**Republic of Iraq**
**Ministry of Higher Education**
**and Scientific Research**
**Al-Nahrain University**
**College of Science**

# *Content Based Text Mining*

***A thesis***
***Submitted to college of Science of Al-Nahrain University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science***

***By***
***Zahraa Raji Mohi Al-Zobaidy***
***(B.Sc. 2005)***

***Supervisor***
***Dr.Taha S. Bashaga***

*January 2009* *Muharam 1430*

بِسْمِ اللهِ الرَّحْمـــــــنِ الرَّحِيمِ

(وَيَسْأَلونَكَ عَنِ الرُّوحِ قُلِ الرُّوحُ مِنْ أَمْرِ رَبِّي وَمَا أُوتِيتُـمْ مِنَ العِلـمِ إِلاَّ قَلِيلاً)

صَدَقَ اللهُ العَظِيمْ

سورة الاسراء الاية (٨٥)

# *Supervisor Certification*

I certify that this thesis was prepared under my supervision at the Department of Computer Science/ College of Science/ Al-Nahrain University, by **Zahraa Raji Mohi Al-Zobaidy** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science**.**

Signature:

Name : **Dr.Taha S.Bashaga**

Title : **Lecture**

Date : **/ /2008**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name : **Dr. Taha S.Bashaga**

Title : **Head of the Department of Computer Science,**
**Al-Nahrain University.**

Date : **/ /2008**

# *Certification of the Examination Committee*

We chairman and members of the examination committees certify that we have studied the thesis entitled "Content **Based Text Mining**" presented by the student **Zahraa Raji Mohi Al-Zobaidy** and examined its content and in what is related to it, and we have found it worthy to be accepted for the degree of Master of Science in computer Science.

Signature:
Name : **Dr. Abdulmonem S. Rahema**
Title : **Assist. Prof.**
Date : **/ /2009**

Signature:
Name : **Dr. Haitham A. Omar**
Title : **Lecture**
Date : **/ /2009**

Signature:
Name : **Dr. Baràa A. Attea**
Title : **Assist. Prof.**
Date : **/ /2009**

Signature:
Name : **Dr. Taha S. Bashaga**
Title : **Lecture**
Date : **/ /2009**
**(Supervisor)**

Approved by the Dean of the College of Science, Al-Nahrain University
Signature:
Name : **Dr. Laith Abdul Aziz Al-Ani**
Title : **Assist. Prof.**
Date : **/ /2009**
**(Dean of college of science)**

# Dedication

**To:**

*The cause of my success that stands beside me with her endless love and support my mother…*

*Every person who supports me in my family…*

*Every friend and person who helped and guides me…*

**Zahraa**

# Acknowledgement

First of all, praise is to **GOD** Who has enlightened me and paved the way to accomplish this thesis.

Then, I would like to express deepest gratitude and appreciation to my supervisor **Dr. Taha S. Bashaga** for his excellent advice, guidance, and cooperation during the course of this work, he has followed my work step by step, So i thank him from the depth of my heart.

My sincere thanks go to members of stuff of Computer Science Department, for their help.

I would like to thank my family for their encouragement during my study.

**Zahraa**

# Abstract

Text mining has become an important research issue since the explosion of digital and online text information, where data might be stored as electronic documents or as text fields in databases, text mining has increased in importance and economic value. One of the important goals in text mining is automatic classification of electronic documents. Computer programs scan text in a document and apply a model that assigns the document to one or more of predefined classes (topics).

The document is manipulated first by performing some natural language processing methods on them to remove unnecessary data from the document which does not convey any useful information, then removing any confusion through putting the words with the same meaning in single word to make the comparison between the documents more easily.

Then, the second step used in this thesis is document classification. In this step the set of documents is divided into training and testing document subsets randomly, both training and testing subset are used for machine learning. A vector space model for Term Frequency Inverse Document Frequency (TFIDF) method together with similarity factor are used, in classifying both the training and testing documents. Finally, the performance of the used classification method will be measured.

Reuters-21578 test collection was used to measure the performance of the proposed classification method in this thesis. The results obtained are encouraging.

# *Table of Contents*

# Table of Contents

**Chapter 1: Principles of Text Mining**

1.1 What is Text Mining? ..…………………….……………....1

1.2 Technologies of Text Mining …………………..…………....2

1.3 Literature Survey ………………………….……………..5

1.4 Aim of Thesis ……………………………….…………......7

1.5 Thesis Layout ……………………………..……..................7

**Chapter 2: Text Mining (Analysis & Technique)**

2.1 Introduction ……………………………………………......8

2.2 Text Mining Application ….……………………...............….9

2.3 Text Categorization or Classification (TC) Technique ..........11

2.3.1 Naïve Bayes ………..................................................14

2.3.2 k-Nearest Neighbor ……….........................................15

2.3.3 Decision Tree ……….................................................16

2.3.4 Support Vector Machine ……………………….........17

2.3.5 TFIDF(Term Frequency-Inverse Document Frequency)..18

2.4 Evaluation Performance of Text Categorization ....................21

2.5 Natural Language Processing in Text Mining .......................23

2.6 Word Stemming ……………………………………….....24

2.6.1 Porter Stemmer ………………...…………...….25

2.6.2 Piace/Husk Stemmer ……………………....................26

2.6.3 Lovins Stemmer …………………………….…….…26

2.6.4 Dawson's Stemmer ……...……………………………...27

2.7 Information Extraction ……………………..……...….........27

2.8 Data Sets Used in Text Categorization ……………………...28

2.8.1 Reuters-21578 ……...……………………….............28

2.8.2 20 Newsgroups (20 NG) …..……………….............30

2.8.3 Classic3 …………………………………………….30
2.8.4 World Wide Knowledge Base (WebKB) …...............31

**Chapter 3: Content Based Text Classification**

3.1 Introduction …........………………………………….....32
3.2 The Structure of Text Mining Software ...............................32
3.3 Data Preprocessing …………………………......................36
3.3.1 Tag Removal ………………….………......................36
3.3.2 Stop Word Removal ……………………………...........37
3.3.3 Special Character Removal …………….……………....38
3.3.4 Porter Stemming ……………………..…..................39
3.4 Data Classification ………….... …………………….…..44
3.4.1 Partitioning Document to Training and Testing …......... 44
3.4.2 TFIDF Weighting Classifier ……………........……….45
3.4.3 Category Construction ……………………..............46
3.4.4 Document Classification Using Cosine Similarity …......51
3.4.5 Evaluation of Classification Measurement …..................52

**Chapter 4: Tests and Results**

4.1 Introduction ………………………………………………54
4.2 Experimental Results ……………………………………...54
4.2.1 Preprocessing Phase Results …………………………...54
4.2.2 Document Classification Results ………………………55
4.2.3 Classification Performance Results ……………………60
4.3 Results Discussion ……………………………………………63

**Chapter 5: Conclusion and Future work**

5.1 Conclusion ……………………………………….…..……65
5.2 Future work ……………………………………………....66

# *List of Abbreviations*

| | |
|---|---|
| **20NG** | 20 News Group |
| **DF** | Document Frequency |
| **IE** | Information Extraction |
| **IR** | Information Retrieval |
| **KDD** | Knowledge Discovery in Database |
| **KDT** | Knowledge Discovery from Text |
| **KNN** | K-Nearest Neighbor |
| **NLP** | Natural Language Processing |
| **P** | Precision |
| **Q&A** | Question  Answering |
| **R** | Recall |
| **SVM** | Support Vector Machine |
| **TC** | Text Categorization or Classification |
| **TF** | Term Frequency |
| **TFIDF** | Term Frequency Inverse Document Frequency |
| **TM** | Text Mining |
| **WebKB** | World Wide Knowledge Base |

# Chapter One
# Principles of Text Mining

# *Chapter One*

# *Principles of Text Mining*

## 1.1 What is Text Mining?

Information that exists on the World Wide Web and the users that have access to it or produce it have reached outrageous numbers. This state is not static, but a dynamic continuingly changing condition that converts the Internet into a chaotic system. Focusing on the needs of the Internet users who access news information from major or minor news portals. This means that if one wants to find information regarding to a specific topic, he will have to search one by one, at least the major portals, and try to find the news of his preference. A better solution is to access every site and search for a specific topic if a search field exists in the portals, doing that by using text mining technologies by extracting features for each text document **[Tak03]**.

TM has been defined as the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources, it is the process of analyzing collections of textual materials in order to capture key concepts and themes, and uncover hidden relationships and trends, and it is sometimes confused with IR which is the process of finding documents that are relevant to a particular topic. When you carry out a keyword search through a search engine, the underlying technology is IR. By contrast, TM provides a way to examine a collection of documents and discover the concepts contained in it, without requiring that you know the precise words or terms authors have used to express those concepts. TM is similar to data mining, except that data mining tools are designed to handle structured data from databases or XML files, but TM can work with unstructured or semi-structured data sets such as emails, full-text documents, HTML files, etc **[Lid00]**.

The problem introduced by TM is obvious: natural language was developed for humans to communicate with one another and to record information, and computers are a long way from comprehending natural language. Humans have the ability to distinguish and apply linguistic patterns to text and humans can easily overcome obstacles that computers cannot easily handle such as slang, spelling variations and contextual meaning. However, although our language capabilities allow us to comprehend unstructured data, we lack the computer's ability to process text in large volumes or at high speeds **[Tak03]**.

## 1.2 Technologies of Text Mining

Some of the technologies that have been developed and can be used in the text mining process are **[Zha05, Hei04 and Lid00]**:

1. **Information extraction:** identifies key phrases and relationships within text. It does this by looking for predefined sequences in text, a process called pattern matching. For example, given the sentence "Area relatives of a man being held hostage in Iraq waited for word about him Saturday as militants threatened to decapitate him, another American and a Brit unless demands were met within 48 hours", information extraction software should identify two American hostages and a British hostage, militants, and the relatives of one of the hostages as people; Iraq as the place; and Saturday as the time. The software infers the relationships between all the identified people, places, and time to provide the user with meaningful information. Almost all text mining software uses information extraction since it is the basis for many of the other technologies discussed below

2. **Topic tracking:** A topic tracking system works by keeping user profiles and, based on the documents the user views, predicts other documents of interest to the user. It has some limitations, however. For example, if a user sets up an alert for “text mining”, she/he will receive several news stories on mining for minerals, and very few that are actually on text mining. Some of the better text mining tools let users select particular categories of interest or the software automatically can even infer the user’s interests based on his/her reading history and click-through information.
3. **Summarization:** Text summarization is immensely helpful for trying to figure out whether or not a lengthy document meets the user’s needs and is worth reading for further information. With large texts, text summarization software processes and summarizes the document in the time it would take the user to read the first paragraph. The key to summarization is to reduce the length and detail of a document while retaining its main points and overall meaning. The challenge is that, although computers are able to identify people, places, and time, it is still difficult to teach software to analyze semantics and to interpret meaning. Generally, when humans summarize text, we read the entire selection to develop a full understanding, and then write a summary highlighting its main points.
4. **Categorization or Classification:** It involves identifying the main themes of a document by placing the document into a pre-defined set of topics. Categorization only counts words that appear and, from the counts, identifies the main topics that the document covers. Categorization often relies on a thesaurus for which topics are predefined, and relationships are identified by looking for broad terms, narrower terms, synonyms, and related terms. This technology is used to classify collection of documents in this thesis.

5. **Clustering:** Is a technique used to group similar documents, but it differs from categorization in that documents are clustered on the fly instead of through the use of predefined topics. Another benefit of clustering is that documents can appear in multiple subtopics, thus ensuring that a useful document will not be omitted from search results. Clustering technology can be useful in the organization of management information systems, which may contain thousands of documents.
6. **Concept linkage:** It connects related documents by identifying their commonly shared concepts and help users find information that they perhaps wouldn't have found using traditional searching methods. It promotes browsing for information rather than searching for it. Concept linkage is a valuable concept in TM, especially in the biomedical fields where so much research has been done that it is impossible for researchers to read all the material and make associations to other research. Ideally, concept linking software can identify links between diseases and treatments when humans can not.
7. **Information visualization:** Visual text mining, or information visualization, puts large textual sources in a visual hierarchy or map and provides browsing capabilities, in addition to simple searching. The user can interact with the document map by zooming, scaling, and creating sub-maps. Information visualization is useful when a user needs to narrow down a broad range of documents and explore related topics.
8. **Question answering (Q&A):** It deals with how to find the best answer to a given question. Q&A can utilize multiple TM techniques. For example, it can use information extraction to extract entities such as people, places, events; or question categorization to assign questions into known types (who, where, when, how, etc.). In addition

to web applications, companies can use Q&A techniques internally for employees who are searching for answers to common questions. The education and medical areas may also find uses for Q&A in areas where there are frequently asked questions that people wish to search.

## 1.3 Literature Survey

Some of related works are listed below:

**1- 2002 "Hany saleeb"(A framework for Recommending text –based Classification Algorithms) [Sal02]:**

PhD thesis addresses text categorization methods and algorithms in two areas: first, the development of tools for supporting users of machine learning and data mining algorithms in the area of text classification, second, is to investigate how text classification is performed on the Web. The structural information that is inherent to documents on the Web is usually neglected. In analyzing Web documents, the relative importance of hypertext tags is investigated in order to ascertain their relative importance in predicting the relevance of unknown documents.

**2- 2003 “Malcolm Walter Corney” (Analyzing Email text authorship for forensic purposes) [Cor03]:**

Master thesis uses machine learning and text categorization algorithm to find authorship characterizations based on sociolinguistic cohorts, such as gender and language background, as a technique for profiling the anonymous message so that the suspect list can be reduced. E-mail messages will contain macro-structural features that can be measured. These features together can be used with the Support Vector Machine learning algorithm (mentioned in section 2.3.4) to classify e-mail messages to an author providing a suitable sample of messages is available for comparison.

**3- 2003”Hiroya Takamura” (Clustering Approaches to Text Categorization) [Tak03]:**

PhD thesis improving the accuracy of text categorization, which is the basis for various applications such as e-mail classification and Web-page classification, using clustering approaches which is usually regarded as an unsupervised learning method and categorization as a supervised learning, and it shows that clustering can be used to improve accuracy of text categorization, this done by using a co-clustering method to improves the performance of Naive Bayes classifiers (mentioned in section 2.3.1).

