

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

1-1-2006

### A semantic partition based text mining model for document classification.

Catherine Inibhunu  
*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

#### Recommended Citation

Inibhunu, Catherine, "A semantic partition based text mining model for document classification." (2006). *Electronic Theses and Dissertations*. 7101.  
<https://scholar.uwindsor.ca/etd/7101>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

# **A Semantic Partition Based Text Mining Model for Document Classification**

by

Catherine Inibhunu

A Thesis

Submitted to the Faculty of Graduate Studies and Research through  
Computer Science in Partial Fulfillment of the Requirements  
for the Degree of Master of Science at the  
University of Windsor

Windsor, Ontario, Canada

2006

©2006 Catherine Inibhunu



Library and  
Archives Canada

Bibliothèque et  
Archives Canada

Published Heritage  
Branch

Direction du  
Patrimoine de l'édition

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*  
*ISBN: 978-0-494-35962-4*  
*Our file* *Notre référence*  
*ISBN: 978-0-494-35962-4*

**NOTICE:**

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

**AVIS:**

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

  
**Canada**

## **Abstract**

Feature Extraction is a mechanism used to extract key phrases from any given text documents. This extraction can be weighted, ranked or semantic based. Weighted and Ranking based feature extraction normally assigns scores to extracted words based on various heuristics. Highest scoring words are seen as important. Semantic based extractions normally try to understand word meanings, and words with higher orientation based on a document context are picked as key features. Weighted and Ranking based feature extraction approaches are used for creating document summaries that can act as their representations in the absence of the original documents. However, these two approaches suffer from some major drawbacks: (1) summaries generated could contain words that seem irrelevant to the document context, (2) sentences containing some key words could be eliminated if ranked lower than a given threshold, (3) summaries must be processed further in order to serve as input for mining algorithms like the Apriori.

This thesis proposes Semantic Partitions (SEM-P) and Enhanced Semantic Partitions (ESEM-P) algorithms based on the semantic orientation of words in a document. This partitioning reduces the amount of words required to represent each document as input for discovering word frequent patterns from a collection of documents, while still maintaining the semantics of the documents. A weighting and ranking heuristic measure for each word term in a partition is used in ESEM-P to prune low ranked terms resulting in improved performance of the ESEM-P over the SEM-P. Identified word frequent patterns are used to generate a document classification model.

Keywords: Text mining, text information mining, unstructured data mining, feature extraction, semantic orientation, text classification, semantic partitions, text summarization.

## **Acknowledgement**

I would like to give thanks to my Supervisor Dr. Christie Ezeife for her guidance through out my entire studies at the University of Windsor both Undergraduate and Graduate studies. Dr. Ezeife gave me advice on numerous issues, including the courses to take, she got me interested in research and gave me an excellent research direction. Dr. Ezeife has carefully reviewed all my reports and given me suggestions on how to make improvements. I truly admire her dedication to teaching and care for students. She always wants the best from her students and is always available to lend any help with or without appointments. Thank you Dr. Ezeife for being such a good person and professor, totally dedicated to students welfare and academic success.

I would like to thank my entire thesis committee members. I thank my external reader, Dr. Zhiguo Hu for teaching me Mathematics, thanks to my internal reader, Dr. Arunita Jaekel for accepting to read my bulky thesis within a very short notice due to replacement. I also thank the thesis chair, Dr. Jessica Chen, for teaching me Software Engineering and being part of my thesis committee. Special thanks to the entire committee for taking their valuable time to review my thesis and provide valuable suggestions.

I am infinitely grateful to my family, my wonderful husband and our lovely children who have given me unconditional love and care through out my studies. I also give special thanks to my parents for installing excellence as a virtue early in my childhood. I finally want to thank the staff at the School of Computer Science and members of the WODD lab for any help they have given to me. Thank you all.

# Table of Contents

Abstract .....	iii
Acknowledgement.....	iv
Table of Contents .....	v
List of Tables.....	vii
List of Figures .....	ix
Chapter 1: INTRODUCTION.....	1
1.2 Text Information Mining .....	2
1.2.1 Information Retrieval .....	2
1.2.2 Text Extraction.....	3
1.2.3 Text Summarization .....	4
1.2.4 Text Classification .....	5
1.2.5 Text Clustering.....	7
1.2.6 Association Rules.....	10
1.3 The Motivation of Thesis.....	12
1.4 The Thesis Contribution .....	15
1.5 Outline of the Thesis.....	16
Chapter 2: RELATED WORKS.....	17
2.1 Feature Extraction Algorithms.....	17
2.1.1 Cue Markers .....	17
2.1.2 ClearStudio.....	20
2.1.3 Rapier and DiscoTex Methods.....	21
2.1.4 Multi-Strategy Approach.....	23
2.2 Ranking and Weighting Mechanisms .....	25
2.2.1 Summarization with Relevance Measure .....	25
2.2.1.1 Relevance Measure.....	26
2.2.1.2 Singular Value Decomposition.....	27
2.2.2 Summarization as Feature Selection for Text Categorization.....	29
2.2.3 Sentence-Selection Heuristic.....	30
2.3 Lexical Analysis .....	33
2.3.1 Efficient Text Summarization with Lexical Chains.....	33
2.3.2 Text Summarization with Lexical Chains.....	35
2.3.3 Semantic Orientation.....	36

2.4 Association Rule Approach .....	37
2.4.1 Multi-pass Apriori (M-Apriori) and Multi-pass-Direct Hashing and Pruning (M-DHP). .....	38
2.4.2 Associating Terms with Text Categories .....	42
2.4.3 Discovering Technological Intelligence.....	44
Chapter 3: PROPOSED ALGORITHMS FOR TEXT INFORMATION MINING .....	45
3.1 Description of Semantic Partition (SEM-P) .....	49
3.1.1 Semantic-Partition Algorithm (SEM-P).....	49
3.2.1 Enhanced Semantic-Partitions Algorithm (ESEM-P).....	69
Chapter 4: PERFORMANCE ANALYSIS .....	70
4.1 Implementation Environments.....	70
4.2 Performance Measures with Existing Text Collections.....	70
4.3 Experiments on Memory Usage and Text Preprocessing Time.....	72
4.4 Experiments on Building Classification Models .....	74
4.5 Analysis of Experimental Results.....	75
4.5.1 Suggestions to Improve the Preprocessing Time of Both the SEMP and ESEM_P .....	76
Chapter 5: CONCLUSIONS AND FUTURE WORK.....	77
Bibliography .....	79
Vita Auctoris .....	84

