in a corpus which is getting multiplied three times.

To reduce this space complexity problem, I have proposed the Enhanced Gibbs sampling algorithm.

6.8 The Enhanced Gibbs sampling algorithm

To reduce the space requirement of the original Gibbs algorithm, I have applied the concept of partitioning the word set N and then executing the algorithm instead of loading the whole word set in a single run. With this we can achieve the reduction in space requirement as the size of N now reduces without any impact on the time complexity.

After each run on a partition the result is stored in separate variables and there is absolutely no need to merge the results of each partition. The variables are treated as global variables for all partitions.

Suppose we consider three partitions of the original word set N. The space complexity becomes:

Space Complexity ~ O (3 * N / P + (D + W) T)

Where P is total number of partitions,

~ O (3 * N / 3+ (D + W) T)

~ O (N + (D + W) T)

The space requirement reduces considerably. Meanwhile the time complexity becomes:

Time Complexity ~ O (ITER * N / P * T * P)

~ O (ITER * N* T)

The time complexity does not change since the algorithm is executed as many times as the number of partitions but for a smaller word set each time.

The proposed algorithm

The enhanced algorithm would require the following steps for execution:

- Read each document, perform tokenization, remove stop words, and apply case folding.
- Generate document-word matrix.
- Generate the vocabulary of the unique words in the collection.
- From the document-word matrix, generate the sparse arrays containing the vocabulary index and document index of each word.
- Apply the Enhanced Gibbs Sampling algorithm to extract the topic from the collection.
- Output the result.

The algorithm is as shown in Figure 6.6.

```
Input:  document-word index, vocabulary-word index, vocabulary, parameters
          value.
Output: topic wise word distribution

Procedure: as described below

 //initialization of Markov chain initial state

 for all partition p Є [1, P] do
          for all words of the current partition p,  n Є [1, N/P] do
          sample topic index z(n) ~ Mult(1/T)
          // increment the count variables
          Cwt(wid(n),t) ++ , Ctd(t,did(n))++, Ct(t) ++ ;

          end for
 end for

 // run the chain over the burn-in period, and check for the convergence.
// Generally for the fixed number of iteration and then take the samples at
  // appropriate lag.

 for all partition p Є [1, P] do
    for iteration i Є [1, ITER] do
        for all words of the current partition p, n Є [1, N/P] do
              topic = z (n)

             // decrement all the count variables, as not to include the
            // current assignment

            Cwt(wid(n),t) -- , Ctd(t,did(n))--, Ct(t) -- ;
          for each topic t Є [1, T] do
                    P(t) = (Cwt(wid(n),t) + β)(Ctd(t,did(n)) + α ) /
                          (Ct(t) + W β))
                end for

          sample topic t from P(t)
          z(i) = t

          // increment all the count variables to consider this new
             // topic assignment
          Cwt(wid(n),t) ++ , Ctd(t,did(n))++, Ct(t) ++ ;
        end for
   end for
end for
```

Figure 6.6 The Enhanced Gibbs Sampling Algorithm

6.8.1 Implementation of the Enhanced Gibbs Sampling Algorithm

This algorithm was implemented and tested on four datasets by varying the parameter values. It was implemented using MATLAB 7.0.1.

The Datasets

To extract the topics we require a text dataset that is rich in different topics. There are large number of textual datasets available which can be most suitable for this type of implementation such as news articles, emails, literature, research papers and abstracts, technical reports. The datasets that we used were:

1. The Cite Seer collection of scientific literature abstracts
2. The NIPS dataset of research papers
3. The Times Magazine articles
4. The Tehelka Magazine articles

The result after the preprocessing is completed on the four datasets is shown in Tab.6.6. This output is now used for the next step i.e. applying the Enhanced Gibbs sampling algorithm with different partitions.

Table 6.6 Output after pre-processing

Parameter Values	Cite Seer	NIPS	Times Magazine	Tehelka Magazine
No. of Total Words(N)	8320	51515	29601	17184
No. Unique Words(W)	683	1485	3820	1772
No. of Documents (D)	474	90	420	125
Time Taken in Seconds	120.656	661.532	410.359	244.86

6.8.2 Output and Comparison of the Enhanced Algorithm

This is the second phase i.e. applying both the Gibbs sampling and the Enhanced Gibbs sampling algorithms once the preprocessing is completed. A number of successive iterations are made through the topic assignment done by random sampling over the dataset. The proposed method does the same but instead of in a single step over the whole dataset, the dataset is divided into successive partitions and the algorithm is applied for each partition.

The output of both the algorithms with their comparisons is shown in Tab. 6.7 and Tab. 6.8. The algorithms were implemented on all the datasets with varying parameter values. I have displayed only two outputs related to the Cite Seer dataset in this section. Each dataset displayed similar results and

there was a considerable reduction in the space complexity when the Enhanced Gibbs sampling was used.

Table 6.7 Output and comparison of both algorithms

Name of Arrays required by the algorithm	Original Algorithm		Proposed Algorithm	
	No Partition		Partition = 2	
	Size	*Bytes*	*Size*	*Bytes*
ct (1,T)	1 x 30	240	1 x 30	240
ctd (T,D)	30x474	113760	30x474	113760
cwt (W,T)	683x30	163920	683x30	163920
did (1, N)	1x 8320	66560	1 x 4160	33280
wid (1, N)	1x 8320	66560	1 x 4160	33280
z (1, N)	1x 8320	66560	1 x 4160	33280
Total Bytes	477600		377760	
Time Taken (secs)	36.438		36.063	

In Tab. 6.7 the values for the parameters are: T = 30, ITER = 1000, α = 1.0 and β = .01 whereas in Tab. 6.8 the result with varying parameter values and partition values is displayed.

Table 6.8 Summary of comparison of both algorithms

Parameters	T = 30 ITER = 1000 α = 1 β = 0.01		T = 10 ITER = 1000 α = 0.05 β = 0.01	
	Time (sec)	*Space (Bytes)*	*Time (sec)*	*Space (Bytes)*
No Partition	36.438	477600	20.516	292320
Partition = 2	36.063	377760	20.344	192480
Partition = 3	35.734	344488	20.078	159208
Partition = 4	35.64	327840	20.094	142560

As can be seen from the observations shown, space complexity reduces significantly whereas the time complexity reduces marginally.