**4- 2005 "Yevgeny Agichtein" (Extracting Relation from Large Text Collections) [Agi05]:**

PhD thesis addresses two fundamental problems in extracting relations from large text collections: (1) Portability: tuning extraction systems for new domains (2) Scalability: scaling up information extraction to large collections of documents. To address the first problem, it develop the Snowball information extraction system, a domain-independent system that learns to extract relations from unstructured text based on only a handful of user-provided example relation instances. To address the second problem, it develop the QXtract system, which learns search engine queries that retrieve the documents that are relevant to a given information extraction system and extraction task. QXtract can dramatically improve the efficiency of the information extraction process, and provides a building block for extracting structured information and text data mining from the web at large.

**5- 2006 “Abdul Kareem Murhij Radhi” (Machine Learning for Text Categorization) [Rad06]:**

PhD thesis uses the field of artificial intelligent via studying machine learning for natural language understanding using decision tree algorithm. The proposed system extracts concepts from printed text in natural language. Extracted data are particularly useful for obtaining a structured database from unstructured documents, and preparing information for database entries. In this work an intersection between machine learning and natural language processing to reach the understanding of the language via extracting concept mapping.

## 1.4 Aim of Thesis

Classifying unstructured data (such as text) is very important topic of text mining, so the subject of this thesis is to analyze collection of documents and classifying then into a set of predefined classes according to the main topic depends on the content of these documents, by extracting features from the documents such as frequency of words or text terms and apply a cosine similarity method to classify each document to which of the predefined classes it will belongs.

## 1.5 Thesis Layout

The outline of the thesis is as following:

- **Chapter Two:** Introduces a background to text mining system, text categorization technique, natural language processing and information extraction with it is relation to text mining.
- **Chapter Three:** Presents the general description of the proposed system with the details of design and implementation of its method.
- **Chapter Four:** Shows the results for each step obtained from implementation of the proposed classification method.
- **Chapter Five:** Gives some conclusion and suggestion for future work.

# Chapter Two

## Text Mining (Analysis & Technique)

# *Chapter Two*
# *Text Mining*
# *(Analysis and technique)*

## 2.1 <u>Introduction</u>:

Text mining or knowledge discovery from text (KDT) deals with the machine supported analysis of text. It uses techniques from IR, information extraction as well as natural language processing (NLP) and connects them with the algorithms and methods of data mining, machine learning and statistics. 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 preprocessing methods and algorithms are required in order to extract useful patterns. TM refers generally to the process of extracting interesting information and knowledge from unstructured text. Text mining tools that allow us to sift through this information with ease will become more and more valuable **[Nso06]**.

Text mining purposes involves **[Tka98]**

- Getting some texts relevant to the domain of interest
- Representing the content of the text in some medium useful for processing (natural language processing, statistical modeling, etc.)
- Doing something with the representation (finding associations, dominant themes, etc.).
- Discover and use knowledge that is contained in a document collection as a whole, extracting essential information from document collections and from a variety of different sources.

## 2.2 Text Mining Application

Unstructured text is very common, and in fact may represent the majority of information available to a particular research and here is some typical application of text mining **[Lid00, Apt94 and Tak03]**:

a. **Analyzing open-ended survey responses.** In survey research (e.g., marketing), it is not uncommon to include various open-ended questions pertaining to the topic under investigation. The idea is to permit respondents to express their "views" or opinions without constraining them to particular dimensions or a particular response format. This may yield insights into customer's views and opinions that might otherwise not be discovered when relying solely on structured questionnaires designed by "experts."
b. **Automatic processing of messages, emails, etc.** Another common application for text mining is to aid in the automatic classification of texts. For example, it is possible to "filter" out automatically most undesirable "junk email" based on certain terms or words that are not likely to appear in legitimate messages, but instead identify undesirable electronic mail. In this manner, such messages can automatically be discarded. Such automatic systems for classifying electronic messages can also be useful in applications where messages need to be routed (automatically) to the most appropriate department or agency; e.g., email messages with complaints or petitions to a municipal authority are automatically routed to the appropriate departments; at the same time, the emails are screened for inappropriate or obscene messages, which are automatically returned to the sender with a request to remove the offending words or content.
c. **Analyzing warranty or insurance claims, diagnostic interviews, etc.** In some business domains, the majority of information is

collected in open-ended, textual form. For example, warranty claims or initial medical (patient) interviews can be summarized in brief narratives, or when you take your automobile to a service station for repairs, typically, the attendant will write some notes about the problems that you report and what you believe needs to be fixed. Increasingly, those notes are collected electronically, so those types of narratives are readily available for input into text mining algorithms.

d. **Investigating competitors by crawling their web sites.** Another type of potentially very useful application is to automatically process the contents of Web pages in a particular domain. For example, you could go to a Web page, and begin "crawling" the links you find there to process all Web pages that are referenced. In this manner, you could automatically derive a list of terms and documents available at that site, and hence quickly determine the most important terms and features that are described. It is easy to see how these capabilities could efficiently deliver valuable business intelligence about the activities of competitors.

e. **In education.** Students and educators can find more information relating to their topics at faster speeds than they can use traditional searching method.

Here are some examples of real text mining application:

- Sales and marketing executives at Compaq Computer Corp. count on text mining tools to analyze company descriptions in their prospect database. The results help executives target customers for new sales and marketing campaigns.
- Linguists at the Universite Catholique de Louvain in Belgium use text mining to analyze summaries of ancient and modern texts. Researchers mine textual information in several languages and use the results to address philological and psychological questions.

- A new text mining project at the University of Louisville Medical Center will let doctors make better use of medical databases. Search results of these medical databases can often yield 2,000 matches, but advanced modeling with Enterprise Miner can reduce the results to 100 highly relevant documents and sort those 100 documents into smaller subgroups or categories **[Wei04]**.

So, in this chapter the theoretical background for TM concentrating on text categorization (classification) technique as powerful tool in text mining, which represent the main principles required to design the proposed text mining algorithm used in the next chapter.

## 2.3 Text Categorization or Classification (TC) Technique

Text categorization is the classification of documents with respect to a set of one or more categories. Each category is associated with a single concept or idea such as sports, science or history, etc. Categorization may be used to filter out documents or parts of documents that are unlikely to contain extractable data, without incurring the costs of more complex natural language processing. For example, a news information retrieval system processes texts in many subject areas. Thus, categorization may be used to route stories to category specific topics **[Cor03]**.

Sometimes the task of TC is to determine the part of speech that a word has in some text; in other cases it may involve classifying a sample into relevant subject categories. Typically the text is represented using some set of features extracted from the sample. This ranges from using each single word as a feature to using logical sentences built to correspond to the meaning of a segment of text **[Tak03]**.

There are two main kinds of TC problems are *Single-label TC* exactly one category must be assigned to a document d. That is, each document has one label assigned to it by the correct category, and it is also called binary categorization; an example of the classifier of this type is support vector machine, while the second is *Multi label TC* any number of categories can be assigned to the specified document d as shown in Fig. [2.1]. That is, each document has one or more labels assigned to it by the correct category, and it is also called multi class categorization, an example of the classifier of this type is naive Bayes (mentioned in section 2.3.1) **[Agi05]**.

Text categorization problem appears in a number of application domains including IR data or text mining and Web searching. The application of text categorization is as following **[Rad06]**:

A. Document organization as in Newswire filtering or grouping of news stories produced by news agencies to useful classes of interest.

B. Text Filtering

   I. Classifying incoming stream of documents news group messages like (Spam e-mail filtering, recommending systems).

   II. Single-label TC in news articles like personalized newspaper (Classifying into two disjoint categories: relevant and irrelevant).

C. Authorship attribution and etc.

Text categorization has become one of the most important techniques in text mining. Its task is to automatically classify documents into predefined categories or classes based on their content. In the real world task, it is very hard to obtain the large labeled documents, which are mostly produced by human **[Pot05].**

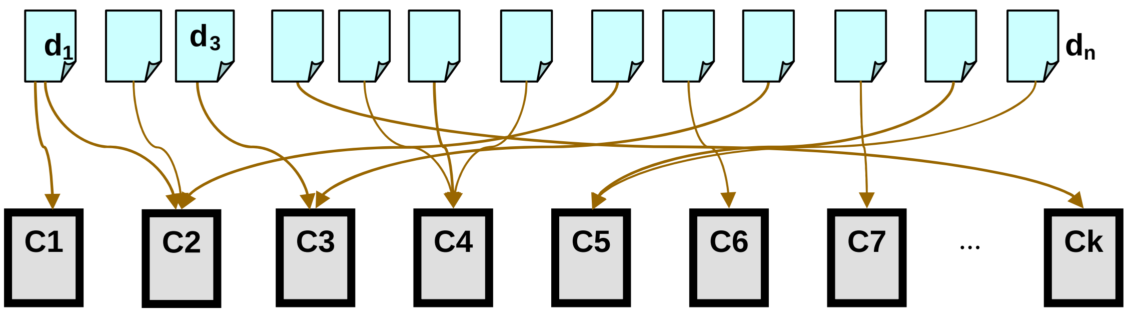


**Figure (2.1): Multi label Text Categorization**

The text categorization task is of assigning free documents expressed in natural language into one or more thematic classes (categories) belonging to the predefined set k classes

C = {c1; : : : : : : : : ; ck}

The construction of a text classifier relies on an initial collection of documents pre-classified under C.

D = {d1; : : : : : : : : ; dn}

Let D be the finite training document set and

W = {w1; : : : : : ;wv}

W are the set containing v distinct labels of training documents to an appropriate class. Then a set of document vectors D and their associated class labels W, will be calculated so the problem in TC is to estimate the correct class label of a free document for the whole proposed classifier. The construction of a text classifier consists of essentially three phases: document indexing by locating document to specific class, classifier learning by constructing the classes and classifier evaluation by calculating the performance of the classifier **[Pen03]**.

Some Methods that are used in text categorization technique are as follows:

## 2.3.1 <u>Naïve Bayes</u>

The naïve Bayes (NB) classifier is a probabilistic model that uses the joint probabilities of terms and categories to estimate the probabilities of categories in a given test document. The naïve part of the classifier comes from the simplifying assumption that all terms are conditionally independent of each other in a given category. Because of this independence assumption, the parameters for each term can be learned separately and this simplifies and speeds the computing operations compared to other classifiers **[Ozq04]**.

In this model classification of test documents is performed by applying the Bayes' rule:

$$P(c_j|d_i)=\frac{P(c_j).P(d_i|c_j)}{P(d_i)} \qquad (2.1)$$

Where $d_i$is a test document and $c_j$ is a category. The posterior probability of each category $c_j$ given the test document$d_i$, i.e.$P(c_j|d_i)$, is calculated and the category with the highest probability is assigned to $d_i$ . In order to calculate $P(c_j|d_i), P(c_j)$and $P(d_i|c_j)$ have to be estimated from the training set of documents. Note that $P(d_i)$ the same for each category so it can be eliminate from the computation. The category prior probability $P(c_j)$ can be estimated as follows:

$$P(c_j)=\frac{\sum_{i=1}^{N} y(d_i,c_j)}{N} \qquad (2.2)$$

Where, N is number of training documents and $y(d_i,c_j)$ =1 if $d_i \in c_j$ and is equal to 0 otherwise. So, prior probability of category $c_j$ is estimated by the fraction of documents in the training set belonging to$c_j$ **[Kam02]**.

### 2.3.2 K-Nearest Neighbor

K-Nearest Neighbor (KNN) is another type of text classification methods. Given a test document, the system finds the k nearest neighbors among the training documents, and uses the categories of the k neighbors to weight the category candidates. K-Nearest Neighbor classification is based on the assumption that the characteristics of members of the same class should be similar, and thus, observations located close together in covariate space are members of the same class **[Pot05]**.

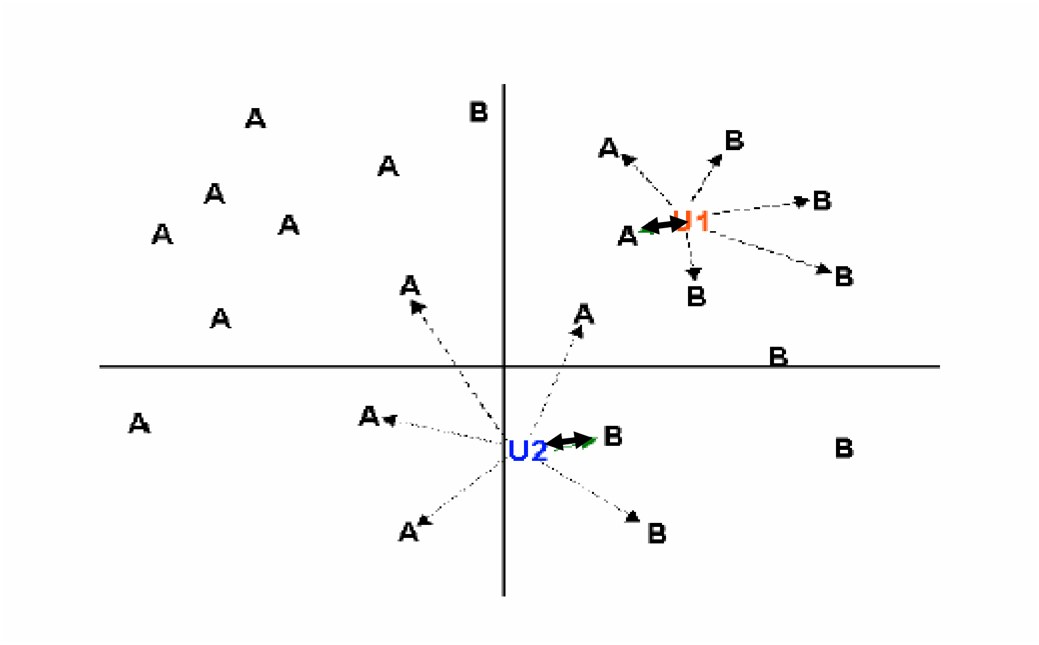


**Figure (2.2): k-Nearest Neighbor [Sal02]**

For example, Fig. [2.2] depicts a simplified example of the classification of unknown observations, U1and U2. It shows, unknown documents U1 and U2 are members of one of two category topics, A or B. Doted arrows represent the distance from the unknown observations to their nearest neighbors. Using a K=1 nearest neighbor, classification rule (double headed solid arrows), to U1 and U2 would be classified as members of groups A and B, respectively. Using K=6 nearest neighbor, classification rule (all arrows), would classify U1 and U2 as members of groups B and A, respectively **[Sal02]**.