## List of Tables

Table 1.1: Highly occurring Terms in Figure 1.2.2 and Figure 1.2.5.....	8
Table 1.1.2: Matrix Representation of Sentences and Clusters $C_1$ and $C_2$ . .....	9
Table 1.1.3: The first Distance Calculation ( $D_0$ ) Generating ( $C_0$ ).....	9
Table 1.1.4: The Second Distance Calculation ( $D_1$ ) Generating ( $C_1$ ).....	10
Table 1.2: A Structured Database with Unique Transaction Ids. ....	10
Table 1.3: Transaction representations of text segments in Fig 1.2.2 and Fig 1.2.5 .....	11
Table 1.4.1: Itemsets generating $C_1$ .....	12
Table 1.4.2: Itemsets with Minimum Support.....	12
Table 1.4.3: $C_1$ generated from $L_1$ .....	12
Table 1.4.4: Generating $L_2$ from $C_2$ .....	12
Table 2.1.1: Vocabulary list of terms from Scientific Paper in Fig 2.1.4.....	24
Table 2.1.2: A Frequency Count of Terms from $G_1$ and $G_2$ from Table 2.1.1 .....	24
Table 2.2.1: Individual Text Sentences .....	27
Table 2.2.1.2: A 2 x 4 Sample Matrix Representation (Matrix A).....	28
Table 2.2.1.3: Transpose of Matrix A .....	28
Table 2.2.2: Breakdown of Text Segment in Figure 2.2.1 .....	31
Table 2.2.3: Term Representation in Table 2.2.2 .....	31
Table 2.4.1: A Sample Transaction Data Set.....	39
Table 2.4.2: Candidate 1 Itemset, $C_1$ .....	39
Table 2.4.3: Frequent 1 Itemset, $L_1$ .....	39
Table 2.4.4: Candidate 2 Itemset, $C_2$ .....	40
Table 2.4.5: Frequent 2 Itemsets, $L_1$ .....	40
Table 2.5: A HashTable Representation of Candidate 2 Itemsets.....	41
Table 3.1.1: A Ranking of Semantic Partitions from Figure 3.1.4.....	56
Table 3.1.2: Groups Formed from Ranked Semantic Partitions.....	57
Table 3.1.3: Candidate 1 itemset, $C_1$ .....	61
Table 3.1.4 Frequent 1 itemset, $L_1$ .....	61
Table 3.1.5: Candidate 2 itemset, $C_2$ .....	62
Table 3.1.6: Frequent 2 Itemset, $L_2$ in $C_2$ .....	62



Table 3.1.7: Candidate 3 Itemset, $C_3$ .....	62
Table 3.1.8: A HashMap Binary Identifier for the Concept Hierarchy in Figure 3.1.4.2.....	65
Table 3.1.9: The Identified Semantic Partitions for Document D6.....	66
Table 3.1.10: An Updated HashMap after Adding Document D6.....	67
Table 3.1.11: A result of Pruning the Contents of Figure 3.1.5 .....	68
Table 4.3.1: Amount of Disc Space Before and After Processing Text.....	72
Table 4.3.2: Execution Time to PreProcess Text Documents .....	72
Table 4.4.1: Execution Time in Seconds for Building Classification Models .....	74

## List of Figures

Figure 1.2.1: An Information Retrieval Procedure.....	3
Figure 1.2.2: Sample Paragraph in a Text Document.....	4
Figure 1.2.2.1: Final Extracted Words from Text Document in Figure 1.2.2 .....	4
Figure 1.2.3: Final summary of Text Segment in Figure 1.2.2. ....	5
Figure 1.2.4: A Sample Term Matrix.....	7
Figure 1.2.5: A Sample Text Document to be Clustered.....	8
Figure 2.1.1: A Text Segment Adapted from (Chuang and Yang, 2000).....	18
Figure 2.1.2: Decomposition of Sentences in Figure 2.1.1 into Individual Segments .....	19
Figure 2.1.3: Sample Extraction Rule in Rapier Adopted from (Califf and Mooney, 2003). ....	22
Figure 2.1.4: A Scientific paper Adapted from (Castillo and Serrano, 2004). ....	23
Figure 2.2.1: A Paragraph of a Text Document .....	26
Figure 2.3.1: Lexical Centroids Adapted from (Salton et. al, 1996).....	34
Figure 3.1: A Collection of 5 Documents {D1, D2, D3, D4, D5}.....	47
Figure 3.1.1: Overall process of text information mining .....	48
Figure 3.1.2 The Semantic Partition Algorithm, (SEM_P).....	49
Figure 3.1.3: Feature Extraction Algorithm .....	50
Figure 3.1.3.1: Documents D1 to D5 after Feature Extraction.....	51
Figure 3.1.3.2: The Generate Semantic Partition Algorithm.....	54
Figure 3.1.4: Semantic Partitions on words in Documents D1, D2, D3, D4 and D5 .....	55
Figure 3.1.4.1: Ranking Semantic Partition Algorithm.....	56
Figure 3.1.4.2: Representation of Group 1 from Table 3.1.2 .....	58
Figure 3.1.4.2a: The Merging Partition Algorithm .....	58
Figure 3.1.4.2b: The Merging Partition Algorithm .....	60
Figure 3.1.5: A Concept Hierarchy for Frequent 1 items, $L_1$ .....	63
Figure 3.1.5.1: The algorithm to Generate Concept Hierarchies.....	64
Figure 3.1.5.2: The algorithm for Generating HashMap Identifier .....	65
Figure 3.1.6: Document D6 to be classified .....	66
Figure 3.1.7: Enhanced Semantic-Partitions Algorithm (ESEM-P).....	69
Figure 4.2: Overall process in the experiments .....	71
Figure 4.3.1: Memory Reduction after Preprocessing Text .....	73
Figure 4.3.2: Comparison on Text Document Preprocessing Times.....	73
Figure 4.4.1 Execution time for Building Classification Model with (Apriori).....	75

## **Chapter 1: INTRODUCTION**

The world has accepted computers as the best means for storing information. This is due to the fact that it is very easy to save data, it is convenient, any one with access to a computer can do it, and most importantly, information stored can be shared among many users, or transferred to other locations. However, as more text documents are stored in large databases, it becomes a huge challenge to understand hidden patterns or relationships between the stored documents. Since text data are not in numerical format, they cannot be analyzed with statistical methods.

Various mechanisms have been proposed for analyzing textual data. These include, clustering algorithms that classify documents into a constant number ( $k$ ) of distinct clusters (Krishna and Krishnapuram, 2001). This becomes a problem when the text documents themselves do not fit into these  $k$  clusters. Categorization is another approach that has been used (Bekkerman and Allan, 2003), where predefined classes are given. A scan performed on source documents assigns each document to the class that best represents it. This approach fits only domain-specific environments, thus documents that do not have predefined categories are not analyzed.

Probabilistic models assign various weights to different words in a document (Meir and Zhang, 2003), but some core key words with low occurrence or frequency end up getting the lowest probabilistic measure leading to poor analysis. Association rules have also been used in creating text summaries. However, the algorithms used are based on the traditional Apriori-like structure that normally performs recursive scans on the entire database to get frequent items. This was proved to be slow and inefficient in (Zaiane and Antonie, 2002).

## **1.2 Text Information Mining**

Text mining is the discovery of not yet known information from different written sources. i.e. text documents. The goal is to be able to link together related documents based on their context. Text mining is different from classic data mining in that natural language text like, letters, journals, books and emails are the initial texts to be mined. These texts have to undergo some preprocessing stages before the actual mining procedure is done, (Hearst, 1997).