6.8.3 Conclusion and future enhancements of Topic Model

The topic model is a statistical language model that relates words and documents through topics. It is based on the idea that documents are made up of a mixture of topics, where topics are distributions over words. Gibbs sampling for implementing LDA has been a very popular model for topic models as compared to alternative methods such as variational Bayes and expectation propagation. Gibbs Sampling, a Markov Chain Monte Carlo method, is highly attractive because it is simple, fast and has very few adjustable parameters.

While the time and space complexity of the topic model scales linearly with the number of documents in a collection, computations are only practical for modest-sized collections of up to hundreds of thousands of documents. In this paper we have proposed an enhanced Gibbs sampled topic model algorithm which scales better than the original as the space complexity gets considerably reduced.

There are number of extensions possible with the topic models, such as author-topic models, author-role-topic models, topic models for images, hidden Markov topic models. Parallel topic models are also an emerging area of interest. The future work will be concentrating on any such extension of the topic model.

6.9 The Multi-Liaison Algorithm

6.9.1 Introduction of the proposed algorithm

The Multi-Liaison algorithm is useful for extracting multiple connections or links between subject and object from natural language input (English), which can have one or more than one subject, predicate and object. The parse tree visualization and the dependencies generated from the Stanford Parser are used to extract this information from the given sentence. Using the dependencies I have generated an output which displays which subject is related to which object and the connecting predicate. Finding the subjects and objects helps in determining the entities involved and the predicates determine the relationship that exists between the subject and the object. The

subjects can either be nouns or even pronouns. Moreover, one subject can be related to multiple objects and vice-versa.

I have named this algorithm 'The Multi-Liaison Algorithm' since the liaison between the subjects and objects would be displayed. The word 'liaison' has been used since the relationship and association between the subjects and predicates are displayed. This output would be useful for natural language processing (NLP), information retrieval, information extraction and also abstractive text summarization.

This algorithm has been published in the '**International Journal of Advanced Computer Science and Applications (IJACSA)'** by The Science and Information (SAI) Organization, ISSN: 2156-5570 (Online) & ISSN: 2158-107X (Print), Volume 2 Issue 5, 2011. It is available online at:
http://thesai.org/Publication/Archives/Volume2No5.aspx.

6.9.2 The Stanford Parser

The Stanford Parser is a probabilistic parser which uses the knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. This package is a Java implementation of probabilistic natural language parsers.

The Stanford dependencies provide a representation of grammatical relations between words in a sentence for any user who wants to extract textual relationships. The dependency obtained from Stanford parser can be mapped directly to graphical representation in which words in a sentence are nodes in graph and grammatical relationships are edge labels. This has been used to extract the relation between multiple subjects and objects when the sentence to be parsed is a little complicated. Stanford dependencies (SD) are triplets: name of the relation, governor and dependent.

6.9.3 The Parse Tree and Dependencies

The parse tree generated by the Stanford Parser is represented by three divisions: A sentence (S) having a noun phrase (NP), a verbal phrase (VP) and the full stop (.). The root of the tree is S.

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that can

easily be understood. The current representation contains approximately 52 grammatical relations. The dependencies are all binary relations. The definitions make use of the Penn Treebank part-of-speech (POS) tags and phrasal labels.

To find the multiple subjects in a sentence our algorithm searches the NP sub tree. The predicate is found in the VP sub tree and the objects are found in three different sub trees, all siblings of the VP sub tree containing the predicate. The sub trees are: PP (prepositional phrase), NP (noun phrase) and ADJP (adjective phrase).

6.9.4 The Multi-Liaison Algorithm details

To execute this algorithm, first we start with parsing a sentence by the Stanford parser and storing the result in some intermediate file so that it can be taken as input for this algorithm. The triplet extraction algorithm[3] has also been considered before finding the liaisons.

The application was written in JAVA using Net Beans IDE 6.5 RC2. It parsed a single sentence of 12 words in 8.35 seconds and displayed the output as shown in the examples below. This algorithm works equally well with simple as well as complex sentences and the output is very clear and precise.

As shown in Figure 6.7, the Multi-Liaison Algorithm takes as input the POS of each word, the parse tree and the typed dependencies [9]. Two functions are then called, the first is the GET_TRIPLETS and the second is the GET_RELATIONSHIP.

[3] Delia Rusu, Lorand Dali, Blaz Fortuna, Marko Grobelnik, Dunja Mladenic, “Triplet extraction from sentences” in Artificial Intelligence Laboratory, Jožef Stefan Institute, Slovenia, Nov. 7, 2008.

```
Function: CONVERT_ SENTENCE (Input_Str)
Returns: POS tagging, Parse tree, Typed Dependencies
Input_Str: Sentence to be parsed

[Run the Stanford parser with Input_Str as input]

Output_Str ← i)  POS of each word
             ii) The parse tree generated
             iii) The typed dependencies
Return Output_Str

Function: MULTI_LIAISON (Output_Str)
Returns: Multiple liaisons or error message
         Function GET_TRIPLETS (Output_Str)
         Function GET_RELATIONSHIP (Output_Str)
Display the multiple liaisons
```

Figure 6.7 The Multi-Liaison Algorithm

As shown in Figure 6.8, the GET_TRIPLETS function takes as input the Stanford Parse Tree and by considering the nodes under the NP sub tree and the VP sub tree, finds all the subjects, objects and predicates.