## 2.3.3 Decision tree

Decision trees are classifiers which consist of a set of rules which are applied in a sequential way and finally yield a decision. They can be best explained by observing the training process, which starts with a comprehensive training set. It uses a divide and conquer strategy: For a training set *M* with labeled documents the word *ti* is selected, which can predict the class of the documents in the best way, e.g. by the information gain. Then *M* is partitioned into two subsets, the subset *M+i* with the documents containing *ti*, and the subset *M-i* with the documents without *ti*. This procedure is recursively applied to *M+i* and *M-i*. It stops if all documents are belonging to the same class. It generates a tree of rules with an assignment to actual classes in the leaves. They are fast and scalable both in the number of variables and the size of the training set **[Nah04]**.

For text mining, however, they have the drawback that the final decision depends only on relatively few terms. A decisive improvement may be achieved by boosting decision trees i.e. determining a set of complementary decision trees constructed in such a way that the overall error is reduced. Use even simpler one step decision trees containing only one rule and get impressive results for text classification **[Tai02]**.

Decision trees which are used to predict categorical variables are called classification trees because they place instances in categories or classes. Decision trees used to predict continuous variables are called regression trees. Decision trees make few passes through the data (no more than one pass for each level of the tree) and they work well with many predictor variables. As a consequence, models can be built very quickly, making them suitable for large data sets. Decision trees that are not limited to unvaried splits could use multiple predictor variables in a single splitting rule. Decision trees handle non-numeric data very well. This ability to accept categorical data minimizes the amount of data transformations and the explosion of predictor variables inherent in neural nets **[Ali06]**.

## 2.3.4 Support Vector Machine(SVM)

A support vector machine is a multi class machine learning method that divides space into a training positive examples side and a negative examples side. The idea of SVM is to find such linear separators. For linearly separable two-class data; the (linear) SVM computes the maximum margin hyperplane that separates the classes. For non-linearly separable data there are two possible extensions. Where SVM are supervised learning methods that belong to a family of generalized linear classifiers. SVM calculates the hyperplanes that separate a positive example from a negative example in hyperspace. The distance between the positive-side hyperplane nearest the negative examples and the negative-side hyperplane nearest the positive examples is called the margin **[Liu03]**.

The SVM method has been applied to tasks such as handwritten digit recognition, object recognition, and text classification. The SVM is trained using preclassified documents to enhance classification performance and it is capable of constructing models that are complex enough to solve difficult real world problems. At the same time, SVM models have a simple functional form and are amenable to theoretical analysis **[Zha05]**.

## 2.3.5 TFIDF (Term Frequency Inverse Document Frequency)

The TFIDF weighting method is a weight classifier often used in TM. This weight is a statistical measure used to evaluate how important is a term to a document in a collection or corpus. The importance of a term increases proportionally to the number of times a term appears in the document but is offset by the frequency of the term in the corpus and it is applications in Vector Space Model .The TFIDF weighting scheme is often used in the vector space model together with cosine similarity to determine the similarity between two documents **[Tur01]**.

Vector space model (or term vector model) is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers. A document is represented as a vector. Each element or cell in the vector corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best known schemes is TFIDF weighting **[Bas03]**.

The definition of term depends on the application. Typically term is a single (separate) word, keyword, or longer phrase. If the words are chosen to be the terms, the number of the element in the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus) of documents. In the Vector Space Model, documents are represented as vectors of terms:

$$T_i =< t_{i1}, t_{i2}, ..., t_{ik}, ..., t_{in} >$$

Where $t_{ik}$ is the $kth$ term in the $ith$ document, and $n$ is a number of terms being used in the document. Each document d is represented as a vector of weights $d = (w^{(1)}, ......., w^{(n)})$ so that documents with similar content have similar vectors (according to a fixed similarity metric). Each $w^{(i)}$ represents a weight of distinct term $t_i$ **[Rad06]**.

TC aims at assigning pre-defined classes to text documents. TM classification task starts by taking a training set of documents that are already labeled with a specific class. The task is then to determine a classification model which is able to assign the correct class to a new document (testing document) of the domain. In order to categorize different document for example stream of newswire story into set of predefined categories using a similarity measure based on TFIDF **[Bil06]**.

This classification method works by calculate the following:

**a) Term Frequency (TF) and Document frequency (DF).**

Now, $w^{(i)}$ for a document d is calculated as the combination of the statistics $TF(t_i,d)$ and $DF(t_i)$.The term frequency $TF(t_i,d)$ is the number of term $t_i$ occurs in document d and the document frequency $DF(t_i)$ is the number of documents in which term $t_i$ occur at least once. The inverse document frequency $IDF(t_i)$ can be calculated from the document frequency.

$$IDF(t_i) = \log_e\left(\frac{D}{DF(t_i)}\right) \qquad (2.3)$$

Here, $D$ is the total number of documents. Intuitively, the inverse document frequency of a term is low if it occurs in many documents and high if the term occurs in only one. The so-called weight $w^{(i)}$ of term $t_i$ in document d is then **[Nah04]**

$$w^{(i)} = TF(t_i,d).IDF(t_i) \qquad (2.4)$$

For example If the term represent a word in the following document that contains 100 total words and a word appears 3 times, then the term frequency of the word in the document is 0.03. One way of calculating DF is to determine how many documents containing the word divided by the total number of documents in the collection. So if a word appears in 1,000 documents out of a total of 10,000,000 then the document frequency is

0.0001. Alternatives to this formula are to take the $\log e$, is mathematics written as *ln* of the document frequency as above. Then in this example IDF would be= $\log e$ (10,000,000/1,000) = 9.21, so TFIDF weight of the word = 0.03 * 9.21 = 0.28 **[Mcs04]**.

**b) Cosine Similarity**

A standard way for similarity measure is the cosine similarity. *cosine similarity* is a measure of similarity between two vectors of *n* dimensions by finding the angle between them, often used to compare documents in text mining. Given two vectors of attributes, $d_1$ and $d_2$, the cosine similarity, θ, is represented using a dot product as

$$\cos\theta = \frac{d_1.d_2}{\|d_1\|.\|d_2\|} \tag{2.5}$$

For text matching, the attribute vectors $d_1$ and $d_2$ are usually the TFIDF vectors of the documents. Since the angle, θ, is in the range of [0, π], the resulting similarity will yield the value of π as meaning exactly opposite, π / 2 meaning independent, 0 means exactly the same, with in-between values indicating intermediate similarities or dissimilarities. Where $\|d\|$ is a Euclidean norm function on real number of the vector $d = [w_1,.......,w_n]$ that assigns a strictly positive length or size to all vectors in a vector space which is captured by the following equation **[Wei04]**.

$$\|d\| = \sqrt{d.d} = \sqrt{\sum_{i=1}^{n}(w_i)^2} \tag{2.6}$$

The set of documents in the collection then may be viewed as a set of vectors in a vector space, in which there is one cell for each term. This representation loses the relative ordering of the terms in each document. This measure suffers from a drawback; two documents with very similar content can have a significant vector difference simply because one is much longer than the other. Thus the relative distributions of terms may be identical in the two documents, but the absolute term frequencies of one may be far larger **[Xin04]**.

To reduce the problem of finding the document(s) most similar to document *d* or not is that by finding the highest angel value. Computing cosine similarity measure between document *d* and each document in the set $D = [d_1, \cdots\cdots\cdots, d_N]$, and then picking off the highest resulting cosine value. In classification the same is done by take document from testing documents and compute cosine similarity between each of predefined class's document and assign the document to the class which gives the highest similarity value **[Kam02]**.

## 2.4 Evaluating Performance of Text Categorization

There are various measures of categorization performance that can be calculated for each classification experiment. These familiar machine learning measures include:

- Precision (P)
- Recall (R)
- F measure performance combined precision and recall
- Breakeven point when P=R
- Error Rate
- Accuracy

Given a classifier whose input is a document, and whose output is list of categories assigned to that document, the recall and precision can be computed at any threshold on this list:

$$recall = \frac{categories\ found\ and\ correct}{total\ categories\ correct} \qquad (2.7)$$

$$precision = \frac{categories\ found\ and\ correct}{total\ categoreis\ found} \qquad (2.8)$$

The category assignments of a binary classifier can be evaluated using a two-way contingency as shown in table (2.1) for each category, which has four cells: where

- cell a count the documents correctly assigned to this category;
- cell b counts the documents incorrectly assigned to this category;
- cell c counts the documents incorrectly rejected from this category;
- cell d counts the documents correctly rejected from this category.

Conventional performance measures are defined and computed from this contingency table. These measures are recall (*r)*, precision (*p*), fallout (f*)*, accuracy (*Acc*) and error (*Err*):

- r = a / (a+c) if (a+c)>0, otherwise undefined;
- p = a / (a+b) if (a+b) >0, otherwise undefined;
- f = (2*r*p) / (r+p) if (r+p) >0, otherwise undefined;
- Acc = (a+d)/n where n = a+b+c+d >0;
- Err = (b+c)/n where n= a+b+c+d >0.

**Table (2.1) a contingency table**

| | Yes is correct | No is correct |
|---|---|---|
| Assigned Yes | a | b |
| Assigned No | c | d |

Although accuracy and error are common performance measures in the machine learning literature and have been used in some evaluations of text categorizations systems, there is a potential pitfall in using them to train or evaluate a binary classifier **[Yan99]**.

Precision measures the quality of classified data and is calculated for every class while Recall measures how good the system is in finding relevant documents for a given class. If the recall and precision are tuned to have an equal value, then this value is called the breakeven point of the system and F-measure score. The p or r measures above may be misleading when examined alone. Usually a classifier exhibits a trade-off between recall and precision, to obtain a high recall usually means sacrificing precision and vice versa **[Hei04]**.

Since the use of Acc and Err may be pitiful in binary classifier, so in this thesis we take the first three measures for the classification performance and ignore the use of accuracy and error measures.

## 2.5 Natural language processing (NLP) in TM

NLP or understanding (NLU) is the branch of linguistics which deals with computational models of language. Its motivations are both scientific (to better understand language) and practical (to build intelligent computer systems). NLP has several levels of analysis: phonological (speech), morphological (word structure), syntactic (grammar), semantic (meaning of multiword structures, especially sentences), pragmatic (sentence interpretation), discourse (meaning of multi-sentence structures), and world (how general knowledge affects language usage). When applied to text mining, NLP could in principle combine the computational (Boolean, vector space, and probabilistic). Clearly, NLP could be a powerful tool for text mining **[Tho99]**.

The general goal of NLP is to achieve a better understanding of natural language by use of computers. Others include also the employment of simple and durable techniques for the fast processing of text. The range of the assigned techniques reaches from the simple manipulation of strings to the automatic processing of natural language inquiries. In addition, linguistic analysis techniques are used among other things for the processing of text **[Apt94].**

The goal of TM, in fact, is "capturing semantic information" as tabular data. NLP includes stemming (morphological level), part-of-speech tagging (syntactic level), phrase and proper name extraction (semantic level), and disambiguation (discourse level). Goals include automating text mark-up for hypertext linkages in digital libraries, and machine learning algorithms for text classification **[Cor03]**.

Statistical method to NLP has been shown to significantly improve the accuracy of proper name classification, part-of-speech tagging, word sense disambiguation, and parsing under certain conditions, and tagging and disambiguation improve probabilistic document retrieval ranking discrimination by some parts of speech. Ultimately, lexical statistics are a reflection of term dependencies which in turn reflect natural languages relation to "naturally occurring dependencies in the physical world". However, higher level NLP proved far inferior to "shallow" tricks like stemming and query expansion in improving the performance of an advanced IR system under rigorous test conditions **[Cla03]**.

Computational linguistics is used as a synonym for NLP by some writers and as a narrower term by others. It is the branch of NLP which deals with finding statistical patterns in large text collections to inform algorithms for NLP techniques such as word sense disambiguation, and bilingual dictionary creation; i.e., computational linguistics is a form of TM. TM sub serves NLP, rather than the reverse **[Liu03]**.

## 2.6 Word Stemming

In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of TM applications. For this reason, a number of **stemming algorithms**, or **stemmers**, have been developed, which attempt to reduce a word to its **stem** or root form. Thus, the key terms of a document are represented by stems rather than by the original words. This not only means that different variants of a term can be conflated to a single representative form it also reduces the dictionary size, that is, the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in a saving of storage space and processing time **[Agi05]**.

Stemming maps several terms into one base form, which is then used as a term in the vector space model. This means that, on average, it increases similarities between documents because they have an additional common term after stemming, but not before. Stemming has a relatively low processing cost, especially when using Porter style stemming. It reduces the index size, and it usually slightly improves results. This makes it very attractive for use in TM **[Beu07]**.

For TM purposes, it doesn't usually matter whether the stems generated are genuine words or not thus, "computation" might be stemmed to "comput"  provided that (a) different words with the same 'base meaning' are conflated to the same form, and (b) words with distinct meanings are kept separate. An algorithm which attempts to convert a word to its linguistically correct root ("compute" in this case) is sometimes called a **lemmatiser**. Stemming, for instance, can reduce all words with the same root to a single form. There are three classes of word normalization:

1. Morphological stemming: terms like ‘retrieving’ use the stem ‘retriev’.
2. Lexicon-based: terms like ‘retrieval’ become ‘retrieve’.
3. Term clustering: Use classes such that terms ‘recover’, ‘fetch’ and ‘bring’ are all equivalent values **[Bil06]**.