### **Why Text Mining**

Researchers for decades have been concentrating on discovering knowledge from structured datasets, however, much of business and government data are stored in textual format and there is a growing need to understand this data. Various mechanisms have been used for mining knowledge from text including, Information retrieval (Salton et al., 1996, Stairmand, 1997, Eiron and McCurley, 2003). Information extraction (Turney, 2002, Chuang and Yang, 2000, Kotcz et al., 2001, Yonatan et al., 2001, McDonald and Chen, 2002, Mooney and Bunesco, 2005). Text clustering (Baker and McCallum, 1998, Nomoto and Matsumoto, 2001, Han et al., 2003, Zhai et al., 2004). Text summarization (Hahn and Mani, 2000, Gong and Liu, 2001, Hu and Liu, 2004, Okumura et al., 2004, Mei and Zhai, 2005). Text Classification (Huang et al., 2004, Castillo and Serrano, 2004), and Association Mining (Holt and Chung, 1999, 2005, Lin and Pantel, 2001, Nahm and Mooney, 2002, Zaiane and Antonie, 2002, Sakurai and Suyama, 2004).

#### **1.2.1 Information Retrieval**

Having a collection of documents, one would like to find documents related to a certain topic. A query is normally submitted to the database and the documents that are evaluated as having some relevance to the submitted query are retrieved, (Eiron and McCurley, 2003). These retrieved documents are normally indexed with a relevance measure where the highest ranked documents are displayed first. An example of such an information retrieval system is the popular “Google” website (Dakova, 2006).

Figure 1.2.1 shows an information retrieval sample after the query “Text information mining” is given as the search key. Documents online that contain any of the terms in the search key are retrieved.

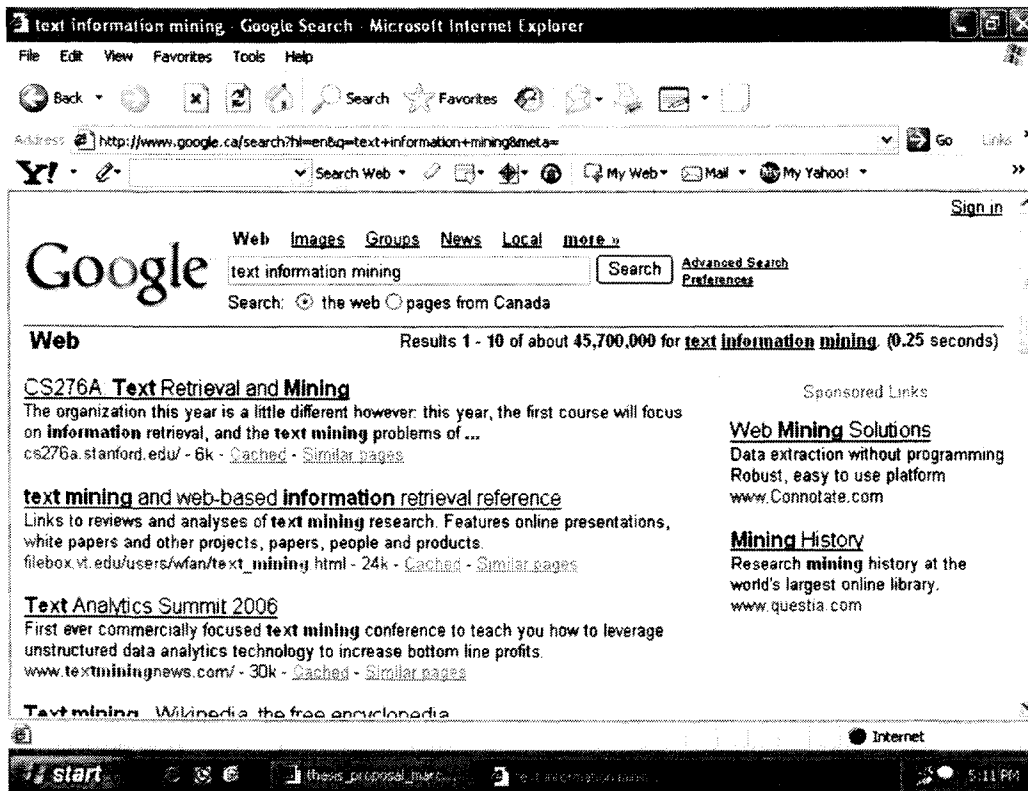


Figure 1.2.1: An Information Retrieval Procedure.

## 1.2.2 Text Extraction

This is the process of identifying specific pieces of data from text documents then extracting it, (Mooney and Bunescu, 2005). This is also referred to as information extraction. One type of information extraction is the named entity recognition described in (Patman and Thompson, 2003) and (Bikel et. al., 1999, Bunescu et al., 2005) where references to particular objects such as names, companies, locations from texts are identified and then extracted.

Several predefined patterns are involved in information extraction, (Sukhahuta and Smith, 2001), these patterns are seen as triggers. That is, if certain terms are found in text, then they trigger an extraction pattern. Sample trigger terms described in (Riloff, 1999) include: “is a”, “with”, “by”, “of”, etc. Now a trigger pattern looks like this; <Subject> is a <subject>, <Subject> by < Subject>, or <Subject> with <Subject> each of the subjects are terms extracted from text, due to the presence of a trigger term.

For example, using the text segment in Figure 1.2.2, suppose the above three patterns are used for text extraction, then the words in Figure 1.2.2.1 are identified as subjects and extracted. These terms are then analyzed depending on users’ needs.

What is backpropagation?” Backpropagation is a neural network learning algorithm. The field of neural networks was originally kindled by psychologists and neurologists who sought to develop and test computational analogues of neurons. Roughly speaking, a neural network is a set of connected input/output units where each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to predict the correct class label of the input samples. Neural network learning is also referred to as connectionist learning due to the connections between units.

Figure 1.2.2: Sample Paragraph in a Text Document

Backpropagation, neural, network, learning, algorithm,  
psychologists, neurologists, neurons, connected, input, output, units,  
adjusting, weights, samples, connectionist, learning

Figure 1.2.2.1: Final Extracted Words from Text Document in Figure 1.2.2

### 1.2.3 Text Summarization

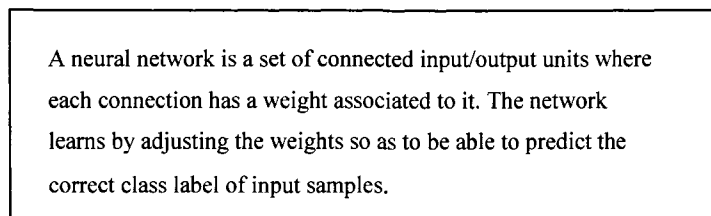
Summarization is the process of obtaining the most important information of a document, (Sengupta et al., 2004). This obtained information is much smaller in content than the original document and therefore referred to as a document summary. There are two techniques used in summarization. The first method is extraction based approach as described in (Okumura et al., 2004), where only contents extracted from a document are

used to create a summary. The second method uses abstraction (Hu and Liu, 2004), some of the contents used in creating the summary do not come from the original document instead, they are selected from predefined sets of vocabulary that act as summary enhancements.