The GET_RELATIONSHIP finds and displays the relationships between the subjects and objects. The algorithm is displayed in Figure 6.9.

```
Function: GET_TRIPLET (Output_Str)
Returns: Multiple subjects, objects and predicates
[Read level 1 of Parse Tree – refer Figure 2]
If tree contains 'NP' or 'NNP' then
        Function GET_SUBJECT (NP sub tree)
Else
        Return error message

If tree contains 'VP' then
       Function GET_PREDICATE (VP sub tree)
       Function GET_OBJECT (VP sub tree)
Else
       Return error message

Function: GET_SUBJECT (NP sub tree)
Returns: Subject(s) and adjective(s)
For (all nodes of NP sub tree) do
    If NP sub tree contains 'NN' or 'NNP' or 'NNS' then
        Store POS as a subject
    If NP sub tree contains 'JJ' then
        Store POS as an adjective
Return the subject(s) and adjective(s)

Function: GET_PREDICATE (VP sub tree)
Returns: Predicate(s)
For (all nodes of VP sub tree) do
    If VP sub tree contains 'VB?' then
        Store POS as a predicate
    Else
        Return error message
Return the predicate(s)

Function: GET_OBJECT (VP sub tree)
Returns: Object(s)
For (all nodes of VP sub tree) do
    If VP sub tree contains 'NP' then
        For (all nodes of VP_NP sub tree) do
                If VP_NP sub tree contains 'NP' or 'NN' then
                    Store POS as an object
                Else
                    Return error message
    Else
        Return error message
Return the object(s)
```

Figure 6.8 The GET_TRIPLETS Function

```
Function: GET_RELATIONSHIP (Output_Str)
Returns: Multiple liaisons / relations
[Read the Stanford typed dependencies from Output_Str]
For (all terms in typed dependencies) do
     If typed dependencies contain 'NSUBJ' then
       Store both words of NSUBJ as S1 and S2
       For each value of subject from GET_SUBJECT do
          If subject matches S2 then
             [Check for predicates]
             For each value of predicate from
             GET_PREDICATE do
                 If predicate matches S1 then
                    [Concatenate subject and predicate as
                        R1]
                    Store R1 in the relation
   If typed dependencies contain 'DOBJ' or 'PREP' then
       Store both the words as D1 and D2
       For each value of object in GET_OBJECT do
          If object matches D2 then
             Store value of object in the relation as R2
Return R1+R2
```

Figure 6.9 The GET_RELATIONSHIP Function

6.9.5 Output of the Multi-Liaison Algorithm

As per the algorithm discussed above, the output is shown below. In the first example, the outputs of the Stanford parse as well as the output of the Multi-Liaison both are displayed including the parse tree. In subsequent examples the parse tree is not displayed but the tagging, dependencies and the Multi-Liaison output is displayed. Figure 6.10 displays the parse tree.

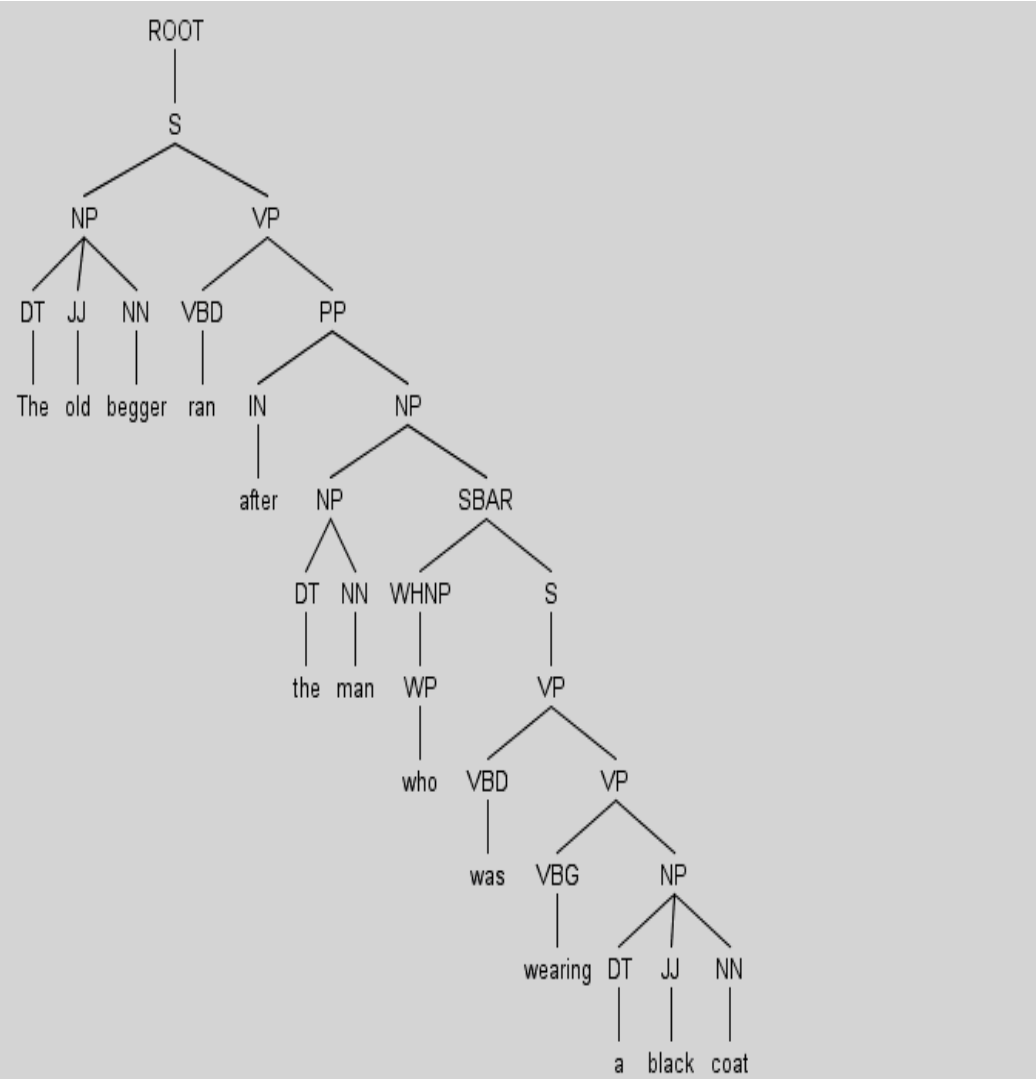

Figure 6.10 The Stanford Parse Tree

Example 1: The old beggar ran after the rich man who was wearing a black coat
The Stanford Parser output:
Tagging:
The/DT old/JJ beggar/NN ran/VBD after/IN the/DT rich/JJ man/NN who/WP was/VBD wearing/VBG a/DT black/JJ coat/NN