Some stemming algorithm provided in NLP is illustrated as follow:

## 2.6.1 <u>Porter Stemmer</u>

The Porter Stemmer is a conflation Stemmer developed by Martin Porter at the University of Cambridge in 1980. The Porter Stemmer is a lexical-based stemmer that 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. This Stemmer is a linear step Stemmer. Specifically it has five basic steps applying rules within each step. Within each step, if a suffix rule matched to a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix was removed or changed, depends on the way defined by the rule. The Porter Stemmer is a very widely used and available Stemmer, and is used in many applications Algorithm. Porter's algorithm is probably the stemmer most widely used in TM & IR research, the porter stemming steps and their condition details will be discussed in appendix B **[Por00]**.

## 2.6.2 <u>Piace/Husk Stemmer</u>

The Paice/Husk Stemmer was developed by Chris Paice at Lancaster University in the late 1980s, and was originally implemented with assistance from Gareth Husk. The Paice/Husk Stemmer removes the endings from a word in an indefinite number of steps. The Stemmer uses a separate rule file, which is first read into an array or list. This file is divided into a series of sections, each section corresponding to a letter of the alphabet. When a word is to be processed, the stemmer takes its last letter and uses the index to find the first rule for that letter. If a rule is accepted then it is applied to the word. If it is not accepted, the rule index is incremented by one and the next rule is tried. However, if the first letter of the next rule does not match with the last letter of the word, this implies that no ending can be removed, and so the process terminates **[Pia94].**

### 2.6.3 Lovins stemmer

The Lovins Stemmer developed by Julie Beth Lovins of Massachusetts Institute of Technology in 1968. The Lovins Stemmer removes a maximum of one suffix from a word, due to its nature as single pass algorithm. It uses a list of about 250 different suffixes, and removes the longest suffix attached to the word, ensuring that the stem after the suffix has been removed is always at least 3 characters long. Then the ending of the stem may be reformed (e.g., by un-doubling a final consonant if applicable), by referring to a list of recoding transformations and so on **[Lov68]**.

### 2.6.4 Dawson's stemmer

The Dawson Stemmer was developed by J.L. Dawson of the Literary and Linguistics Computing Centre at Cambridge University. It is a complex linguistically targeted Stemmer that is strongly based upon the Lovins Stemmer, extending the suffix rule list to approximately 1200 suffixes **[Daw74]**.

## 2.7 Information Extraction(IE)

IE has emerged as a joint research area between TM and NLP. It is the process of extracting predefined information about objects and relationships among them from streams of documents and usually storing this information in pre-designed templates. It is very important to associate information extracting with streams of documents rather than static collections. As an example, some might extract information like promotions and sales from a stream of documents **[Hea00]**.

The information extracted might be the events, companies involved, or dates. Usually IE proceeds in two steps; first, it divides a document into relevant and irrelevant parts, and then fills the predefined templates with the information extracted from the relevant parts. In short, IE systems scan streams of documents in order to transform the associated documents into much smaller bits of extracted relevant information which are easier to be apprehended **[Cla98]**.

In general, the web contains millions of pages whose text hides data that would be best exploited in structured form. If we could build structured tables from the information hidden in unstructured text, then we would be able to run more complex queries and analysis over these tables, and report precise results. Extracting structured information from text has long been a rich subject of research **[Moo05]**.

The goal of IE methods is the extraction of specific information from text documents. These are stored in data base or files and are then available for further use. Natural language text contains much information that is not directly suitable for automatic analysis by a computer. However, computers can be used to sift through large amounts of text and extract useful information from single words, phrases or passages. The main task is to extract parts of text and assign specific attributes to it **[Bil06]**.

Two areas within IE are that of IR and TM. Both these techniques develop methods and tools for analyzing large data sets and for searching for unexpected relationships in the data, and involve the development of combinatorial pattern matching with statistical techniques and database methods. Data Mining (also known as Knowledge Discovery in Databases (KDD)) assumes that the data to be mined is already in the form of a structured database, whereas Text Mining discovers useful knowledge from unstructured, free text **[Hea00]**.

## 2.9 Data Sets Used in Text Categorization

There are many document data sets available for performing text categorization experiment on it and calculate categorization performance some of them are **[Tak03, Lin02]**:

### 2.9.1 Reuters-21578

The Reuters21578, Distribution 2.0 test collection is available from David D. Lewis professional home page, currently:

**http://www.research.att.com/~lewis**

The collection consists of 22 data files. Each of the first 21 files (reut2-000 through reut2-020) contains 1000 documents, while the last (reut2-021) contains 578 documents. An HTML DTD file describing the data file format and six files describing the categories used to index the data. All files are available uncompressed, and in addition a single zipped UNIX tar archive of the entire distribution is available as reuters21578.tar.gz. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The collection has only 21,578 documents, and thus is called the Reuters-21578 collection **[Tai02]**.

The files are in HTML format. Rather than going into the details of the HTML language, describe here is an informal description way on how the HTML tags are used to divide each file, and each document, into sections. Each of the 22 files begins with a document type declaration line: *<!DOCTYPE lewis SYSTEM "lewis.dtd">* The DTD file lewis.dtd is included in the distribution. Following the document type declaration line are individual Reuters articles marked up with HTML tags, as described below. Each article starts with an "open tag" of the form *<REUTERS>*. Each article ends with a "close tag" of the form:

*</REUTERS>* in all cases the *<REUTERS>* and *</REUTERS>* tags are the only items on their line **[Cla98]**.

Reuters-21578 data is used for binary classification, where each document is allowed to belong to one category. The documents can be assigned to 135 topic categories. The 10 most frequent categories contain about 75% of the documents which is they (earn, acq, ship, money, corn, grain, trade, interest, wheat, crude).

There are five major versions of the Reuters benchmarks:

1. Reuters-22173 (by Lewis)
2. Reuters-22173 (by Apte)
3. Reuters-22173 (by Weiner)
4. Reuters-21578 (All, by Apte)
5. Reuters-21578 (Top 10 categories, by Lewis)

Within this thesis, the benchmark Reuters-21578(Top 10 categories, by Lewis) data corpus is used that have been most popular among TC researchers. The results obtained allow determining the relative difficulty of these subsets, thus establishing an indirect means for comparing TC systems that have, or will be, tested on these different subsets **[Hea00]**.

This collection has several characteristics that make it interesting for TC experimentation:

- Similarly to many other applicative contexts, i.e. each document may belong to any category.
- The set of categories is not exhaustive, i.e. some documents belong to no category at all.
- The distribution of the documents across the categories is highly skewed, in the sense that some categories have very few documents classified under them while others have thousands **[Nah04]**.

### 2.9.2 20 Newsgroups (20NG)

This dataset is a collection of 20,000 messages collected from 20 different net news newsgroups. One thousand messages from each of the twenty newsgroups were chosen at random and partitioned by newsgroup name.

### 2.9.3 Classic3

This dataset is a collection of abstracts from three categories: *Medicine* (1033 abstracts from medical journals), *CISI* (1460 abstracts from IR papers); *Cranfield* (1399 abstracts from aerodynamics papers).

### 2.9.4 World Wide Knowledge Base (WebKB)

The World Wide Knowledge Base dataset (WebKB) is a collection of 8282 web pages obtained from four academic domains. The WebKB was collected by Craven (1998).

# Chapter Three

## Content Based Text Classification

# *Chapter Three*

# *Content Based Text Classification*

## 3.1 Introduction:

This thesis, deals with one of the popular area in text mining, namely text categorization which is the process of automatically labeling unlabeled text documents by their corresponding categories or classes based entirely on their content. A category can be a subject under which a document should be catalogued or stored. There are many different classification algorithms that evolved from different areas in statistics, machine learning and neural network. This chapter presents the use of text categorization technique in text mining. The well known TFIDF classifier method will be analyzed and implemented to newswire data to extract the main topic in it.

A proposed software system designed and implemented for text classification, started for the given input document for training and testing and to the final classified document will be discussed in details in the next sections.

## 3.2 The Structure of the Text Mining Software:

TM has the potential to reduce information overload by providing a user with patterns from the underlying text. In this thesis Text mining scenario starts by extracting the useful text from a large collection of document like removing HTML tags, stop words, special character and then applying some natural language processing techniques whish is represented in Porter stemming method as discussed in the previous chapter and weighting the word using TFIDF weighting method, finally perform categorization method to find out each document belong to which one of the predefined categories as shown in Fig. [3.1].

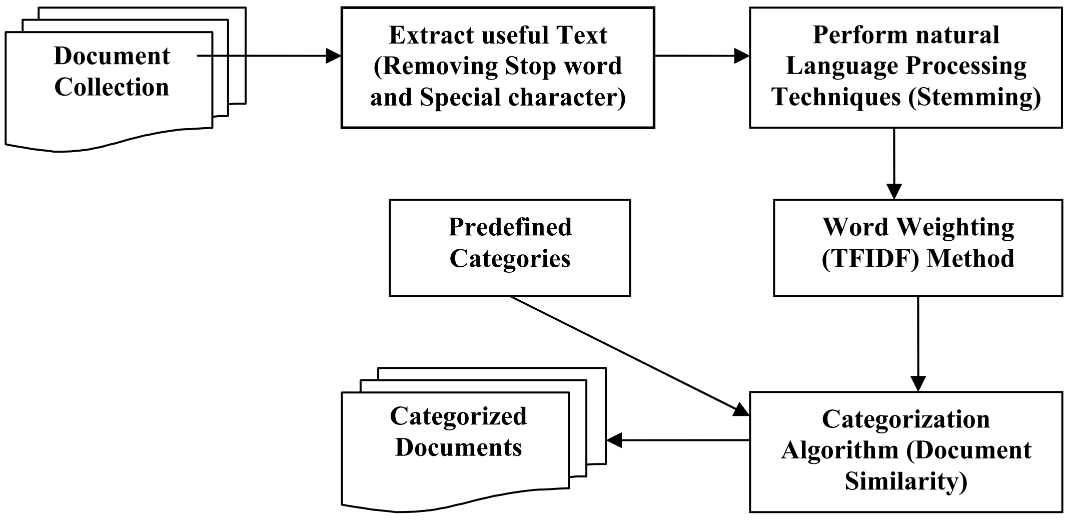

Document Collection
Extract useful Text (Removing Stop word and Special character)
Perform natural Language Processing Techniques (Stemming)
Predefined Categories
Word Weighting (TFIDF) Method
Categorized Documents
Categorization Algorithm (Document Similarity)

**Figure (3.1): Text mining workflow as proposed in this thesis**

TM goal is often have to handle large documents collections and automatically spot important information from these documents. The labeling of such large corpuses of documents is too expensive and impractical. In categorization technique the dominant approach is feature selection using various criteria. Traditional methods for feature subset selection in text categorization use an evaluation function that is applied to single words. All words are independently evaluated and sorted according to the assigned criterion. A software system as proposed in this thesis designed and implemented based on TFIDF weighting classifier method. The structure of the established system consists of two main modules as shown in Fig. [3.2], it consists of the data preprocessing phase as shown in Fig [3.2a] which is the first step in text categorization. In this step, a document transformation was made into a suitable representation for the learning algorithm of the classification task in the same time. The text transformation usually includes the following:

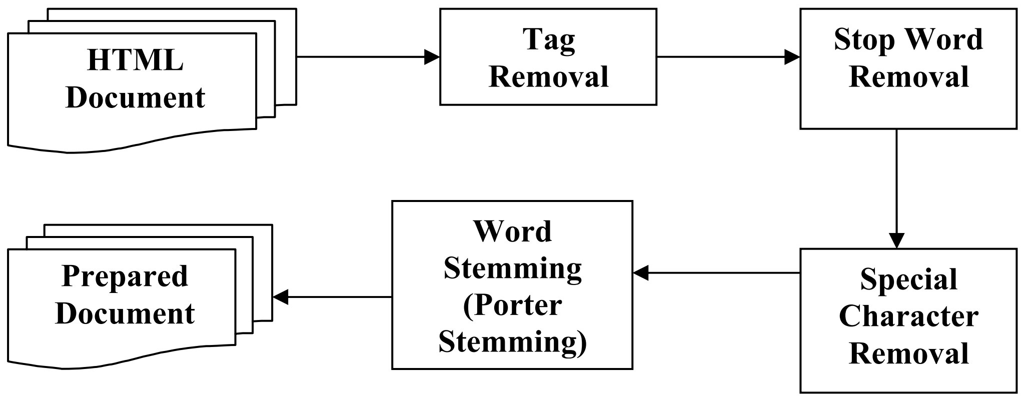
1. Remove HTML tags (since the chosen documents are all in HTML format).
2. Remove stop words.
3. Remove special characters.
4. Perform word stemming for English language.

Stop words are frequent words that carry little useful information such as pronouns, prepositions or conjunctions. Word stemming is the process of removing suffixes to generate word stems. This is done by grouping words that have the same conceptual meaning such as 'walk', 'walker', and 'walking.' Porter stemmer a well-known algorithm for this task is used for stemming in this thesis. All these processes are used to prepare data to the next phase.