First documents are identified with a certain predefined set of known document types, (Brandow et al, 1995), this includes, {headlines, outlines, minutes, biography, chronologies etc}. Given the genre of the document, key sentences are identified.

Using the summarization described in (Kupiec et al., 1995), the text segment in Fig 1.2.2 is summarized as follows;

Sentence lengths are identified, the sentences with a length greater than five is seen as potential for inclusion in a summary. There are 5 sentences;  $S = \{S1, S2, S3, S4, S5\}$  in Figure 1.2.2, each having the following recorded lengths  $|S1| = 7$ ,  $|S2| = 13$ ,  $|S3| = 12$ ,  $|S4| = 13$ ,  $|S5| = 9$ . The top ranking sentences are then selected from the set  $S$ ,  $S' = \{S2, S3, S4\}$ . Using the discourse marker in (Marcu, 1999) further discussed in section 2,  $S3$ , and  $S4$  are further evaluated. The “,” character is seen as a sentence divider and one part of the sentence is seen as a nucleus and the other part is a satellite (Marcu, 1999). The final summary will contain parts of  $S3$  and  $S4$  as seen in Figure 1.2.3.



A neural network is a set of connected input/output units where each connection has a weight associated to it. The network learns by adjusting the weights so as to be able to predict the correct class label of input samples.

Figure1.2.3: Final summary of Text Segment in Figure 1.2.2.

#### 1.2.4 Text Classification

Given predefined classes, classification is the process of assigning appropriate classes to subsets of a database also called supervised learning, (Huang et al., 2004). A classification model is normally generated using parts (samples) of a database; this is normally called the training data set, (Yu et al., 2003). The remaining portions of the

database are classified using the training set; this portion is normally referred to as testing set. The accuracy of the classification model depends on how well the testing data is classified using the generated classification model.

Various methods are used in developing classification models and these include decision trees described by (Chickering et al., 1997), Naives Bayesian in (Good, 1965, Calvo et al., 2004), distance based algorithms such as Support Vector Machines (SVMs) in (Vapnik, 1995).

**Text Classification with Naïve Bayesian Theory, (Calvo et al., 2004).**

Let there be a set C of predefined classes,  $C = \{C_1, C_2, C_3\}$ , and each  $C_i$  is composed of predefined set of terms,  $C_i = \{t_1, t_2, \dots, t_m\}$ , where m is the number of terms in each  $C_1$ . Given a document d, d is to be classified as either in  $C_1, C_2$  or  $C_3$  using posterior probability,  $\Pr(d_i | c_i)$ . This is the probability that a document  $d_i$  and a class  $c_i$  occur together. This is calculated as;  $\Pr(c_i | d_i) = p(d_i | c_i)p(c_i) / p(d)$  where  $c_i$  is a class and  $d_i$  is the document. If d is taken to be the text paragraph in Figure 1.2.2, and  $C_1 = \{\text{neural}\}$ ,  $C_2 = \{\text{networks}\}$ , and  $C_3 = \{\text{computing}\}$ , then classification is carried out as follows.

First preprocessing is done by removing stop words where common words such as {the, an, a, an, of, was, by etc.,} are eliminated. The remaining list of words together with their frequency count is; {backpropagation 2, neural/neurons 4, networks 4, learning 3, algorithm 1, field 1, originally 1, kindled 1 .....weights 1}. Total number of words remaining in d is taken as 55. The number of times d contains any of the terms in the three predefined classes is recorded as follows;

$$\begin{aligned} P(w_i | \text{neural}) &= p(\text{neural} | w_i) p(\text{neural}) / p(w_i) \\ &= (3/55) (50/100) = 0.0273 \\ P(w_i | \text{networks}) &= p(\text{networks} | w_i) p(\text{networks}) / p(w_i) \end{aligned}$$



$$= (4/55) (50/100) = .0364$$

$$P(w_i | \text{computing}) = p(\text{computing} | w_i) p(\text{computing}) / p(w_i)$$

$$= 0(.5) = 0$$

$P(\text{computing})$ ,  $p(\text{neural})$  and  $p(\text{networks})$  are prior probabilities estimated from a training set, in this example 50% prior probability is used. The document is assigned to category  $C_1$  and  $C_2$  as their posterior probability is greater than zero.

### 1.2.5 Text Clustering

Clustering is the process of grouping objects into classes with similar components (Zhang et al., 2002, Han et al., 2003). A collection of data objects that are similar is called a cluster. In machine learning clustering is referred to as unsupervised learning; there are no predefined classes or training labels used. More often, clustering is known as learning by observation (Krishna and Krishnapuram, 2001).

In Text Information Mining, documents are represented as a data matrix such that if there are  $n$  number of documents to be clustered, and each is represented by  $m$  terms, an  $m \times n$  matrix is created. Figure 1.2.4. shows a representation of  $n$  documents with  $m$  terms each.

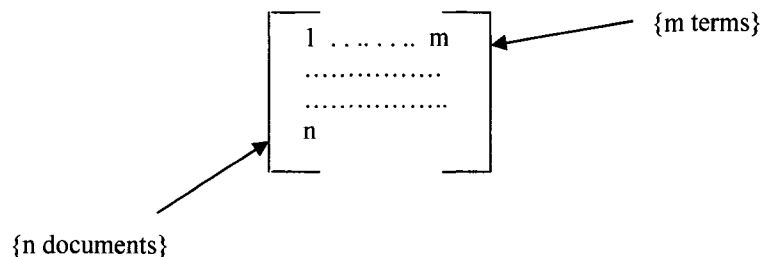


Figure 1.2.4: A Sample Term Matrix

The most common method for data clustering is the  $k$ -means clustering (Nomoto and Masumoto, 2001),  $k$  represents the number of clusters to be generated after the clustering procedure is done.

Neural networks involve long training times and are therefore more suitable for applications where this is feasible. They require a number of parameters for applications where this is feasible. They require a number of parameters that are typically best determined empirically, such as the network topology or "structure." Neural networks have been criticized for their poor interpretability, since it is difficult for humans to interpret the symbolic meaning behind the learned weights. These features initially made neural networks less desirable for data mining. Advantages of neural networks, however, include their high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained. In addition, several algorithms have recently been developed for the extraction of rules from trained neural networks. These factors contribute towards the usefulness of neural networks for classification in data mining.

Figure 1.2.5: A Sample Text Document to be Clustered

Suppose a data set has 3 dimensions and the cluster has 2 points  $x, y$  and a centroid  $z$ ;  $x = (x_1, x_2, x_3)$ ,  $y = (y_1, y_2, y_3)$  and  $z = (z_1, z_2, z_3)$ .  $z_1 = (x_1 + y_1)/2$ ,  $z_2 = (x_2 + y_2)/2$ ,  $z_3 = (x_3 + y_3)/2$ . The cluster centroid is randomly selected, each point in the matrix is assigned to the nearest cluster center and then a new centroid for each cluster is recalculated using the new cluster member values. For example, using Figure 1.2.2 and Figure 1.2.5 as two text documents, the frequencies of the highly occurring words in both documents are recorded in Table 1.1, (Krishna and Krishnapuram, 2001).