Parse Tree:

```
(ROOT
  (S
    (NP (DT The) (JJ old) (NN beggar))
    (VP (VBD ran)
      (PP (IN after)
        (NP
          (NP (DT the) (JJ rich) (NN man))
          (SBAR
            (WHNP (WP who))
            (S
              (VP (VBD was)
                (VP (VBG wearing)
                  (NP (DT a) (JJ black) (NN coat)))))))))))
```

Typed Dependencies:

```
det(beggar-3, The-1)
amod(beggar-3, old-2)
nsubj(ran-4, beggar-3)
det(man-8, the-6)
amod(man-8, rich-7)
prep_after(ran-4, man-8)
nsubj(wearing-11, man-8)
aux(wearing-11, was-10)
rcmod(man-8, wearing-11)
det(coat-14, a-12)
amod(coat-14, black-13)
dobj(wearing-11, coat-14)
```

The Multi-Liaison Output:

```
Subject: 1
NN beggar
Predicate: 3
VBD ran
VBD was
VBG wearing
Object: 2
NN man   JJ rich
NN coat  JJ black
```

Relationship:

```
beggar - ran  - man
man - wearing  - coat
```

Figure 6.11 Example 1

As shown above, the Multi-Liaison Algorithm displays the relationship between the subject and object (beggar and man) as well as the relationship between the two objects (man and coat).

Example 2: The dog and the cat ran after the mouse and the mongoose

Tagging:
The/DT dog/NN and/CC the/DT cat/NN ran/VBD after/IN the/DT mouse/NN and/CC the/DT mongoose/NN

Typed Dependencies:
det(dog-2, The-1)
nsubj(ran-6, dog-2)
det(cat-5, the-4)
conj_and(dog-2, cat-5)
nsubj(ran-6, cat-5)
det(mouse-9, the-8)
prep_after(ran-6, mouse-9)
det(mongoose-12, the-11)
prep_after(ran-6, mongoose-12)
conj_and(mouse-9, mongoose-12)

The Multi-Liaison Output:
Subject: 2
NN dog
NN cat
Predicate: 1
VBD ran
Object: 2
NN mouse
NN mongoose

Relationship:
dog - ran – mouse - mongoose
cat - ran - mouse - mongoose

Figure 6.12 Example 2

Example 3: Jack and I visited the zoo with our children

I have also considered pronoun as a subject and therefore have got the relationship with 2 subjects in terms of noun and pronoun.

Tagging:
Jack/NNP and/CC I/PRP visited/VBD the/DT zoo/NN with/IN our/PRP$ children/NNS

Typed Dependencies:
nsubj(visited-4, Jack-1)
conj_and(Jack-1, I-3)
nsubj(visited-4, I-3)
det(zoo-6, the-5)
dobj(visited-4, zoo-6)
poss(children-9, our-8)
prep_with(visited-4, children-9)

The Multi-Liaison Output:
Subject: 2
NNP Jack
PRP I
Predicate: 1
VBD visited
Object: 2
NN zoo
NNS children
PRP$ our

Relationship:
Jack - visited - zoo - children
I - visited - zoo - children

Figure 6.13 Example 3

All the three examples shown in the figures above have different number of subjects and objects and the relationship between them is also not similar. The Multi-Liaison Algorithm output in this way can be very useful for Text Mining applications where a variety of sentences are to be mined.

6.9.6 Conclusion and future enhancements

The proposed algorithm which displays the relationships between subjects and objects in sentences where there are multiple subjects and objects. The Stanford parser output was used to generate this result.

This algorithm would be usable not only by Text Mining experts and computational linguists but also by the computer science community more generally and by all sorts of professionals like biologists, medical researchers, political scientists, business and market analysts, etc. In fact it would be easy for users not necessarily versed in linguistics to see how to use and to get value from the simple relationship that is displayed so effectively.

Chapter 7: Future Enhancements

The enhancements in Text Mining have already been discussed in the related chapters. However one article[1] interested me as it is very much related to what exactly Text Mining is supposed to do.

During a series of hearings, the U.S. Senate Select Committee on Intelligence showed that prior to September 11, 2001, the American intelligence community had collected a significant amount of data about the men who attacked the World Trade Center and the Pentagon. The various intelligence agencies were simply unable to connect the dots. In his report, Richard C. Shelby, then vice chairman of the committee, stressed that agencies need powerful new tools to analyze the huge volumes of information they bring in.

Text-mining software is one of the front-line tools that the government is now using to tease out valuable connections. These specialized search engines can quickly sift through mountains of unstructured text—anything that's not carefully arranged in a database or spreadsheet—and pull out the meaningful stuff. They can infer relationships within data that are not stated explicitly. It is something we do all the time automatically but is enormously complicated for computers. "We bridge the gap between information and action," says Barak Pridor, CEO of ClearForest, a text-mining company.

The result of years of research at facilities such as Bell Labs and the Palo Alto Research Center, Text Mining applications have long been used in business. But more government agencies, including the Defense Intelligence Agency, the Department of Homeland Security, and the FBI, are using them to evaluate the multitude of e-mail messages, phone call transcripts, memos, foreign news stories, and other pieces of intelligence data these agencies collect each day.

Software from companies such as Autonomy, ClearForest, and Inxight Software can locate words and phrases the same way an ordinary search engine does. But that's just the beginning. Such applications are clever enough to run conceptual searches, locating, say, all the phone numbers and

[1] http://www.pcmag.com/article2/, Cade Metz, 'Uncovering telltale patterns'

place names buried in a collection of intelligence communiqués. More impressive, the software can identify relationships, patterns, and trends involving words, phrases, numbers, and other data.