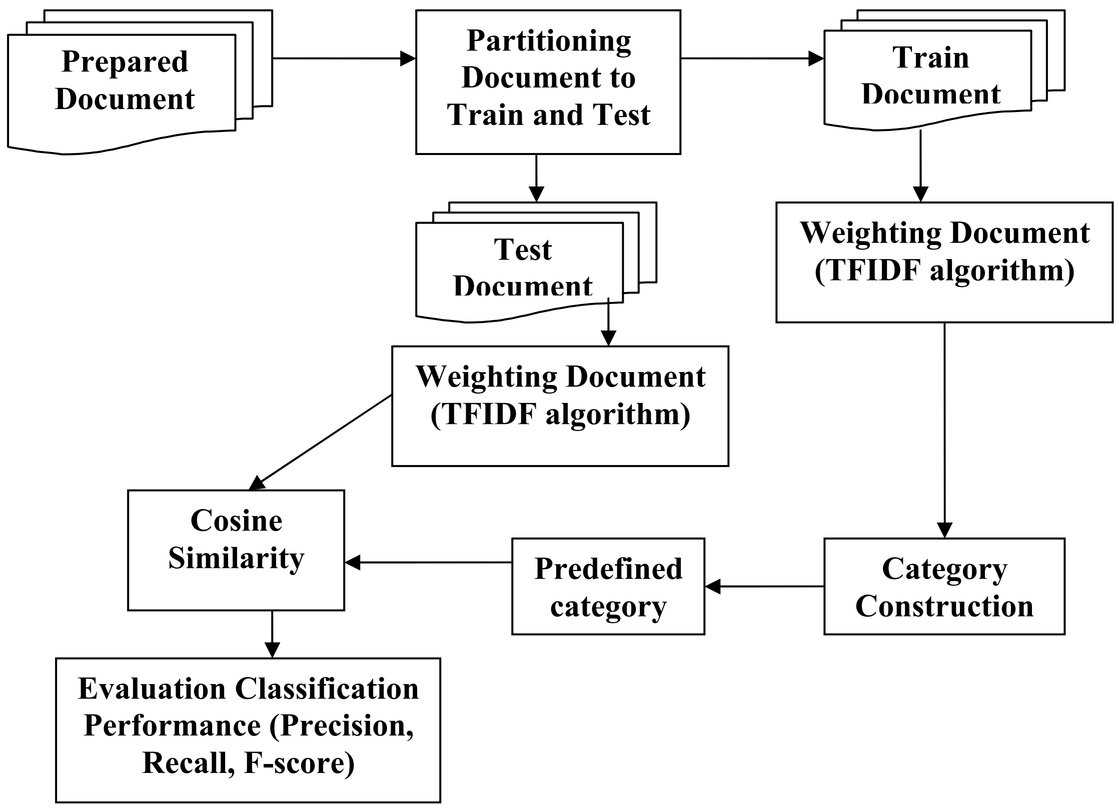
The second phase is the Classification Phase as shown in Fig [3.2b], which is the process of partitioning the documents into train and test documents then weighting words depending on their frequencies in the documents will be calculated. There are few ways for determining the word frequencies but there are two basic observations about text:

1. The more times a word occurs in a document, the more relevant it is to the topic of the document.
2. The more times the word occurs throughout all documents in the collection, the more poorly it discriminates between documents.

The term used for TFIDF weighting classification is a single word which is particularly related to finding out and constructing category for the classification purposes by dividing training set documents between the predefined category and applying cosine similarity to all document in each category to ensure that each document in the right category after that the classification of test document set is done by taking each document and compute cosine similarity with all categories and finding out to which category it will belong  then the performance measure of text classification calculation will be applied for TFIDF weighting algorithm. These models will be discussed in details in the next sections.



a. The block diagram of the data preprocessing ,


Prepared Document
Partitioning Document to Train and Test
Train Document
Test Document
Weighting Document (TFIDF algorithm)
Weighting Document (TFIDF algorithm)
Cosine Similarity
Predefined category
Category Construction
Evaluation Classification Performance (Precision, Recall, F-score)

b. The block diagram of the data classification

Figure (3.2): Categorization System Modules

## 3.3 Data Preprocessing:

All the HTML documents are preprocessed to retain only the bodies of each document discarding the tags in HTML documents. Then special characters will be eliminated from the document, and all words are will be converted to lower case. After this the stop word will be removed from the document. Finally word stemming will be performed to return each word to its root.

### 3.3.1 Tag Removal

When a web browser displays a page, it reads from a plain text file, and looks for special codes or "tags" that are marked by the < and > signs. The general format for the HTML tag is: <tag_name>string of text</tag_name> as an example, the title for this section uses a header tag: <h3>What are HTML tags?</h3> This tag tells a web browser to display the text What are HTML tags?.

In this step, an HTML tags will be removed from the document where the tags are not important in extracting useful features to be used for document classification.

**Algorithm (3.1) Removing tags.**

{ The implemented step to remove tags from HTML document}

```
Input:  HTML file.
Output: no-t: Text file after removing tags.
Procedure:
1. Initialize flag←0
2. Loop while not end of HTML file
3.   Read line from the file
4.     Check If the line do not contain tags Then
5.       Put line into no-t file
6.     Else
7.     Check If the line contain one or more tags in the same line Then
```

Continue

```
8.        Cut the line and Take the words between tags
9.        Put the word into no-t file
10.   Else
11.   Check If the line contain tag beginning and it is ending in another line Then
12.       Put the part before tag to no-t file
13.       Loop while flag=0
14.          Read line from HTML file
15.          Check If the line contain tag end Then
16.             Set flag←1
17.       End Loop
18.  Put the remain of line to no-t file
19.  Set  flag←0
20. End Loop
```

## 3.3.2 Stop Word Removal

Among many words, some words are very frequent to work as a useful feature, such as the verbs "be" and "have" can be seen in almost any documents. These words and many others words are called stop-words. Table 3.1 show sample of these stop words but the full set of stop words with their description as given in appendix A. These words are often removed from the data set. One problem in stop word elimination is that a word can be a stop-word for a data set, but can be a useful feature for another data set. But this problem is not a big issue since this state is not so frequent and removing stop word will be useful by making  data analyzes be easier .

**Table (3.1) Sample of Stop words**

| | | | | | | |
|---|---|---|---|---|---|---|
| a | an | the | and | but | if | or |
| because | as | until | while | of | at | by |
| for | with | about | against | between | into | through |

**Algorithm (3.2) Removing stop word.**

{The implemented step to remove stop word from the document}

**Input:** no-t: Text file, set of stop word.

**Output:** no-stop: Text file after removing stop word.

**Procedure:**

```
1. Loop while not end of no-t file
2.   Read word from the file
3.   Check If the word is not in stop word set Then
4.       Put the word to no-stop file
5. End Loop
```

## 3.3.3 Special Character Removal

Like stop word there are many characters appears in HTML documents do not carry any useful information, such as © or &. While many similar lists are available on the Web as stop word these character are too frequent and it is not useful to extract feature from it and removing it from data set will be useful for analyzing data more easily. The Special characters are listed in Table 3.2.

**Algorithm (3.3) Removing Special Character.**

{The implemented step to remove special character from the document}

**Input:** no-stop: Text file, set of special character.

**Output:** no-sp: Text file after removing special character.

**Procedure:**

```
1. Loop while not end of no-stop file.
2.   Read word from the file
3.   Check If word is not in special character Or not contain a special character Then
4.       Put word to no-sp file
5. End Loop
```

Table (3.2) Sample of Special Character

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ! | é | 0 | # | $ | **Á** | % | & | ' | ÷ | ( | ) | { |
| } | ê | 1 | [ | ] | " | * | + | - | . | / | : | ; |
| ♠ | È | ‡ | ♣ | ♥ | † | ◆ | ★ | ☆ | ← | ↑ | → | |
| **Â** | **À** | **2** | **Ã** | **Ä** | **Å** | **Æ** | **Ç** | **Ë** | **É** | **Ê** | **Ì** | **Í** |
| **Î** | ë | **3** | **Ï** | **Ð** | **Ñ** | **Ò** | **Ó** | **Ô** | **Õ** | **Ö** | **Ø** | **Ù** |
| **Ú** | î | **4** | **Û** | **Ü** | **Ý** | **Þ** | **ß** | **á** | **ã** | **ä** | **å** | **æ** |
| **è** | ï | **5** | **ì** | **í** | **ð** | **ñ** | **ò** | **ó** | **õ** | **ö** | **ø** | **ú** |
| < | ô | 6 | > | = | **¡** | ? | @ | \ | ^ | _ | ~ | \| |
| € | ü | 7 | ‚ | *ƒ* | **ª** | „ | … | † | ‡ | ˆ | ‰ | ‹ |
| ‘ | ù | 8 | ’ | “ | **²** | ” | • | — | ~ | ™ | › | œ |
| ¢ | û | 9 | £ | ¤ | **³** | ¥ | ¦ | § | ¨ | © | « | ¬ |
| ® | **ý** | | ¯ | ° | **º** | ± | 23 | ´ | µ | ¶ | · | ¸ |
| ¹ | **þ** | **ÿ** | ^ | » | **¿** | ¼ | ½ | ¾ | × | à | â | ç |

## 3.3.4 <u>Porter Stemming</u>

Stemming is used to reduce variant word forms to common root. One of the powerful algorithm used in stemming is Porter stemming which is the basic algorithm used in this work that presents a simple algorithm for English language words stemming it consists a set of rules as discussed in the previous chapter applied sequentially to the words in the text one after the other as commands in a program.

**Algorithm (3.4) Porter Stemming.**

{The implemented step to applying Porter stemming on the document}

**Input:** no-sp Text file, a vowel set as (a, e, i, o, u, y), a double set as (bb ,dd ,ff ,gg ,mm ,nn ,pp ,rr ,tt), a valid li-ending set as (c ,d ,e ,g ,h ,k ,m ,n ,r ,t ).

**Output:** stemmed: Text file.

**Procedure:**

```
1.  Initialize short← no, Initialize c←0
2.  Loop while not end of  no-sp file
3.  Read word from the file
4.      Calculate R1 is the region after the first non-vowel following a vowel
5.          Or the end  of the word if there is no non-vowel
6.      Calculate R2 is the region after the first non-vowel following  a vowel in R1
7.            Or the end of the word if there is no such non- vowel
8.      Call Algorithm (3.7) Short Syllable(flag)
9.      Check If flag = yes and R1 is null Then          Set short ← yes
10.     Call Algorithm (3.5) Exception (c)
11.     Check If c=1 Then       GoTo line 23
12.     Check If word begins with gener Or commun Or arsen Then
13.          Set R1 to be the remainder of the word
14.     Check If word is one of (sky, news, howe, atlas, cosmos, bias, andes) Then
15.         Leave word invariant , GoTo line 23
16.     Check If  suffix And condition is true Then
17.          Do actions as shown in table (3.3), GoTo line 23
18.     Check If short=yes Then     Add e to word , GoTo line 23
19.     Check If word end with e And in( R1 or R2)  And short=no Then    Delete e
20.     Check If word end with small letter (L) And in R2 And preceded  1 letter Then
21.          Delete small letter (L)
22.     Call Algorithm (3.6) Check Suffix
23.     Put the word after changing to stemmed file,   Set c←0   , Set short←no
24. End Loop
```

**Table (3.3) Word Suffix and Action applied on it**

| *Suffix* | *Condition* | *Action* |
|---|---|---|
| sses | non | Replace by ss |
| ied, ies | Preceded by more than one letter | Replace by i |
| ied, ies | Preceded by one letter | Replace by ie |
| s | Preceded by vowel not immediately before it | Delete s |
| us, ss | non | Do nothing |
| ogi | In R1 and preceded by small letter (L) | Replace by og |
| ed, edly, ing, ingly | Preceding by a vowel | Delete it |
| at or bl or iz | non | Add e |
| y | Preceded by a non-vowel which is not the first letter of the word | Replace by i |
| al, ance, ence, er, ic, able, ible, ant, ement, ment , ent, ism, ate, iti , ous, ive, ize | In R2 | Delete it |
| ion | Preceded by s or t | Delete it |
| ative | In R1 and In R2 | Delete it |
| li | In R1 and preceded by a valid li-ending | Delete it |
| ful, ness | In R1 | Delete it |
| Double as(bb ,dd ,ff ,gg ,mm ,nn ,pp ,rr ,tt) | non | Remove the last letter |

**Algorithm(3.5) Exception**

{ The implemented step to find if the word is exception or not}

```
Input: word.
Output: new word, c.
Procedure:
1. Initialize c←0
2. Check If word is in table (3.4) Then
3.     Convert as shown in table (3.4) , Set c ← 1
4. Return new word ,c
```

**Table (3.4) Exception Word and Conversion**

| *Word* | *Converted To* |
|---|---|
| skis | ski |
| skies | sky |
| dying | die |
| lying | lie |
| tying | tie |
| idly | idl |
| gently | gentl |
| ugly | ugli |
| early | earli |
| only | onli |
| singly | singl |

**Algorithm (3.6) Suffix**

{ The implemented step to check the suffix of the word}

**Input :** word

**Output :** new word

**Procedure:**

```
1. Check If suffix of word as in table (3.5) And in R1  Then
2.     Replace suffix with appropriate as shown in table (3.5)
3. Return (new word)
```

**Table (3.5) Word Suffix and Replacing applied on them**

| ***Suffix*** | ***Replaced By*** |
|---|---|
| tional | tion |
| eed, eedly | ee |
| enci | ence |
| anci | ance |
| abli | able |
| entli | ent |
| izer, ization | ize |
| ational, ation,ator | ate |
| alism, aliti, alli, alize | al |
| fullness, fulli | ful |
| ousli, ousness | ous |
| iveness, iviti | ive |
| biliti, bli | ble |
| lessli | less |
| icate, iciti, ical | ic |

**Algorithm 3.7 Short Syllable**

{The implemented step to check the short syllable of the word}

**Input:** Single word.

**Out put**: Boolean flag.

**Procedure**:

```
1. Initialize flag← no
2. Check If the word contain a vowel followed by a non-vowel except x,w ,y Then
3.     Set flag ← yes
4. Check If the word beginning by a vowel and followed by a non-vowel Then
5.     Set flag ← yes
6. Return flag
```

## 3.4 Data Classification

After elimination of data which is not useful in extracting features from text then the processing phase will be started by extracting features and then use the extracted features to find the categories for each document and then classifying the data according to which category it will belongs and calculating the classification performance measurement.

### 3.4.1 Partitioning Document to Training and Testing

Randomly divide the dataset into two parts, a training set and a testing set (the testing set is usually smaller than the training set, depending on the total size of the dataset). The selection of the document will be randomly chosen from the whole documents be generated based on number of the document size and put the document assigned by it to the train set until obtaining the required number of training documents. Then put the rest of articles into the test set. Where in this work the training document to be taken 70% from the whole documents collection and the remain 30% will goes to the testing documents as powerful partitioning to get high performance of the system.

## 3.4.2 TFIDF Weighting Classifier

TFIDF is the most common weighting method used to describe document collections in the Vector Space Model, in TC. Given a set of D total number of documents in it then applying the TFIDF weighting classifier for each word by compute TF and IDF as mentioned in the previous chapter where term is used as a single word and generate a vector space model contain each word and it is weight in the document number d according to the weighting equation that was mentioned in the previous chapter. Applying this algorithm for both training and testing files.