Terms	Frequency
Neurons	7
Networks	5
Learning	3
Backpropagation	2
Algorithm	1

Table 1.1: Highly occurring Terms in Figure 1.2.2 and Figure 1.2.5

These words are picked to represent each document in a frequency matrix [2 x 5], Document1 ( $d_1$ ) = 5 4 3 2 1 and Document2 ( $d_2$ ) = 6 6 0 3 2, using Euclidean distance measure defined as;  $d(i, j) = \sqrt{\sum (X_i - Y_j)^2}$ . Let  $X_i, X_j = d_1, d_2$  respectively, and

suppose one wants to cluster the document in two clusters,  $C_1$ , and  $C_2$ , random numbers are picked to represent the centroids for both  $C_1$  and  $C_2$ , as shown below;

$X_i$	$X_j$	$C_1$	$C_2$
5	6	3	1
4	6	3	1
3	0	3	1
2	3	3	1
1	2	3	1

Table 1.1.2: Matrix Representation of Sentences and Clusters  $C_1$  and  $C_2$ .

Then the distances between the columns in Table 1.1.2 is calculated as follows;

$$d(X_i, C_1) = (5-3)^2 + (4-3)^2 + (3-3)^2 + (2-3)^2 + (1-3)^2 = \sqrt{4+1+0+1+4} = \sqrt{10}$$

$$d(X_i, C_2) = (6-3)^2 + (6-3)^2 + (0-3)^2 + (3-3)^2 + (2-3)^2 = \sqrt{9+9+9+1+1} = \sqrt{28}$$

$$d(X_j, C_1) = (5-1)^2 + (4-1)^2 + (3-1)^2 + (2-1)^2 + (1-1)^2 = \sqrt{16+9+4+1+0} = \sqrt{30}$$

$$d(X_j, C_2) = (6-1)^2 + (6-1)^2 + (0-1)^2 + (3-1)^2 + (2-1)^2 = \sqrt{25+25+1+4+1} = \sqrt{56}$$

$D_i$  represents the  $i$ th iteration in distance measure between a data set and a cluster centroid while  $C_i$  represents the clustering allocated for the  $i$ th iteration. For example,  $D_0$  in Table 1.1.3 shows the first calculation of differences in distances between columns in Table 1.1.2.  $C_0$  represents the first clusters assigned to the documents.

$D_0 = \begin{bmatrix} \sqrt{10} & \sqrt{28} \\ \sqrt{30} & \sqrt{56} \end{bmatrix} = \begin{bmatrix} 3.16 & 5.29 \\ 5.48 & 7.48 \end{bmatrix}$	$C_0 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \text{group 1}$ $\phantom{C_0} = \text{group 2}$
---	---

Table 1.1.3: The first Distance Calculation ( $D_0$ ) Generating ( $C_0$ )

Based on the minimum distance, only one cluster is assigned to both documents.

Recalculating the new centroid in each group is then done, in this case the centroid at group 2 remains unchanged at (1, 1, 1, 1, 1) as no objects are assigned to it, but the new centroid for group 1 has changed as follows;  $C_2 = ((5+6)/2, (4+6)/2, (3+0)/2, (2+3)/2, (1+2)/2) = (11/2, 10/2, 3/2, 5/2, 3/2)$ . Now after recomputing the centroids, the documents are assigned to new centroid as shown on Table 1.1.4.

$$D_1 = \begin{bmatrix} 7.0171 & 6.325 \\ 5.48 & 7.48 \end{bmatrix} \quad C_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{array}{l} = \text{group 1} \\ = \text{group 2} \end{array}$$

Table 1.1.4: The Second Distance Calculation (D1) Generating (C1)

New centroid for group I is recomputed, this re-computation of cluster centroid continues until each group member remains unchanged.

### 1.2.6 Association Rules

Association rules have been used extensively in data mining research where transactions are stored in a structured database, see Table 1.2. Associations among these different transactions are discovered through association algorithms which are based on the original Apriori (Agrawal and Srikant, 1994) and frequent pattern trees in (Han et al., 2000). Rules of the form  $x \rightarrow y$  are generated where both  $x$  and  $y$  are subsets of the database but  $x \neq y$ .

TID	Item-sets
TID1	Eggs, Milk, Bread
TID 2	Milk, Egg, Bread
.	.
TIDn	.

Table 1.2: A Structured Database with Unique Transaction Ids.

Extracting the commonly occurring itemsets in the different transactions normally generates association rules. For example, if several transactions contain milk, bread and eggs, then a rule of the form  $\{\text{milk, bread}\} \rightarrow \{\text{eggs}\}$  can be generated. This is some kind of market basket analysis where a prediction can be made that whenever a customer purchases milk and bread, then there is a high possibility that they will also buy eggs. In marketing analysis, keeping the three items closer in a grocery store could increase sales. In text information mining the same relations can be identified from different documents, (Zaiane and Osmar, 2002; Phan et, al 2005). First data preprocessing is done by removing

stop words and stemming, normally the porters stemming algorithm described in (Rijsbergen et.al, 1980) is used to remove suffixes of related words and having a single word as the representation. For example, having a set of words  $S$ ,  $S = \{\text{consider, considerably, considered}\}$ , all the words in  $S$  share the same prefix “consider”. The stemming algorithm identifies suffixes of the form  $\{\text{ably, ered, ies, ation etc.}\}$ . When stemming is applied to  $S$ , all these suffixes are removed and one word is left to represent the entire set  $S$ .

A document  $D_1$  with  $n$  sets of words is represented as a transaction.  $D_1 = \{w_1, w_2, \dots, w_n\}$ . The words  $w_1$  to  $w_n$  are the itemsets in the transaction. Just like in a market transaction, the document is represented with a unique ID.

For example, using the text extraction patterns described in section 1.2.2, if three terms are selected to represent each of the text segments in Fig 1.2.2 and Fig 1.2.5;  $d_1$  and  $d_2$  respectively, then  $d_1 = \{\text{neural, networks, learning}\}$   $d_2 = \{\text{networks, topology, neural}\}$ . This forms the transaction table in Table 1.3.

Document ID	Itemset1	Itemset2	Itemset3
D1	Neural	Network	Learning
D2	Network	Topology	Neural

Table 1.3: Transaction representations of text segments in Fig 1.2.2 and Fig 1.2.5

Using the Apriori algorithm in (Agrawal and Srikant, 1994), Table 1.3 can now be mined. First scan of the transactions results to Table 1.4.1, where each itemset is matched with its frequency count in the entire transaction set. If minimum support = 2, with the total number of itemsets in the transactions as 5, the percentage support is  $2/5 = 40\%$ . Table 1.4.2 shows the itemsets with support  $\geq$  minimum support, this is  $L_1$ .  $C_2$  is generated by performing a join of  $L_1$  with itself, see Table 1.4.3,  $L_2$  is the set with minimum support from  $C_2$  as seen on Table 1.4.4.