Using statistical and mathematical analysis, the programs can sift through thousands of documents and determine how certain words relate to each other. If a news story says that "Zacarias Moussaoui was a follower of the Islamic cleric Abu Qatada while living in London," a Text Mining application can identify Moussaoui and Qatada as people, identify London as a place, and determine the relationship among the three.

In theory, a human analyst could pick up those connections easily, but manually sifting through the enormous volumes of information is often impractical. Fortunately, Text Mining applications can deal with these and other similar functions.

Summary

The enormous amount of information stored in unstructured texts cannot simply be used for further processing by computer, which typically handles text as simple sequences of character strings. Therefore, specific processing methods and algorithms are required in order to extract useful patterns. Text Mining refers generally to the process of extracting interesting information and knowledge from unstructured text. Text Mining represents a significant step forward from text retrieval. It is a relatively new and vibrant research area that is changing the emphasis in text-based information technologies from low level 'retrieval' and 'extraction' to higher level 'analysis' and 'exploration' capabilities. Given the large amount of data available today in the form of text, tools that automatically find interesting relationships, hypothesis or ideas, or assist the user in finding these would be extremely useful and current research area.

In this thesis, the work on Text Mining has been divided in three main sections – Text Clustering, Text Classification and Text Summarization. The Text Pre-processing and Text Transformation which are the preliminary steps before actually a Text Mining algorithm can be implemented have been discussed first. A comparative between the different stemming algorithms has been discussed in detail.

In subsequent sections the three broad applications mentioned above and their related algorithms and methods which are currently popular have been discussed.

Text Clustering is the unsupervised method of Text Mining of gathering or dividing related documents in such a way that documents within cluster are similar to each other whereas the documents across clusters are different. The algorithms discussed are the K-Means, the DBSCAN and the SNN. A new algorithm 'SNNAE' has been proposed and its implementation details and comparative between the others is also given.

Text Classification or Text Categorization is the supervised method of Text Mining where we have a training class of documents which have pre-defined

classes associated with them and using these training documents, classifiers are modeled or learned so that they can be applied on the new documents which have not been pre-classified i.e. the testing class of documents. In this section the Naïve Bayes (Two variants), kNN, decision trees and support vector machines are discussed. Their comparatives are also given. A new method based on kNN – 'The Novel kNN' has been designed and implemented.

Text Summarization deals with producing a synopsis / summary of a single document or set of documents. It deals with abstractive and extractive methods. In this section apart from discussing what is already available, two innovative algorithms have been designed. One is related to the topic model – a probabilistic model that automatically learns the topics contained in a set of documents. The method developed is based on the Latent Dirichlet Allocation which is used to find the latent semantic structure that is topics in the case of text collections. The method applied is the Gibbs sampling – a kind of Markov Chain Monte Carlo Method. The proposed algorithm is the 'Enhanced Gibbs Sampling'. Another algorithm related to part-of-speech (POS) tagger - 'The Multi-Liaison Algorithm' has been designed which can be useful in text summarization when the semantics are also to be studied. This can be part of natural language processing as well as Text Mining application.

All the related algorithms mentioned above have been published in either International Journals or International Conference Proceedings.

With today's need to handle and process collections that are orders of magnitude much larger, scalable and parallel Text Mining methods are required. A lot of work has been going and in this field and a lot more remains to be done.

Appendix A

The stop words list is available on the site of the Onix Text Retrievel Tookit and the site is: http://www.lextek.com/manuals/onix/stopwords1.html. This stopword list is probably the most widely used stopword list. It covers a wide number of stopwords without getting too aggressive and including too many words which a user might search upon. This wordlist contains 429 words.

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop - is another site where stopwords are available.

a	about	above	across	after	again	against	all
almost	alone	along	already	also	although	always	among
an	and	another	any	anybody	anyone	anything	anywhere
are	area	areas	around	as	ask	asked	asking
asks	at	away					
back	backed	backing	backs	be	became	because	become
becomes	been	before	began	behind	being	beings	best
better	between	big	both	but	by		
came	can	cannot	case	cases	certain	certainly	clear
clearly	come	could					
did	differ	different	differently	do	does	done	down
down	downed	downing	downs	during			
each	early	either	end	ended	ending	ends	enough
even	evenly	ever	every	everybody	everyone	everything	everywhere
face	faces	fact	facts	far	felt	few	find
finds	first	for	four	from	full	fully	further
furthered	furthering	furthers					
gave	general	generally	get	gets	give	given	gives
go	going	good	goods	got	great	greater	greatest
group	grouped	grouping	groups				
had	has	have	having	he	her	here	herself
high	high	high	higher	highest	him	himself	his
how	however						
i	if	important	in	interest	interested	interesting	interests
into	is	it	its	itself			
just							
keep	keeps	kind	knew	know	known	knows	
large	largely	last	later	latest	least	less	let
lets	like	likely	long	longer	longest		
made	make	making	man	many	may	me	member
members	men	might	more	most	mostly	mr	mrs
much	must	my	myself				

necessary	need	needed	needing	needs	never	new	new
newer	newest	next	no	nobody	non	noone	not
nothing	now	nowhere	number	numbers			
of	off	often	old	older	oldest	on	once
one	only	open	opened	opening	opens	or	order
ordered	ordering	orders	other	others	our	out	over
part	parted	parting	parts	per	perhaps	place	places
point	pointed	pointing	points	possible	present	presented	presenting
presents	problem	problems	put	puts			
quite							
rather	really	right	right	room	rooms		
said	same	saw	say	says	second	seconds	see
seem	seemed	seeming	seems	sees	several	shall	she
should	show	showed	showing	shows	side	sides	since
small	smaller	smallest	so	some	somebody	someone	something
somewhere	state	states	still	still	such	sure	
take	taken	than	that	the	their	them	then
there	therefore	these	they	thing	things	think	thinks
this	those	though	thought	thoughts	three	through	thus
to	today	together	too	took	toward	turn	turned
turning	turns	two					
under	until	up	upon	us	use	used	uses
very							
want	wanted	wanting	wants	was	way	ways	we
well	wells	went	were	what	when	where	whether
which	while	who	whole	whose	why	will	with
within	without	work	worked	working	works	would	
year	years	yet	you	young	younger	youngest	your
yours							