**Algorithm (3.8) TFIDF Weighting**

{The implemented step of the TFIDF weighting method}

**Input:** train: Text file or test: Text file.

**Output:** word vector, weight vector.

**Procedure:**

```
1.  Loop while not end of input(train or test) file
2.    Read single document from file
3.    Loop while not end of document          //compute term frequency
4.      Read word from the document
5.      Check If word is found in word vector Then
6.          Increment it is frequency by 1
7.      Else
8.          Add the word to word vector
9.          Set it is frequency to 1
10.   End Loop
11. End Loop
12. Loop while there is more document     //compute document frequency
13.   For each word in word vector
14.       Calculate number of document contain this word at least once
15.       Set document frequency by number of document contain this word
```

Continue

```
16.     End For
17.  End Loop
18.  For each word in word vector
19.      Set weight vector  =frequency*log base e( D/document frequency)
20.  End For
21. Return word vector, weight vector
```

## 3.4.3 Category Construction

In this thesis a new presentation is used for categorization based on the idea of placing the training document into a set of empty predefined categories. Instead of using a specific value of predefined category to classify document the category will be constructed according to the frequency measure of words in documents, for a single keyword for each category.

Category construction is implemented in the following steps:

**a. Locating Document to Appropriate Category**

In this step a training set of documents will be weighted using TFIDF weighting method as discussed in the previous chapter. Then a set of empty predefined category with the head keywords (earn, corn, trade, acq, grain, interest, crude, wheat, ship, money) each keyword represent a single category when the key word found in the document and it is weight is higher than the other categories keyword if found , this document placed in that category of high keyword weight and if the document do not contain any keyword of the set of predefined category it will go to the other which is a set of not predefined category, this process continue until there are no documents in the training set.

**Algorithm (3.9) Category Construction.**

{The implemented step of category construction }

**Input:** train: Text file, word vector, weight vector, Set of category keywords (earn, corn, trade, acq, grain, interest, crude, wheat, ship, money).

**Output:** Set of document assigned to each category, Other: Text file.

**Procedure:**

```
1.  Initialize cat← empty, max←0
2.  Loop while not end of train file
3.    Read single document from the file
4.    Loop while not end of Document
5.      Read word from the document
6.      Check If word is in a category keyword And word weight >max Then
7.          Set cat ← word, Set max ← word weight
8.    End Loop
9.    Check If max ≠0 Then
10.       Put this document in the category with keyword cat
11.       Increase the number of document assigned to this category
12.       Set max←0
13.   Else
14.       Put the document to other file
15.       Increase other counter by 1
16. End Loop
```

## b. Filtering Document Using Similarity

After getting the category information from the training set document check the correctness of the learning document by finding the similarity between the document in the same category , this done by take each category C and calculate the cosine similarity as mentioned in the previous chapter for every document in that category with the remain document in the same category .

Since the similarity must be implemented between n*n vectors there is a problem arise when the two vectors of document not equal, then we take the union of the two document vector weight and give 0 value for the word weight if it does not appear the document, assume that the similarity of the first document with the second document give the value $\alpha_i$ the value of the similarity will be from $\alpha_1 \cdots\cdots\cdots \alpha_n$ where n is the total number of documents in the category C then

$$Similarity = \frac{\sum_{i=1}^{n} \alpha_i}{n} \tag{3.1}$$

Where if $Similarity \leq 0.1$ then the document is not appropriate to be in the category C move it to unlabeled document to examine it with other category and if $0.1 < Similarity \leq 1$ then the document is labeled correctly and stay in this category. Applying this operation to each category then examine the unlabeled document which is stored in other category also with each category to find an appropriate category to them if it found.

The value 0.1 chosen by trail and error process between 0 and 1, and the value 0.1 appears appropriate to chosen whether the document belong to the category if the similarity > 0.1 or not otherwise.

**<u>Algorithm (3.10) Document Filtering.</u>**

{The implemented step of Document Filtering}

**Input:** Set of k categories Text files, Array of number of document labeled to each category, Array of other document which is not labeled to any category.

**Output:** Array of document labeled to each category, Array of document not Labeled to any category, other Text file.

**Procedure:**

```
1. For i=1 To number of categories
2.    Loop while not end of file i
3.       Read single doc from the file
4. Call Algorithm (3.12) Cosine Similarity(Doc, the rest document in category i)
```

Continue

```
5.       Check If similarity result >= 0 And <= 0.1 Then
6.           Move doc to other file , Put i in the beginning of the document
7.           Decrease number of labeled document to this category
8.           Increase other counter
11.   End Loop
12. End For
13. Return (number of labeled document to each category, other counter).
```

Computing cosine similarity method for a given document in each category with a collection of other document in the given category is described in algorithm (3.11).

**Algorithm (3.11) Cosine Similarity.**

{ The implemented step of computing cosine similarity measure }

**Input:** document with it is weight as doc1, Set of k document with their weights.

**Output:** similarity.

**Procedure:**

```
1.  Initialize sum←0, norm1 ←0, norm2←0, total←0,similarity←0
2.  For i=1 To k
3.      Read document i from the file
4.      For each word in doc1
5.          Put word in word array and word weight in weigth1 and 0 in weight2
6.      End For
7.      For each word in document i
8.          Check If word in word array Then
9.              Put word weight in weight2
10.         Else
11.             Put word  in word array and word weight in weight2 and 0 in weight1
13.     Loop while there is a word in word array
14.         Calculate sum=sum+ weight1*weight2
15.         Calculate   norm1=norm1+ square(weight1)
16.         Calculate   norm2=norm2+ square(weight2)
17.     End Loop
```

Continue

```
18.   Calculate  norm1=square root(norm1),  norm2=square root(norm2)
19.   Calculate total=total + (sum/(norm1*norm2))
20.  Set sum,norm1,norm2←0,  Set word array,weight1,weight2←empty
21. End For
22. Calculate similarity= total/number of document
23. Return(similarity)
```

Before classifying testing set of documents, check the document in the other category to find out if there is any document in it could be classified correctly to any of the predefined category by applying similarity between each document in it and all document in each category and if the similarity value is greater than 0.1 then assign the document to the category with highest similarity value otherwise, keep it in the other category document.

**Algorithm (3.12) Classifying Other.**

{The implemented step of computing cosine similarity method for each document in other category with each of predefined category}

**Input:** set of k categories Text files, other: Text file.

**Output:** number of document labeled to each category.

**Procedure:**

```
1. Initialize max ← 0
2. Loop while not end of other file
3.   Read single doc from other file
4.   For i=1 To k
5.     Check If i is not in the doc Then   // if it not examine before in category i
6.       Call Algorithm (3.12) Cosine Similarity(Doc, all document in category i)
7.       Check If max < similarity Then
8.           Set max← similarity
9.           Set cat ← i
10.  End For
```

Continue

```
11. Check If max >0.1 Then
12.     Move doc to category cat
13.     Increment labeled document by 1
14.     Decrement other counter by 1
15.     Set max←0
16. End Loop
17. Return (labeled document to each category).
```

## 3.4.4 Document Classification Using Cosine Similarity

This is the final step in classification that is assign each of testing document to which category it will belong. Cosine Similarity measure is used to classify each document in the testing document and assign it to the correct category from the set of predefined category by applying the cosine similarity between each of the document in the testing with all document in each category and if the similarity value is greater than 0.1 then assign the document to the category with highest similarity value otherwise, move the document in the other category document.

**Algorithm (3.13) Document Classification.**

{The implemented step of document classification for testing document by applying cosine similarity}

**Input:** Set of k categories Text files, test: Text file.

**Output:** number of document labeled to each category.

**Procedure:**

```
1. Initialize max ← 0
2. Loop while not end of test file
3.   Read  single doc from other file
4.   For i=1 To k
5.     Call Algorithm (3.12) Cosine Similarity(Doct, all document in category i)
6.     Check If max < similarity Then
7.         Set max← similarity
8.         Set cat ← i
```

Continue

```
9.  End For
10. Check If max >0.1 Then
11.     Move doc to category cat
12.     Increment labeled document by 1
13.     Set max←0
14. End Loop
15. Return (labeled document to each category).
```

## 3.4.5 Evaluation of Classification Measurement

Text categorization systems classically have a parameter controlling their willingness to assign categories. Classifiers make binary decisions to assign categories that if it belongs or not for each category. Many measures have been used, including recall and precision, break-even point or F-measure as discussed in the previous chapter. Each of these measures is designed to evaluate some aspect of the text categorization performance of a system; however, none of them convey identical information.

The following algorithm calculates the performance measurement (Recall, Precision and F-measure) for each category in the classification system.

**Algorithm (3.14) Classification Measurement.**

{The implemented step of classification performance measurement}

```
Input: array of labeled document to each category, other counter: counter of document
       in other category, D: total number of document.
Output: recall, precision, f-score.
Procedure:
1. Initialize sumf, sump, sumr, f-score, precision, recall, sum ←0
2. Set a =D-other counter
3. For i=1 To k       //compute parameter b
4.     Set b(i)= number of labeled document to category i
5. End For
```

Continue

```
6.  For i=1 To k       //compute parameter c
7.      Set c(i) = number of labeled document to category i+ other counter
8.  End For
9.  For i=1 To k
10.     r(i)=a/(a+c(i))          // compute recall for each category
11.     sumr=sumr + r(i)          // compute sum of recall for all category
12. End For
13. For i=1 To k
14.     p(i)=a/(a+b(i))        // compute precision for each category
15.     sump=sump + p(i)        // compute sum of precision for all category
16. End For
17. For i=1 To k
18.     f(i)=(2 *p(i)*r(i))/(p(i)+r(i))        //compute f-score for each category
19.     sumf=sumf + f(i)                       //compute sum of f-score for each category
20. End For
21. f-score=sumf/k, recall=sumr/k, precision=sump/k     //total recall, precision, f-score
22. Return (f-score, recall, precision).
```

# Chapter Four

## Test & Results

# *Chapter Four*

# *Tests and Results*

## 4.1 <u>Introduction</u>

This chapter is dedicated to demonstrate the results of the tests which have been conducted to asses the performance of the established system. In the next section the step of the test procedures are illustrated.

Some samples of the preprocessing stages are shown. Finally, some tables of the processing stage results are given, the Reuters21578 newswire collection which is discussed in chapter two will be used to test the results of the proposed system.

## 4.2 <u>Experimental Results</u>

In this section some samples of the results of the preprocessing, Classification algorithm and evaluation of classification performance are listed in the following sub-sections.

### 4.2.1 <u>Preprocessing Phase Results</u>

The input is one selected HTML document form Reuter newswire articles to the preprocessing phase, the first step in the preprocessing phase is the conversion of an HTML document to the text file contains the document without tags, for example the original HTML document file size is 65 KB after removing tags it will be 64 KB.

The next step in this stage is to eliminate the special character and stop words from the document and convert all character to the small letter case, for the same example the number of word in the documents are 6632 after the elimination of the special character and stop words it was 5822 and the file size was 60 KB.

Finally performing the Porter stemming algorithm on the documents resulted from the previous step, for the same example the number of stemmed words in the documents was 2163 while the file size effected very little.

### 4.2.2 Document Classification Results

In document classifications the first step is splitting the documents into train and test randomly as used in this thesis, for example take a sample of 1000 documents and then partitioning them into 700 for training and 300 for testing according to the given algorithm as discussed in the previous chapter, the selection of the documents are randomly not ordered, run the program in 3 different times on the same taken sample and show the result after each run. Then take all the Reuters21578 collection of documents and also partitioning them according to the previous algorithm into 15104 for training and 6474 for testing, there are different binary classifiers to distinguish each topic in the collection of document. The TFIDF weighting classifier represents the stemmed document into vectors of weight for each document. Then category will be constructed depending on the weight vector gained from this classifier applied on training set of data and the number of document assigned to each category will be calculated.

Sample of document will assign to each category for the 1000 document after taking a single keyword by applying category construction algorithm for the 3 runs are listed in the table (4.1) below, while the assigned number of document to each category for the whole document collection after applying category construction algorithm are listed in table(4.2) below.

**Table (4.1) Assigned 700 Train Document in Category Construction for Sample of 1000 Document**

| *Category* | *Assigned Train Document* | | |
|---|---|---|---|
| | *Run1* | *Run2* | *Run3* |
| earn | 115 | 102 | 87 |
| acq | 35 | 50 | 45 |
| money | 61 | 65 | 40 |
| grain | 65 | 50 | 66 |
| crude | 81 | 100 | 30 |
| trade | 44 | 40 | 77 |
| interest | 70 | 65 | 40 |
| wheat | 72 | 80 | 71 |
| ship | 38 | 36 | 56 |
| corn | 30 | 32 | 25 |
| Other | 89 | 80 | 163 |

**Table (4.2) Assigned 15104 Train Document in Category Construction for Whole Reuters21578 Collection**

| *Category* | *Assigned Train Document* |
|---|---|
| earn | 1403 |
| acq | 1213 |
| money | 1087 |
| grain | 1344 |
| crude | 1849 |
| trade | 1743 |
| interest | 1801 |
| wheat | 2185 |
| ship | 850 |
| corn | 650 |
| Other | 979 |

When the category constructed using a single keyword the classification of the document will be applied by using cosine similarity first to filtering the category constructed apply cosine similarity to the train data then classifying the document in the other category to put them in appropriate category as discussed in the previous chapter, sample of the result of this process for 1000 document sample after applying 3 runs will be shown in table (4.3), while the assigned number of document to each category for the whole document collection after applying the above process to it are listed in table(4.4) below.