Itemset $C_1$	Support Count
Neural	3
Network	3
Learning	1
Topology	1
Advantages	1

Table 1.4.1: Itemsets generating  $C_1$

Itemset $L_1$	Support_Count
Neural	3
Networks	3

Table 1.4.2: Itemsets with minimum support

Itemset $C_1$	Support_count
Neural networks	3

Table 1.4.3:  $C_1$  generated from  $L_1$

Itemset $L_2$	Support_count
Neural networks	3

Table 1.4.4: Generating  $L_2$  from  $C_2$

Finally  $L = \{L_1 \cup L_2\} = \{\{\text{neural, network}\} \cup \{\text{neural network}\}\}$ , the following rule is then derived;  $\text{neural} \rightarrow \text{networks}$ .

### 1.3 The Motivation of Thesis

Text extraction serves as the baseline for input to all the other text information mining methods described in section 1.2. However, there is a huge challenge as to how sentence segments should be extracted from text for them to yield important information about the original document. At the same time, when a segment is extracted, can it be combined with other extracted segments to form a document summary?, (Chuang et. al., 2000). Since documents are not structured in a standard way, does structuring the data before extraction of features make the procedure more feasible? These are some of the important questions that are addressed by (Mooney and Bunescu, 2005).

Accuracy in feature selection is regarded as one criteria for measuring a text-summarization mechanism (Forman, 2003). If a user looks at extracted document segments, they should be able to infer what would be the real context of the original text. Any system that provides such knowledge would be ideal for text information mining. However, present systems are not able to handle documents from multiple sources as they

are mostly domain specific, and do not generate meaningful and timely information, (Hahn and Mani, 2000).

For a document summary to represent an original document, the semantics in the summary should be a component of the original document. Terms that are closer in meaning should be grouped in the same summary. Various studies have been done on summarization using lexical analysis by (Silber and McCoy, 2000 ), discovery of rules by understanding the lexical knowledge in the document (Sakurai and Suyama, 2000) and using semantic orientation of document segments in (Turney and Littman, 2003).

Redundancy is another huge problem for document summaries. Various researchers have introduced weighting measures and ranking mechanisms to deal with this problem (Gong and Liu, 2001; Kotcz et. al., 2001; McDonald and Chen, 2002). Terms scoring lower than predefined thresholds are seen as redundant and therefore eliminated.

In traditional databases, the Apriori approach in (Agrawal and Srikant, 1994) was the foundation of many rule-generation algorithms. It was originally designed for identifying frequent patterns in structured data. Apriori requires several scans of the entire database thereby taking up more processing power and memory. This leads to much inefficiency. Improvements to the Apriori approach could still lead to generation of frequent patterns in text documents (Holt and Chung, 1991; Zaiane and Antonia, 2002; Phan et. al., 2005).

The final rules in both data mining and text mining do not have any linkage to the original transactions. For example, if a rule such as  $\{A,B,C \rightarrow D\}$  is generated, this rule clearly shows that all the four itemsets must be frequent but, the transactions which they come from are not given. In market basket analysis, this aspect might not be necessary, however in text information mining, the original source of a frequent itemset could be useful in assigning concepts to different documents, and this would normally form a linkage between documents with same amount of frequent itemsets.

By combining feature extraction, summarization and association rule mechanisms, a robust system that provides text understanding by linking together documents that are identified as having some common grounds could be developed. Feature extraction could be used to identify unique terms in a document; these terms could serve as the document summary. Words in such a summary can be used to represent a document in a transaction database. Several such summaries from different documents could be applied to a data mining algorithm such the Apriori described in (Agrawal and Srikant, 1994), thereby finding associations between different documents.

The major problem with text information mining is the amount of words that are to be processed, but this could be significantly reduced when subsets of the documents are used. Summarization with weighting and ranking mechanisms appear to be more promising in eliminating redundant words. However, existing weighting and ranking algorithms do not consider semantic orientation of words and therefore suffer from the following drawbacks: (1) summaries generated could contain words that seem irrelevant to the understanding of the document context; (2) key words in a document could be eliminated if a sentence containing these words is ranked lower than a given threshold; (3) if summaries are to serve as input for mining algorithms like the Apriori, then further processing of each summary must be done as a summary is often not structured like a relational database transaction.

This thesis proposes Semantic Partitions (SEM-P) and Enhanced Semantic Partitions (ESEM-P) algorithms based on the semantic orientation of words in a document. This partitioning reduces the amount of words required to represent a document as input for discovery of frequent patterns while still maintaining the semantics of a document. A weighting and ranking heuristic measure for each term in a partition is used in ESEM-P to prune low ranked terms resulting in improved performance on the ESEM-P over the SEM-P. Identified frequent patterns generate concept hierarchies and hash map identifiers for visualization purposes.



## 1.4 The Thesis Contribution

Given a collection of text documents  $t$ , the thesis proposes two algorithms for text information mining based on frequent pattern generations. The main aim is to be able to find associations between the documents in  $t$  using the Apriori algorithm, (Agrawal and Srikant, 1994). However, a huge challenge exists when dealing with text documents since text documents must undergo several preprocessing stages before any actual mining can be done.

This thesis contributes to the text information mining problem as follows;

1. Providing a new system that links together text documents based on their semantic content.
2. Providing a structured representation of a text document that portrays the exact semantic content of a document. This structure acts as a document summary.
3. The structured summaries in (2) serves as individual items in a transaction dataset.
4. Unlike other systems, the proposed system contains much fewer words per document leading to improved computation time and storage space.
5. Identifying frequent patterns from related documents then generating concept hierarchies that act as classification models.
6. Documents containing frequent patterns in (5) are represented in a Hashmap identifier for visualization purposes.

Using a natural language process described in (Brill, 1992), understanding the part of speech for each word in a document is done. Unlike the classification system used by (Hu and Liu, 2004) where customer reviews are grouped based on specific adjectives in sentences, the proposed system uses words identified as nouns. The WordNet ontology described in (Miller, 1995) is used to retrieve the meanings of each noun, (semantics). Words found to have similar meanings are said to be semantically related and therefore form semantic partitions. No two semantic partitions in the same document can have the same elements. For example, let  $S_1$  and  $S_2$  be two semantic partitions for a certain document  $D$ .  $S_1 = \{t_1, t_2, t_3\}$  and  $S_2 = \{w_1, w_2, w_3\}$ , all terms  $t_i$  in  $S_1$  are semantically

related, no term  $t_i$  exists in  $S_2$  unless  $t_i = w_i$  and  $\{S_1\} = \{S_2\}$ . Therefore  $S_1$  and  $S_2$  must be distinct.

Using semantic partitions greatly reduces the amount of words processed by the Apriori algorithm resulting in reduced computation time in comparison to the categorizer algorithm described in (Zaiane and Antonie, 2002) which finds associations between documents. The proposed algorithms will be faster, less expensive and more scalable compared to related text mining algorithms discussed in the literature that are mostly domain specific (Gong and Liu, 2001, Zaiane and Antonie, 2002, Hu and Liu, 2004, Mooney and Califf, 2005).