Appendix B

Some popular Text Mining tools available:

Product Name	Pre-processing	Clustering	Categorizing	Summarizing	API
Commercial					
Clearforest	√	√		√	
Copernic Summarizer	√			√	
dt Search	√			√	
Insightful Infact	√	√	√	√	√
Inxight	√	√	√	√	√
SPSS Clementine	√	√	√	√	
SAS Text Miner	√	√	√	√	
TEMIS	√	√	√	√	
WordStat	√	√	√	√	
Open Source					
GATE	√	√	√	√	√
RapidMiner	√	√	√	√	√
Weka / WEA	√	√	√	√	√
R / tm	√	√	√	√	√

Publications

International Level:

Sr. No.	Name of Journal/Organization	Year of Publication	Title of Paper
1	International Journal of Computer Technology and Applications (IJCTA) - Volume 2 Issue 6/ November - December 2011/ pg. 1930-1938, ISSN:2229-6093. http://ijcta.com/vol2issue6.php	2011	A Comparative Study of Stemming Algorithms
2	International Journal of Computer Information Systems by Silicon Valley Publishers (UK), ISSN: 2229-5208, October 2011 issue and is available at http://www.svpublishers.co.uk/#/ijcis-oct-2011/4557969965.	2011	The Enhanced Gibbs Sampling for Topic Model
3	International Journal of Advanced Computer Science and Applications (ISSN: 2156-5570) Vol. 2, No. 5 (2011), p. 130--134. http://thesai.org/Publication/Archives/Volume2No5.aspx	2011	The Multi-Liaison Algorithm
4	IEEE Computer Society & World Research Organization (CSIE09) ISBN 978-0-7695-3507-4/08, DOI 10.1109/CSIE2009.997, Pg. 436 BMS Number CFP0960F-CDR http://www.computer.org/portal/web/csdl/doi/10.1109/CSIE.2009.997	2009	The Shared Nearest Neighbor Algorithm with Enclosures (SNNAE)
5	Macmillan Publisher, Institute Of Mangement Technology, Ghaziabad. ISBN 0230-63469-9 Pg. 221	2008	Discovering Communication Threads In Emails Using A Conceptual Clustering Approach
6	INCRUIS 2006 – International Conference on Resource Utilization, Kongu Engineering College, Tamil Nadu. ISBN 81-7764-940-x, Pg. 916	2006	Fuzzy Clustering for The Student Resource Utilization

National Level:

Sr. No.	Name of Journal/Organization	Year of Publication	Title of Paper
1	SPCTS, DAIICT & IEEE	2007	An Approach Towards The SNN Clustering Algorithm **(Awarded 2nd Prize)**
2	Tele Tech 2005 National Seminar on Applied Computing Tech., IETE, Rajkot.	2005	Ontology Mining for Virtual Reality
3	Business Information Management Conference, IMT Ghaziabad.	2005	Data Quality - A Stepping Stone in Business Intelligence
4	Business Information Management Conference,IMT Ghaziabad	2005	Text Data Mining for Knowledge Discovery in Business Intelligence
5	National Conference on Information & Communication Technology - 2005, Technology Today, Ahmedabad.	2005	Knowledge Discovery - Using Grids
6	National Conference on Information & Communication Technology - 2005, Technolgy Today, Ahmedabad.	2005	Ontology Clustering -An Insight
7	Gyanodaya : Next Generation IT, Gyanganga Institute , Jabalpur.	2005.	Datamining - The Metadata
8	Embedded Systems And Emerging Trends, IETE, Vadodara.	2005	An Approach Towards Embedded Databases
9	National Level Technical Paper Presentation Competion, S.V.Institute, Kadi & Amoghsiddhi Edu. Society, Sangli	2005	A Study of Contemporary Databases
10	National Level Technical Paper Presentation Competition , S.V.Institute, Kadi & Amoghsiddhi Edu. Society, Sangli	2005	An Insight to Data Mining and Data Warehousing **(Awarded First Prize)**

Bibliography

[1] Hearst, M. Untangling Text Data Mining .In the Proceedings of ACL'99:the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June 20-26, 1999.

[2] Keno Buss Literature Review on Preprocessing for Text Mining. Software Technology Research Laboratory, http://www.cse.dmu.ac.uk/STRL/.

[3] G. Salton , A. Wong , C. S. Yang, A vector space model for automatic indexing (http:/ / doi. acm. org/ 10. 1145/ 361219. 361220), Communications of the ACM, v.18 n.11, p.613-620, Nov. 1975.

[4] Eiman Tamah Al-Shammari, "Towards An Error-Free Stemming", in Proceedings of ADIS European Conference Data Mining 2008, pp. 160-163.

[5] Frakes W.B. "Term conflation for information retrieval". Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval. 1984, 383-389.

[6] Frakes William B. "Strength and similarity of affix removal stemming algorithms". ACM SIGIR Forum, Volume 37, No. 1. 2003, 26-30.

[7] Funchun Peng, Nawaaz Ahmed, Xin Li and Yumao Lu. "Context sensitive stemming for web search". Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007, 639-646.

[8] Galvez Carmen and Moya-Aneg•n F˜lix. "An evaluation of conflation accuracy using finite-state transducers". Journal of Documentation 62(3). 2006, 328-349

[9] J. B. Lovins, "Development of a stemming algorithm," Mechanical Translation and Computer Linguistic., vol.11, no.1/2, pp. 22-31, 1968.

[10] Harman Donna. "How effective is suffixing?" Journal of the American Society for Information Science. 1991; 42, 7-15 7.

[11] Hull David A. and Grefenstette Gregory. "A detailed analysis of English stemming algorithms". Rank Xerox Research Center Technical Report. 1996.

[12] Kraaij Wessel and Pohlmann Renee. "Viewing stemming as recall enhancement". Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. 1996, 40-48.