**Table (4.3) Assigned 700 Train Document after Apply Cosine Similarity to Filter Train Document and Classify Document in Other Category for Sample of 1000 Document.**

| *Category* | *Assigned Train Document* | | |
|---|---|---|---|
| | *Run1* | *Run2* | *Run3* |
| **earn** | **100** | **65** | **74** |
| **acq** | **30** | **43** | **52** |
| **money** | **75** | **69** | **36** |
| **grain** | **50** | **33** | **57** |
| **crude** | **79** | **81** | **34** |
| **trade** | **50** | **56** | **70** |
| **interest** | **80** | **76** | **95** |
| **wheat** | **65** | **33** | **33** |
| **ship** | **44** | **93** | **60** |
| **corn** | **27** | **66** | **76** |
| **Other** | **100** | **85** | **113** |

**Table (4.4) Assigned 15104 Train Document after Apply Cosine Similarity to Filter Train Document and Classify Document in Other Category for Whole Reuters21578 Collection.**

| *Category* | *Assigned Train Document* |
|---|---|
| **earn** | 1370 |
| **acq** | 1087 |
| **money** | 1100 |
| **grain** | 986 |
| **crude** | 1700 |
| **trade** | 1113 |
| **interest** | 1546 |
| **wheat** | 1743 |
| **ship** | 1310 |
| **corn** | 833 |
| **Other** | 2316 |

After classifying each document in the training set to which category it will belong, then take the document in the testing set and apply cosine similarity for each document with all category and put this document to the correct category if found where assigned number of document to each category is used to calculate the parameter of the classification performance measurement as shown in the next section, the result of this process which is discussed in the previous chapter for the sample of 1000 documents after applying 3 runs on the same sample will be listed in table (4.5) below, while the result of this process on the whole document collection from Reuters21578 will be listed in table (4.6) below.

**Table (4.5) Classified 300 Test Document for 1000 Document**

| Category | Assigned Test Document | | |
|---|---|---|---|
| | Run1 | Run2 | Run3 |
| earn | 55 | 37 | 46 |
| acq | 24 | 13 | 30 |
| money | 36 | 20 | 9 |
| grain | 30 | 37 | 23 |
| crude | 15 | 10 | 30 |
| trade | 23 | 33 | 27 |
| interest | 11 | 20 | 15 |
| wheat | 7 | 5 | 3 |
| ship | 43 | 40 | 47 |
| corn | 10 | 15 | 15 |
| Other | 46 | 70 | 55 |

**Table (4.6) Classified 6474 Test Document for Whole Document in Reuters21578 Collection.**

| Category | Assigned Test Document |
|---|---|
| earn | 843 |
| acq | 410 |
| money | 733 |
| grain | 397 |
| crude | 536 |
| trade | 313 |
| interest | 1000 |
| wheat | 871 |
| ship | 620 |
| corn | 231 |
| Other | 520 |

## 4.2.3 Classification Performance Results

In this thesis experimented performed on a subset of ten classes namely: *acq, earn, corn, crude, trade, grain, interest, money, ship and wheat.* The precision, recall and F-score measure over each class used as a representation for the performance, these measures needs a parameter as discussed in chapter 2 which is a and it will be the whole document in the problem except the document in the other category and it is value for the sample of 1000 document is 1000-146=854 for the first run, 1000-155=845 for the second run and 1000-168=832 for the third run, while it is value for the whole Reuters21578 document collection is 21578-2836=18742 all these value is obtained from taking the document number from the problem minus the number of document in other category for both train and test document.

Table (4.7) show the value of b and c for each class for the sample of 1000 word for the first, second and third run respectively. While table (4.8) show the value of b and c for the whole document collection in Reuters21578.

**Table (4.7) b and c Value for each Class in 1000 Document sample**

| ***Category*** | ***earn*** | ***acq*** | ***money*** | ***grain*** | ***crude*** | ***trade*** | ***interest*** | ***wheat*** | ***ship*** | ***corn*** |
|---|---|---|---|---|---|---|---|---|---|---|
| ***Run1*** | | | | | | | | | | |
| ***b*** | 155 | 54 | 111 | 80 | 94 | 73 | 91 | 72 | 87 | 37 |
| ***c*** | 301 | 200 | 257 | 226 | 240 | 219 | 237 | 218 | 233 | 183 |
| ***Run2*** | | | | | | | | | | |
| ***b*** | 102 | 56 | 89 | 70 | 91 | 89 | 96 | 38 | 133 | 81 |
| ***c*** | 257 | 211 | 244 | 225 | 246 | 244 | 251 | 193 | 288 | 236 |
| ***Run3*** | | | | | | | | | | |
| ***b*** | 120 | 82 | 45 | 80 | 64 | 97 | 110 | 36 | 107 | 91 |
| ***c*** | 288 | 250 | 213 | 248 | 232 | 265 | 278 | 204 | 275 | 259 |

**Table (4.8) b and c Value for each Class in whole Reuters21578 Document Collection.**

| *Category* | *earn* | *acq* | *money* | *grain* | *crude* | *trade* | *interest* | *wheat* | *ship* | *corn* |
|---|---|---|---|---|---|---|---|---|---|---|
| *b* | 2213 | 1497 | 1833 | 1383 | 2236 | 1426 | 2546 | 2614 | 1930 | 1064 |
| *c* | 5049 | 4333 | 4669 | 4219 | 5072 | 4262 | 5382 | 5450 | 4766 | 3900 |

Where the value of b is calculated by the total number of document assigned to the given class in both train and test set and the value of c is the total number of document assigned to a given class added to the number of document in other category.

Table (4.9), table (4.10) and table (4.11) shows the value of precision, recall and F-measure for the sample of 1000 document in first, second and third run respectively, while table (4.12) shows the value of precision, recall and F-measure for the whole Reuters21578 document collection.

**Table (4.9) Classification Performance of 1000 Document in Run1**

| *Category* | *Precision* | *Recall* | *F-measure* |
|---|---|---|---|
| *earn* | 0.84 | 0.73 | 0.78 |
| *acq* | 0.94 | 0.81 | 0.87 |
| *money* | 0.88 | 0.76 | 0.82 |
| *grain* | 0.91 | 0.79 | 0.84 |
| *crude* | 0.90 | 0.78 | 0.83 |
| *trade* | 0.92 | 0.79 | 0.85 |
| *interest* | 0.90 | 0.78 | 0.83 |
| *wheat* | 0.92 | 0.79 | 0.85 |
| *ship* | 0.90 | 0.78 | 0.84 |
| *corn* | 0.95 | 0.82 | 0.88 |
| *total* | 0.91 | 0.78 | 0.84 |

**Table (4.10) Classification Performance of 1000 Document in Run2**

| *Category* | *Precision* | *Recall* | *F-measure* |
|---|---|---|---|
| ***earn*** | 0.89 | 0.76 | 0.82 |
| ***acq*** | 0.93 | 0.80 | 0.86 |
| ***money*** | 0.90 | 0.77 | 0.83 |
| ***grain*** | 0.92 | 0.79 | 0.85 |
| ***crude*** | 0.90 | 0.77 | 0.83 |
| ***trade*** | 0.90 | 0.77 | 0.83 |
| ***interest*** | 0.89 | 0.77 | 0.83 |
| ***wheat*** | 0.95 | 0.81 | 0.88 |
| ***ship*** | 0.86 | 0.74 | 0.80 |
| ***corn*** | 0.91 | 0.78 | 0.84 |
| ***total*** | 0.90 | 0.77 | 0.83 |

**Table (4.11) Classification Performance of 1000 Document in Run3**

| *Category* | *Precision* | *Recall* | *F-measure* |
|---|---|---|---|
| ***earn*** | 0.87 | 0.74 | 0.80 |
| ***acq*** | 0.91 | 0.76 | 0.83 |
| ***money*** | 0.94 | 0.79 | 0.86 |
| ***grain*** | 0.91 | 0.77 | 0.83 |
| ***crude*** | 0.92 | 0.78 | 0.84 |
| ***trade*** | 0.89 | 0.75 | 0.82 |
| ***interest*** | 0.88 | 0.75 | 0.81 |
| ***wheat*** | 0.95 | 0.80 | 0.87 |
| ***ship*** | 0.88 | 0.75 | 0.81 |
| ***corn*** | 0.90 | 0.76 | 0.82 |
| ***total*** | 0.90 | 0.76 | 0.83 |

**Table (4.12) Classification Performance of Reuters21578 Document Collection**

| *Category* | *Precision* | *Recall* | *F-measure* |
|---|---|---|---|
| ***earn*** | 0.89 | 0.78 | 0.83 |
| ***acq*** | 0.92 | 0.81 | 0.86 |
| ***money*** | 0.91 | 0.80 | 0.85 |
| ***grain*** | 0.93 | 0.81 | 0.87 |
| ***crude*** | 0.89 | 0.87 | 0.83 |
| ***trade*** | 0.92 | 0.81 | 0.86 |
| ***interest*** | 0.88 | 0.77 | 0.82 |
| ***wheat*** | 0.87 | 0.77 | 0.82 |
| ***ship*** | 0.90 | 0.79 | 0.84 |
| ***corn*** | 0.94 | 0.82 | 0.88 |
| ***total*** | 0.90 | 0.79 | 0.85 |

## 4.3 Related Results and Discussions

In this section, we briefly overview some results which are most relevant to the present one. Table (4.13) contains the authors and the dataset they used to test their results and the type of classifier used to classify the data and their corresponding performance represented by one of the classification measures used.

Performance of the classification method used in this thesis is measured as shown in the previous section where the difference between the precision and recall values is small and the value of F-measure is about %85. As compared to the others listed in table (4.13), our classification can be considered as good and shows that the breakeven point can be reached and these lead to more system quality and strong in classifying each document to its relevant category. After applying many runs to the document collection, the changing of precision value is very small and almost gives high value so system quality is high and the same is done for recall value but it is value is less than precision so the accuracy of the system to find relevant documents is good as discussed in chapter two.

**Table (4.13) Summary of Related Results**

| **Authors** | **Dataset** | **Classifier** | **Measurement Results** |
|---|---|---|---|
| **Joa98** | Reuters | SVM | %86 break even point |
| **Wei٠٤** | Reuters | Decision Tree | %87 break even point |
| **Yan99** | Reuters | SVM | %86 F-measure |
| **Bak98** | 20NG | Naïve Bayes | %85 accuracy |
| **Nig98** | WebKB | Naïve Bayes | %82 accuracy |

# Chapter Five

# Conclusion & Future Work

# *Chapter Five*

# *Conclusion and Future Work*

## 5.1 Conclusions

According to the tests results presented in chapter four, the following conclusions have been derived:

1. TFIDF weighting method with cosine similarity has very good performance in practice. But its drawback is the computational effort during classification, where basically the similarity of a document with respect to all other documents of a training set has to be determined.
2. Intuitively, one expects that the more information that is available, the better one should do. The more knowledge we have, the better we can make decisions. Similarly, the more documents available to be classified, the better the used classification method work and more powerful.
3. When using binary classification the decision taken to assign each document to the correct category will be much better and more accurate to classification, since the document goes to the more similar class while in multi class categorization the document will go to more than one similar class.
4. Extracting structured information from text documents often requires sophisticated and computationally intensive techniques. Processing every available document is not feasible for the web or for large searchable newswire document collection as Reuters21578. It is also highly inefficient, since only a small fraction of documents in a collection are often useful for most information extraction tasks.

## 5.2 Future Work

1. Using another type of data to be processed and classified such as email messages.
2. Using another kind of text mining techniques such as clustering to extract topics in the document.
3. Using an HTML document in Arabic and classifying them according to their topics and do not discard the tags but make some evaluation to increase the weight of the used term (separate word, keyword, and longer phrase).

# *References*

## References

- **[Agi05]** Y. Agichtein, *"Extraction relations from Large Text collections"*, PhD Thesis, Columbia university/ USA, 2005.
- **[Ali06]** N. B. Ali,*" Classes Mining using Artificial Neural Network"*, M.Sc. Thesis, Dep. Computer science, Iraqi Commission for Computers and Information/ Iraq, 2006.
- **[Apt94]** C. Apte, F. Damerau, and S. Weiss, *"Automated Learning of Decision Rules for Text Categorization"***,** ACM Trans. Information Systems, Vol. 12, No. 3, pp. 233–251, 1994.
- **[Bak98]** D. Baker and A. K. McCallum, *"Distributional clustering of words for text classification"*, ACM International Conference. Information Retrieval, pp. 96-103, 1998.
- **[Bas03]** A. Basu, C. Watters and M. Shepherd, *"Support Vector Machines for Text Categorization"*, IEEE Intelligent System, 2003.
- **[Beu07]** R. C. Bunescu**,** *" Learning for Information Extraction: From Named Entity Recognition and Disambiguation to Relation Extraction"***,** Ph.D. Thesis, Texas University, 2007.
- **[Bil06]** M. Bilenko, *"Learnable Similarity Functions and Their Application to Record Linkage and Clustering"***,** Ph.D. Thesis, Dep. Computer sciences, Texas University, 2006.
- **[Cla03]** E. C. and R. J. Mooney, *" Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction"*, Mary Journal of Machine Learning Research, vol.4, pp. 177-210, 2003.

## References

- **[Cla98]** M. E. Califf, *"Relational Learning Techniques for Natural Language Information Extraction"*, Ph.D. Thesis, Dep. of Computer sciences, Texas University /USA 1998.

- **[Cor03]** M. W. Corney, *"Analyzing E-mail Text authorship for Forensic purposes"*, M.Sc. Thesis, Queensland university of Technology/ Australia, 2003.

- **[Daw74]** J.L. DAWSON, *"Suffix Removal and Word Conflation"*, ALLC Bulletin, pp.33-46, 1974.

- **[Hea00]** M. Hearst, *"Text Mining: Instruments for Scientific Discovery"*, IMA Text Mining workshop, 2000.

- **[Hea98]** M. A. Hearst, *"Support Vector Machine"*, California University/USA, IEEE Intelligent System, pp.18-28, 1998.

- **[Hei04]** K. E. Heinrich, *"Finding Functional Gene Relationships Using the Semantic Gene Organizer"*, M.Sc. thesis, Tennessee University/USA, 2004.

- **[Joa98]** Joachims, *"Text categorization with support vector machines: learning with many relevant features"*, Lecture Notes in Computer Science series, pp 137-142, no. 1398, 1998.

- **[Kam02]** N. Kamolvilassatian, *"Property-Based Feature Engineering and Selection"*, M.Sc. Thesis, Dep. Computer sciences, Texas University/USA, 2002.

- **[Lid00]** E. Liddy, *"Text Mining"*, the Center for Natural Language Processing in the School of Information Studies, Syracuse

University/USA, American society of information Science, vol.27 No.1,2000.