### **1.5 Outline of the Thesis**

The rest of the thesis is organized as follows: Chapter 2 reviews existing works on text information mining. Detailed description of the proposed algorithms is presented in chapter 3. Implementation and testing details are in Chapter 4 and the conclusions and discussions on future works are presented in Chapter 5.

## **Chapter 2: RELATED WORKS**

In this chapter, algorithms that have explored the text information mining are reviewed. The review consists of algorithms of four different structures which include; extraction based, (Chuang and Yang, 2000, Yonatan et. al., 2001, Castillo and Serrano, 2004, Mooney and Califf, 2005), ranking and weighting, (Gong and Liu, 2001, Kotecz et. al., 2001, McDonald and Chen, 2002), lexical and semantic Analysis in (Silber and McCoy, 2000, Turney and Littman, 2003, Sakurai and Suyama, 2004) and association based in (Holt and Chung, 1999, Zaiane and Antonia, 2002, Kongthon, 2004).

### **2.1 Feature Extraction Algorithms**

There is a huge challenge as to how sentence segments should be extracted from text for them to yield important information about the original document. When a segment is extracted, can it be combined with other extracted segments to form a document summary? “A summary will not be as good as an abstract”, (Chuang and Yang, 2000). Since documents are not structured in a standard way, does structuring the data before extraction of features make the procedure more feasible? All these are some of the questions asked by researchers trying to understand text documents (Chuang and Yang, 2000, Yonatan et. al., 2001, Castillo and Serrano, 2004, Mooney and Califf, 2005).

#### **2.1.1 Cue Markers**

An automatic text summarizer was developed by (Chuang and Yang, 2000). They proposed a method that used cue markers in extracting segments from sentences and by providing a set of key words; segments that include those key words were used to create a document summary.

First a sentence was identified as having different segments called clauses. Special phrases described in (Marcu, 1996) as “cue markers” were used to identify the different segments in a sentence. The phrases include words like words like; “there is”, “but”, “because”, “if”, “however”, “with” etc. The idea is to understand what parts of a sentence can be understood if separated from the other parts of a sentence. Cue markers act as a splitting spot for any sentence.

For example, the first sentence in Figure 2.1.1 can be split into two segments S1, and S2. The first segment S1 contains, [with the fast growing popularity of the internet and the worldwide web also known as {"www" or the "web"}] and second segment S2 contains [there is also a fast growing demand for web access to databases]. The splitting spot is identified by the presence of the words "there is" in the middle of the sentence. Further evaluation of the two segments S1 and S2 is done to identify any more special phrases in each individual segment. S1 contains the word "with" and no special phrase is identified in S2. S1 is seen as the subordinate of S2 due to the presence of the word "with" in S1, (Marcu, 1996). A subordinate segment is called a *satellite* and is seen as a description of another segment, S1 is the *satellite*. S2 is the segment being described in S1 and is referred to as a *nucleus*. A *nucleus* is taken as the main part of a sentence and can therefore act as a representation of the entire sentence.

Several relations between segments in a document are then formed based on what phrases are present in a segment; these relations are called "rhetoric relations". A relation  $r$  is defined as  $r(\text{name}, \text{satellite}, \text{nucleus})$ , where name is the relation formed between two segments, satellite represents the segment containing a cue marker and nucleus is the segment that does not contain any cue marker. The presence of the words such as "with" and "however" forms a justification relation. From the segments S1 and S2, a relation  $r$  is formed as;  $r(\text{justification}, s1, s2)$ .

More relations as described in (Marcu, 1996) includes thesis and antithesis; the antithesis relation identifies the presence of a word such as "but", "problem", "difficult" and "impossible" in a segment. A thesis relation is identified by the presence of words such as "in support" in a segment.

With the fast growing popularity of the internet and the world wide web also known as {"www" or the "web"}, there is also a fast growing demand for web access to databases. However, it is especially difficult to use relational database management {(RDBMS)} software with the web. One of the problems with using RDBMS software on the web is the protocols used to communicate in the web with the protocols used to communicate with RDBMS software.

Figure 2.1.1: A Text Segment Adapted from (Chuang and Yang, 2000).

[ With the fast growing popularity of the internet and the world wide web also known as {"www" or the"web"} 1], [ there is also a fast growing demand for web access to databases 2]. [However, it is especially difficult to use relational database management {(RDBMS)} software with the web 3].  
 [ One of the problems with using RDBMS software on the web is the protocols used to communicate in the web with the protocols used to communicate with RDBMS software 4].

Figure 2.1.2: Decomposition of Sentences in Figure 2.1.1 into Individual Segments

Figure 2.1.2 shows the identified segments of the original text in Figure 2.1.1. Each segment is given a numerical number. The second and third sentences are not segmented individually as no splitting spots are identified. The following relations are then developed between all the identified segments;  $r(\text{justification}, 1,2)$ ,  $r(\text{antithesis}, 3,2)$ ,  $r(\text{antithesis}, 4,2)$ ,  $r(\text{antithesis}, 4,1)$ , and  $r(\text{antithesis}, 3, 1)$ .

Several facts about a segment identified as a nucleus are maintained in a feature vector. A feature vector  $f = \langle PO, WF, BW, CS, CN \rangle$  where  $PO$  = Position of a segment in the original document,  $WF$  = Word frequency in a segment,  $BW$  = Bonus Words; (predefined set of words),  $CS$  = No of times a segment appears as a satellite and  $CN$  = Number of times a segment appears as a nucleus. The segments with highest scoring vectors are used to create a summary. If bonus words  $BW = \{WWW, \text{web}, \text{RDBMS}\}$ , a feature vector  $F_i$  represents the entries for a segment  $i$ . Using Figure 2.1.2, if  $i = 4$ , then  $f_4 = \langle 1,4,4,2,0 \rangle = 11$ ,  $i = 2$  then  $f_2 = \langle 1,3,3,0,3 \rangle = 10$ . The other two segments are disqualified as they do not contain any of the words in set  $BW$ .

**Problem with this algorithm:**

This method suffers in three main areas: (1) it is limited to a certain domain of the supplied key words and does not scale well to documents with varying topics, (2) it is not that obvious how predefined set of words; (bonus words) are to be supplied, one has to know the contents of the documents to know what would be seen as a feature necessary to be picked as a bonus words, and (3) if a bonus word is not found in a document, this could lead to discriminating segments that might be core to the meaning of a text document.

### **2.1.2 ClearStudio**

ClearStudio was proposed by (Yonatan et. al, 2001), a system for mining text through information extraction. The system includes rules for defining important features to be extracted from a document. These features include events, facts and words with meaning within the extraction domain.

The system consists of several steps:

1. For each document, extraction is performed. This extraction aims at identifying events, facts and any words that have some kind of meaning to the document domain. Example of such an event would be a management change in an organization, an example is given below.

Rules used to discover special events in a document are developed using DIAL (Declarative Information Analysis Language), a language developed in (Fisher et. al 1995) for information extraction purposes.