[13] Krovetz Robert. "Viewing morphology as an inference process". Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. 1993, 191-202.

[14] Mayfield James and McNamee Paul. "Single N-gram stemming". Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. 2003, 415-416.

[15] Melucci Massimo and Orio Nicola. "A novel method for stemmer generation based on hidden Markov models". Proceedings of the twelfth international conference on Information and knowledge management. 2003, 131-138.

[16] Mladenic Dunja. "Automatic word lemmatization". Proceedings B of the 5th International Multi-Conference Information Society IS. 2002, 153-159.

[17] Paice Chris D. "Another stemmer". ACM SIGIR Forum, Volume 24, No. 3. 1990, 56-61.

[18] Paice Chris D. "An evaluation method for stemming algorithms". Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. 1994, 42-50.

[19] Plisson Joel, Lavrac Nada and Mladenic Dunja. "A rule based approach to word lemmatization". Proceedings C of the 7th International Multi-Conference Information Society IS. 2004.

[20] Porter M.F. "An algorithm for suffix stripping". Program. 1980; 14, 130-137.

[21] Porter M.F. "Snowball: A language for stemming algorithms". 2001.

[22] R. Sun, C.-H. Ong, and T.-S. Chua. "Mining Dependency Relations for Query Expansion in Passage Retrieval". In SIGIR, 2006

[23] Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra and Kalyan Kumar Datta. "YASS: Yet another suffix stripper". ACM Transactions on Information Systems. Volume 25, Issue 4. 2007, Article No. 18.

[24] Toman Michal, Tesar Roman and Jezek Karel. “Influence of word normalization on text classification”. The 1st International Conference on Multidisciplinary Information Sciences & Technologies. 2006, 354-358.
[25] Xu Jinxi and Croft Bruce W. “Corpus-based stemming using co-occurrence of word variants”. ACM Transactions on Information Systems. Volume 16, Issue 1. 1998, 61-81.
[26] Hull D. A. and Grefenstette,“ A detailed analysis of English Stemming Algorithms”, XEROX Technical Report, http://www.xrce.xerox.
[27] Rui Xu, Student Member, IEEE and Donald Wunsch II, Fellow, IEEE “Survey of Clustering Algorithm”, IEEE Transactions on Neural Networks, Vol. 16, No.3, May (2005).
[28] Martin Ester. Hans-Peter Kriegel, Jorg Sandar, Xiaowei Xu,” A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” KDD 96, Portland, OR, pp. 226-231 (1996).
[29] Hetal Bhavsar and Anjali Jivani, “An approach towards the Shared Nearest Neighbor (SNN) Clustering Algorithm”,in the National Conference SPCTS ’07 at DAII-CT, Gandhinagar, India, September (2007).
[30] R. A. Jarvis and E. A. Patrick. “Clustering using a Similarity Measure Based on Shared Nearest Neighbors,” IEEE Transaction on Computers, Vol. C-22, No.11,November (1973).
[31] Levent Ertoz, Michel Steinbach and Vipin Kumar, “Finding clusters of different sizes, shapes,and densities in noisy, high dimensional data”, accepted for SIAM International Conference on Data Mining (2003).
[32] A.M.Fahim, A.M Salem, F.A. Torkey and M.A. Ramadan “Density Clustering Algorithm Based on Radius of Data (DCBRD)”, Georgian Electronic Scientific Journal:Computer Science and Telecommunications, vol. 11, No.4, 2006.
[33] Levent Ertöz, Michael Steinbach, Vipin Kumar, “ A New SNN clustering algorithm and its applications”, Workshop on Clustering High Dimensional Data and its Applications,” Second SIAM International Conference on Data Mining, Arlington, VA, (2002)
[34] A, McCallum, K. Nigam, L. Ungar, “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching,” KDD 2000, pp. 169-178 (2000).
[35] Zhiwei SUN, Zheng ZHAO, Hongmei WANG, Maode MA, Liangang and Yantai SHU, “A Fast Clustering Algorithm Based on Grid and Density”, in IEEE, CCECE/CCEGI, Saskatoon, May 2005.
[36] YasserEl-Sonbaty, M.A. Ismail, and Mohamed Farouk, “An Efficient Density Based Clustering Algorithm for Large Databases”, In Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, 2004.
[37] Susan Dumais et al., “Inductive Learning Algorithms and Representations for Text Categorization “,Proceedings of the seventh international conference on Information and knowledge management, ISBN:1-58113-061-9 doi>10.1145/288627.288651
[38] Joachims T, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Machine Learning, 1998. 11398:137-142.
[39] Yang Yi ming, I.iu Xin. A Re-Examination of Text Categorization Methods. Proceedings of ACM SIGIR Conference on Research and Development of Information Retrieval, Berkeley,California, 1999.42.
[40] Weiss S M, Damerau C F J. Maximizing Text Mining Performance. IEEE Intelligent Systems. New York: IEEE Press, 1999.12l.
[41] C. Apte, F. Damerau, and S. Weiss. Towards Language Independent Automated Learning of Text Categorization Methods.online on google scholar.
[42] Hofmann, Thomas (1999). "Probabilistic Latent Semantic Indexing". Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval. http://www.cs.brown.edu/~th/papers/Hofmann-SIGIR99.pdf.
[43] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I; Lafferty, John (January 2003). "Latent Dirichlet allocation". Journal of Machine Learning Research **3**: pp. 993–1022. doi:10.1162/jmlr.2003.3.4-5.993. http://jmlr.csail.mit.edu/papers/v3/blei03a.html.
[44] David Newman, Padhraic Smyth, Mark Steyvers. “Scalable Parallel Topic Models”. Journal of Intelligence Community Research and Development (2006).
[45] Griffiths, T.L., and Steyvers, M., “Finding Scientific Topics”, National Academy of Sciences, 101 (suppl. 1) 5228–5235, 2004.
[46] Griffiths, T. L., & Steyvers, M., “A probabilistic approach to semantic representation”, In Proceedings of the 24th Annual Conference of the Cognitive Science Society, 2002.