- **[Lin02]** X. Lin, *"Text-mining based journal splitting"*, Intelligent Enterprise Technology Laboratory, HP Laboratories Palo Alto, 2002.

- **[Liu03]** T. Y. Liu, Y. Yang, H. Wan, H. J. Zeng, Z. Chen and W. Y. Ma**,** *"Support Vector Machines Classification with A Very Large-scale Taxonomy"***,** SIGKDD exploration, vol. 7,pp 36-43,2003.

- **[Lov68]** J.B. LOVINS, *"Development of a Stemming Algorithm"*, Mechanical Translation and computation Linguistics. Vol. 11, pp. 23-31, 1968.

- **[Mcs04]** F. Mcsherry, *"Spectral Methods for Data analysis"*, PhD Thesis, Dep. Computer science & engineering, Washington University/ USA, 2003.

- **[Nig98]** A. Nigam, K. McCallum, S. Thrun and T. M. Mitchell, *"Learning to classify text from labeled and unlabeled documents"*, American Association for Artificial Intelligence, pp. 792-799, 1998.

- **[Por00**] M.F Porter, *"An algorithm for suffix stripping"***,** Program, vol.14, pp. 130-137, 2000.

- **[Moo05]** R. J. Mooney, and R. Bunescu, *"Mining Knowledge from Text Using Information Extraction"*, SIGKDD exploration, vol.7, pp. 3-10, 2005.

- **[Nah04]** U. Y. Nahm, "*Text Mining with Information Extraction"*, Ph.D. Thesis, Dep. Computer sciences, Texas University /USA, 2004.

**References**

- **[Nso06]** G. C. Nsofor, *"A comparative Analysis of Predictive Data-Mining Techniques"*, M.Sc. Thesis, Tennessee University/ USA, 2006.

- **[Ozg04]** A. Ozgur,*" Supervised and Unsupervised Machine Learning Techniques or Text Document Categorization "*, M.Sc. thesis, Dep. Computer engineering, Bogazici University/ Turkey, 2004.

- **[Pen03]** F. Peng, *"Language Independent Text Learning with Statistical n-Gram Language Models"*, PhD Thesis, Dep. Computer science, Waterloo University/ Canada, 2003.

- **[Pia94]** C.D. Paice, *"An evaluation method for stemming algorithms"*, ACM-SIGIR94, pp. 42–50, 1994.

- **[Pot05]** E. M. Pothos and N. Chater, *"Unsupervised Categorization and Category Learning"*, the Quarterly Journal of Experimental Psychology, 2005.

- **[Rad06]** A. K. M. Radhi, *"Machine Learning for Text Categorization"*, PhD Thesis, Dep. Computer science, Technology University/ Iraq, 2006.

- **[Sal02]** H. Saleep, *"A framework for recommending Text-Based classification algorithms "*, PhD Thesis, Computer Science School and Information Systems, London University, 2002.

- **[Tai02]** H. Taira, *"Text Categorization using Machine Learning"*, PhD Thesis, Dep. of Information Processing, Msgill University/ France, 2002.

- **[Tak03]** H. Takamura, *"Clustering Approach to Text Categorization"*, PhD Thesis, Comenius University, Slovakia, 2003.

## References

- **[Tho99]** C. A. Thompson, M. E. Califf, R. J. Mooney, *" Active Learning for Natural Language Parsing and Information Extraction "*, International Machine Learning Conference, pp.406-414, Slovenia, 1999.

- **[Tka98]** D. S. Tkach, *"Information Mining with the IBM Intelligent Miner Family "*, IBM Software Solution white paper, International Business Machines Corporation, 1998.

- **[Tka98]** D. Tkach, *"Text Mining Technology Turning Information into Knowledge "*, IBM software solution, 1998.

- **[Tur01]** P. Turney, *"Mining the Web for Synchronous"*, National Research council/ Canada, Conference on Machine Learning (ECML), Freiburg, Germany, pp. 491–502, NRC 44893,2001.

- **[Wei04]** S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles and T. Hampp, *" Maximizing Text Mining Performance "*, Watson Research Center, IEEE Intelligent Systems, pp. 63-69, 2004.

- **[Xin04]** Z. Xinhua, *"Building Maximum Entropy Text Classifier Using Semi-supervised Learning"*, PhD Thesis, Computing National School, Singapore University/ Singapore, 2004.

- **[Yan99]** Y. Yang, *"An Evaluation of Statistical Approaches to Text Categorization "*, Kluwer Academic Publishers. Manufactured in The Netherlands, Vol. 1, pp. 69–90, 1999.

- **[Zha05]** Z. Zhang, *"Tapping into the power of Text Mining"*, Communication of ACM, 2005.

# Appendix A

# Appendix A

# An English Stop word List

An English stop word list. Many of the forms below are quite rare (e.g. "yourselves") but included for completeness.

## Pronouns Forms

1st person single is shown in Table A.1

**Table A.1 1st Person Single**

| | | | |
|---|---|---|---|
| i | me | my | myself |

1st person plural is shown in Table A.2

**Table A.2 1st Person Plural**

| | | | | |
|---|---|---|---|---|
| we | us | our | ours | ourselves |

Second person Is shown in Table A.3

**Table A.3 Second Person**

| | | | | |
|---|---|---|---|---|
| you | your | yours | yourself | yourselves |

Third person singular is shown in Table A.4

**Table A.4 Third Person Singular**

| | | | | | |
|---|---|---|---|---|---|
| he | him | his | himself | she | her |
| hers | herself | it | its | itself | |

Third person plural is shown in Table A.5

**Table A.5 Third Person Plural**

| | | | | |
|---|---|---|---|---|
| they | them | their | theirs | themselves |

## Other Forms

Is shown in Table A.6

**Table A.6 Other Forms**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| what | which | who | whom | this | that | these | those |

## Verb Forms

Is shown in Table A.7

**Table A.7 Verb Forms**

| | | | | | | |
|---|---|---|---|---|---|---|
| will | can | may | must | might | should | shall |
| be | am | is | are | was | ought | were |
| been | being | have | has | had | could | having |
| do | does | did | doing | would | | |

## Compound Forms

Increasingly encountered nowadays in formal writing

1- Pronoun + verb are shown in Table A.8.

**Table A.8 Compound Forms (pronouns + verb)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| i'm | you're | he's | she's | it's | we've | we're | they're |
| i've | you've | they've | i'd | you'd | he'd | she'd | we'd |
| they'd | i'll | you'll | he'll | she'll | we'll | they'll | |

2- Verb + negation are shown in Table A.9.

**Table A.9 Compound Forms (verb + negation)**

| | | | | |
|---|---|---|---|---|
| isn't | aren't | wasn't | weren't | hasn't |
| haven't | hadn't | doesn't | don't | didn't |

3- Auxiliary + negation are shown in Table A.10

**Table A.10 Compound Forms (auxiliary + negation)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| won't | wouldn't | shan't | shouldn't | can't | cannot | couldn't | mustn't |

## Miscellaneous Forms

Is shown in Table A.11.

**Table A.11 Miscellaneous Forms**

| | | | | |
|---|---|---|---|---|
| let's | that's | who's | what's | here's |
| there's | when's | where's | why's | how's |

## The Rest

Overlap among prepositions, conjunctions, adverbs, etc. is shown in Table A.12.

**Table A.12 the Rest**

| | | | | | | |
|---|---|---|---|---|---|---|
| a | an | the | and | but | if | or |
| because | as | until | while | of | at | by |
| for | with | about | against | between | into | through |
| during | before | after | above | below | to | form |
| up | down | in | out | on | off | over |
| under | again | further | then | once | here | there |
| when | where | why | how | all | any | both |
| each | few | more | most | other | some | such |
| no | nor | not | only | own | same | so |
| than | too | very | | | | |

## Commonest in English

Just for record, the following words are among the commonest in English. Is shown in Table A.13.

**Table A.13 Commonest in English**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| one | every | least | less | many | now | ever | never | say | says |
| said | also | get | go | goes | just | made | make | put | see |
| seen | whether | like | well | back | even | still | way | take | since |
| long | high | old | new | second | first | five | four | three | two |

.

# Appendix B

# Porter Stemming Rules

Before applying the rule on the word here are some definition must be applied:

- Define a *vowel* as one of ***a e i o u y***
- Define a *double* as one of ***bb dd ff gg mm nn pp rr tt***
- Define a *valid* **li**-*ending* as one of ***c d e g h k m n r t***
- *R*1 is the region after the first non-vowel following a vowel, or the end of the word if there is no such non-vowel.
- *R*2 is the region after the first non-vowel following a vowel in *R*1, or the end of the word if there is no such non-vowel.
- Define a *short syllable* in a word as either (*a*) a vowel followed by a non-vowel other than ***w***, ***x*** or ***Y*** and preceded by a non-vowel, or (*b*) a vowel at the beginning of the word followed by a non-vowel. So *rap*, *trap*, *entrap* end with a short syllable, and *ow*, *on*, *at* are classed as short syllables. But *uproot*, *bestow*, *disturb* do not end with a short syllable. A word is called *short* if it ends in a short syllable, and if *R*1 is null. So *bed*, *shed* and *shred* are short words, *bead*, *embed*, *beds* are not short words.

Step 0:

Search for the longest among the suffixes, ***' , 's , 's'*** and remove if found.

Step 1*a*:

Search for the longest among the following suffixes, and perform the action indicated.

***sses*** replace by ***ss***

***ied***, ***ies*** replace by ***i*** if preceded by more than one letter, otherwise by ***ie*** (so *ties* -> *tie, cries* -> *cri*)

***s*** delete if the preceding word part contains a vowel not immediately before the ***s*** (so *gas* and *this* retain the ***s***, *gaps* and *kiwis* lose it)

***us ss*** do nothing

Step 1*b*:

Search for the longest among the following suffixes, and perform the action indicated.

***eed, eedly*** replace by ***ee*** if in *R*1

***ed, edly*, *ing, ingly*** delete if the preceding word part contains a vowel, and then if the word ends ***at***, ***bl*** or ***iz*** add ***e*** (so *luxuriat* -> *luxuriate*), or if the word ends with a double remove the last letter (so *hopp* -> *hop*), or if the word is short, add ***e*** (so *hop* -> *hope*)

Step 1*c*:

Replace suffix ***y*** or ***Y*** by ***i*** if preceded by a non-vowel which is not the first letter of the word (so *cry* -> *cri*, *by* -> *by*, *say* -> *say*)

Step 2:

Search for the longest among the following suffixes, and, if found and in *R*1, perform the action indicated.

***tional***: replace by ***tion***

***enci***: replace by ***ence***

***anci***: replace by ***ance***

***abli***: replace by ***able***

***entli***: replace by ***ent***

***izer ization***: replace by ***ize***

***ational ation ator***: replace by ***ate***

***alism aliti alli***: replace by ***al***

***fulness***: replace by ***ful***

***ousli ousness***: replace by ***ous***

***iveness iviti***: replace by ***ive***

***biliti bli***: replace by ***ble***

***ogi***: replace by ***og*** if preceded by ***l***

***fulli***: replace by ***ful***

***lessli***: replace by ***less***

***li***: delete if preceded by a valid ***li***-ending

Step 3:

Search for the longest among the following suffixes, and, if found and in *R*1, perform the action indicated.

***tional***: replace by ***tion***

***ational***: replace by ***ate***

***alize***: replace by ***al***

***icate iciti ical***: replace by ***ic***

***ful ness***: delete

***ative***: delete if in *R*2

Step 4:

Search for the longest among the following suffixes, and, if found and in *R*2, perform the action indicated.

***al ance ence er ic able ible ant ement ment ent ism ate iti ous ive ize*** delete

***ion*** delete if preceded by ***s*** or ***t***

Step 5:

Search for the following suffixes, and, if found, perform the action indicated.

***e*** delete if in *R*2, or in *R*1 and not preceded by a short syllable

***l*** delete if in *R*2 and preceded by ***l***

Finally, turn any remaining ***Y*** letters in the word back into lower case.

# الخلاصة

تعدين النص اصبح قضية بحث مهمة منذ انفجار معلومات النص الرقمية على الانترنيت، بينما البيانات قد تخزن كوثائق الكترونية او كأجزاء بنيوية في قواعد البيانات، ممازاد في أهمية تعدين النص وقيمته الاقتصادية. احدى الاهداف المهمة في تعدين النص هو التصنيف الالي للوثائق الالكترونية. برامج الحاسوب تمسح النص في أي وثيقة وتطبق نموذج الذي يخصص الوثيقة الى واحد أو أكثر من الأصناف المعرفة (المواضيع).

الخطوة الاولى تتم بمعالجة الوثيقة من خلال تنفيذ بعض طرق معالجة اللغة الطبيعية عليها لازالة البيانات الغير ضرورية منها والتي لاتحمل اي قيمة معرفية وتزيل التشويش من خلال وضع الكلمات التي بنفس المعنى بكلمة واحدة لعمل المقارنة بين الوثائق بسهولة أكثر.

الخطوة الثانية المستعملة في هذه الأطروحة هي تصنيف الوثيقة. في هذه الخطوة مجموعة الوثائق تقسم الى مجموعات ثانوية للتدريب والاختبار،وكلامن وثائق التدريب والاختبار تستعمل لتعليم الماكنة. تستعمل طريقة متجه نموذج الفضاء لتردد التعبير وتردد الوثيقة المعكوس مع عامل التشابة، في تصنيف كلا من وثائق التدريب واختبار الوثيقة. اخيرا، يقاس أداء طريقة التصنيف المستخدمة.

مجموعة الوثائق المخصصة ( رويتر ٢١٥٧٨ ) أستعملت لأختبار طرق التصنيف و لقياس أداء طريقة التصنيف المقترحة في هذه الاطروحة. ان النتائج المكتسبة نتائج مشجعة.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

# تعدين النص استنادا للمحتوى

رسالة

مقدمة الى كلية العلوم في جامعة النهرين كجزء من متطلبات نيل شهادة درجة الماجستير في علوم الحاسبات

من قبل
**زهراء راجي محي الزبيدي**
(بكالوريوس ٢٠٠٥ )

المشرف

د. طه سعدون باشاغا

كانون الثاني ٢٠٠٩ محرم ١٤٣٠