[47] Griffiths, T., “Gibbs sampling in the generative model of Latent Dirichlet Allocation”, Technical report, Stanford University (2002).
[48] D. Newman, C. Chemudugunta, P. Smyth, M. Steyvers, “Analyzing Entities and Topics in News Articles using Statistical Topic Models”, LNCS 3975, Intelligence and Security Informatics. Springer. (2006).
[49] R. M. Neal (1993) “Probabilistic Inference Using Markov Chain Monte Carlo Methods”, http://www.cs.utoronto.ca/_radford/review.abstract.html.
[50] G. Heinrich, “Parameter estimation for text analysis”, Technical Report, 2004.
[51] D. Newman and S. Block, “Probabilistic topic decomposition of an eighteenth-century American newspaper”, J. Am. Soc. Inf. Sci. Techno., 57(6):753--767, 2006.
[52] Steyvers, M. & Griffiths, T., “Probabilistic topic models”, In T. Landauer, D. S. McNamara, S. Dennis, & W. Kintsch (Eds.), Handbook of Latent Semantic Analysis. Hillsdale, NJ: Erlbaum, 2007.
[53] Griffiths, T. L., & Yuille, A., “A primer on probabilistic inference”, to appear in M. Oaks ford and N. Chater (Eds.). The probabilistic mind: Prospects for rational models of cognition. Oxford: Oxford University Press.
[54] D. Heckerman, “A Tutorial on Learning with Bayesian Networks”, In Learning in Graphical Models, M. Jordan, Ed... MIT Press, Cambridge, MA, 1999.
[55] H. Guo and W.H. Hsu, “A survey of algorithms for real-time Bayesian network inference”, AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, 1-12, Edmonton, Alberta, 2002.
[56] Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M., “Markov Chain Monte Carlo in Practice: A Roundtable Discussion'', The American Statistician, Vol. 52, pp. 93-100, 1998.
[57] M. Girolami and A. Kaban, “On an equivalence between PLSI and LDA”, In Proceedings of SIGIR 2003. http://citeseer.ist.psu.edu/girolami03equivalence.html.
[58] Kevin P. Murphy, “An introduction to graphical models”, University of Columbia, 2001.
[59] D. Klein, C. D. Manning, “Fast exact inference with a factored model for natural language parsing” in Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10, 2003.
[60] D. Klein, C. D. Manning, “Accurate unlexicalized parsing” in Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430, 2003.
[61] Delia Rusu, Lorand Dali, Blaz Fortuna, Marko Grobelnik, Dunja Mladenic, “Triplet extraction from sentences” in Artificial Intelligence Laboratory, Jožef Stefan Institute, Slovenia, Nov. 7, 2008.
[62] D. Lin, P. Pantel, “DIRT - Discovery of inference rules from text” in Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001. pp. 323-328, 2001.
[63] J. Leskovec, M. Grobelnik, N. Milic-Frayling, “Learning sub-structures of document semantic graphs for document summarization” in Proceedings of the 7th International Multi-Conference Information Society IS 2004, Volume B. pp. 18-25, 2004.
[64] J. Leskovec, N. Milic-Frayling, M. Grobelnik, “Impact of linguistic analysis on the semantic graph coverage and learning of document extracts” in National Conference on Artificial Intelligence, 2005.
[65] O. Etzioni, M. Cafarella, D. Downey, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A.Yates. Unsupervised named-entity extraction from the Web: An experimental study. Artificial Intelligence, Volume 165, Issue 1, June 2005, Pages 91-134.
[66] Marie-Catherine de Marneffe, Bill MacCartney, Christopher D. Manning, “Generating typed dependency parses from phrase structure parses” in LREC 2006.
[67] Marie-Catherine de Marneffe, Christopher D. Manning, “The Stanford typed dependencies representation” in COLING Workshop on Cross-framework and Cross-domain Parser Evaluation, 2008.
[68] Marie-Catherine de Marneffe, Christopher D. Manning, “The Stanford typed dependencies manual” in Revised for Stanford Parser v1.6.2, February, 2010.
[69] Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, Christopher D. Manning, “Parsing to Stanford dependencies: Trade-offs between speed and accuracy” in 7th International Conference on Language Resources and Evaluation (LREC 2010).
[70] Dick Grune and Ceriel Jacobs, “Parsing Techniques – A Practical Guide,” in Proceedings of the 8th International Conference, CICLing 2007, Mexico City, A. Gelbukh (Ed), pp. 311-324, Springer, Germany, 2007.

Books:

[1] Jiawei Han and Micheline Kamber, 'Data Mining Concepts and Techniques', Elsevier (2001).
[2] Christopher Manning et al.,'An Introduction to Information Retrieval', Cambridge UP, 2009.
[3] Manu Kochady 'Text Mining Application Programming' , Thomson India Edition, 2006.
[4] C.S.R. Prabhu, "Data Warehousing – Concepts, Techniques, Products and Applications", PHI
[5] Margaret H. Dunham, "Data Mining – Introductory and Advanced Topics, Pearson Edu.
[6] IBM, "An Introduction to Building the Data Warehouse", PHI
[7] Alex Berson, Stephen J. Smith, "Data Warehousing, Data Mining & OLAP", Tata McGraw Hill
[8] David Hand, Heikki Mannila, Padhraic Smyth, "Principles of Data Mining", PHI
[9] Daniel T. Larose, "Data Mining Methods and Models", Wiley-Interscience.
[10] Bart Kosko, 1995, " Neural Networks And Fuzzy Systems", PHI, New Delhi.
[11] Ronen Feldman, James Sanger, "The Text Mining Handbook", Cambridge University Press, 2006
[12] Ritu Arora, "Text Mining: Classification and Clustering", University of Alabama at Birmingham
[13] Thomas Miller, "Data and Text Mining", Pearson Education, 2008.
[14] Daniel Jurafsky & James Martin, "Speech & Language Processing ", Pearson Education
[15] Steven Bird, Ewan Klein & Edward Loper, "Natural Language Processing with Python", O'Reilly