#### **Basic Elements in DIAL:**

The language is designed to capture sequences and patterns from text. The language identifies elements like:

1. Predefined sets of strings.  
An example of predefined strings is {"merger", "union", "collaboration"}.
2. Word Class elements: Predefined sets of phrases that share semantic meaning.  
An example, WC-States, this would hold a list of all states in the US.
3. ASCII characters for example HTML tags would be captured with @HTMLTAG, and capital letters with @capital.
4. Compound features: This would include a combination of several features including the above three. For example, Ohio State would match the @Capital and WC-States.
5. Recursion is applied to capture all predefined features.

### **Rule Definition as Created with ClearStudio:**

Let  $P=\{P_1, P_2, ..P_n\}$  be a set of Patterns defined using DIAL, and  $N$  be a set of constraints operating on elements in  $P$ . Each  $i$  element in  $N = \{N_{ij}\}$ , a set of constraints operating on  $P_j$ . A rule  $R$  is defined as a conjunction of clauses  $C_i = B_i \rightarrow H_i$ , where elements in  $B$  are sets of literals from  $P$  and  $H$  is a head such that  $H$  is implied by conjunction of the literals in  $B$  and satisfying constraints in  $N$ . For example, suppose one wants to retrieve information about companies merging activities. Then, a rule must be developed for automatic identification of key elements regarding mergers. A sample of such a rule is:

```
Merger(C1,C2):-Company(Comp1) “and “ Company(Comp2),  
                WC-Merger(Merger),  
                Verify(Comp1, !@personName),  
                Verify(Comp2, !@personName).
```

The above rule looks for two company names  $C_1$  and  $C_2$ , it also looks for a word “merger” in a set of predefined words  $WC$ -Merger. The constraint is that the company names are not people’s names.

### **2.1.3 Rapier and DiscoTex Methods**

The same rule based approach was taken by (Califf and Mooney, 2003) in the Rapier system. The idea is to have a set of documents and a predefined structure called template, the system extracts words from the documents and fills the template with these words. The words extracted are called Slot Fillers (Fillers). To be able to extract these fillers, some information are expected of the surroundings of the filler in the underlying text document, and these are seen as patterns. The set of words before the filler is called pre-fillers patterns (pre-p) and the set of words after the filler is called post-filler patterns (pos-p). Using the pre-p, pos-p and fillers, rules are generated to aid in extraction procedures as discussed below.

First, text segments are tagged using the part of speech tags (POS) in (Brill, 1992), where words are marked with corresponding syntactic categories; nouns, verbs, adjectives, adverbs etc. For example, “a simple sentence” is tagged as “a<AT>simple <JJ>sentence<NN>”, where AT = Singular article, JJ=Adjective and NN=Noun. This tagging is done so as to create rules with certain constraints thereby eliminating some strings.

Pre-Filler Pattern	Filler Pattern	Post-Filler Pattern
1. syntactic:{NN,NNP}	1.word: undisclosed Syntactic : JJ	1. semantic: price
2. list: length 2		

Figure 2.1.3: Sample Extraction Rule in Rapier Adopted from (Califf and Mooney, 2003).

A rule is normally represented in three columns, the first column is the pre-p pattern and its constraints, the second column represents the filler and the third column represents a pos-p as shown in Fig 2.1.3. The rule extracts the value “undisclosed” from phrases such as “sold to the bank for an undisclosed amount” or “paid GESHneir Flooring an undisclosed price”. Two constraints are placed on column 1 enforcing that a pre-filler pattern should consist of nouns and proper nouns and must be of length 2, the middle column enforces that the term “undisclosed” must be present and is the term to be extracted (filler), and column 3 indicates that a post-filler pattern should be the term “price” or its synonyms. Using the set of extracted terms from various documents, DiscoTex system in (Mooney and Bunescu, 2005), organizes the terms in a structured database and then applies traditional mining algorithms like Apriori in (Agrawal and Srikant, 1994).

### **Problem with this extraction approach**

A different set of data structure is created for each topic domain; this is quite tedious and also takes up much processing time and memory space. The worst is that as new concepts and constraints are to be placed in the extraction process, new rules must be created to accommodate changes.



### 2.1.4 Multi-Strategy Approach

A different approach in text extraction was taken by (Castillo and Serrano, 2004). They used parallelism to develop HYCLA (Hybrid Classifier), a multi-strategy classification system. The system contains several learners and each learner takes two stages. The preprocessing phase and the elimination phase.

#### Preprocessing phase

A system receives a training sample of either scientific or hypertext documents. Four vocabularies are developed from any document received. For example, Fig 2.1.4 is a sample of a scientific document that is divided into four sections, each section forms a vocabulary group as seen in Table 2.1.4. The four groups are G1, G2, G3 and G4; G1 contains title words, G2 contains abstract words, G3 contains the plain text following the abstract and G4 contains the words in the reference list.

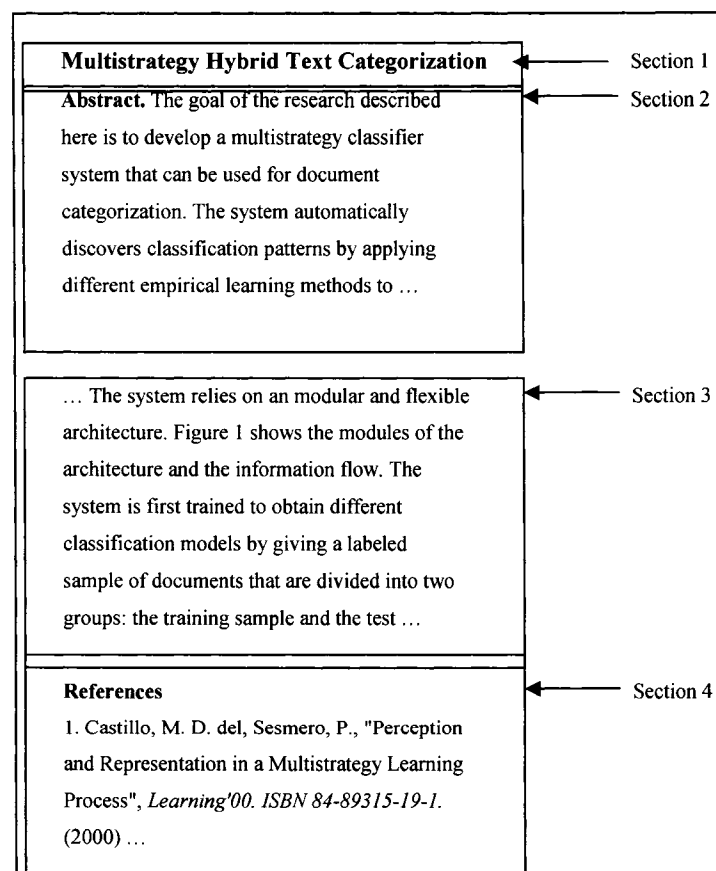


Figure 2.1.4: A Scientific paper Adapted from (Castillo and Serrano, 2004).

Group	Vocabulary List
G1	Multistrategy, hybrid, text, categorization
G2	Goal, research, described, develop, multistrategy, classifier, system, used, document, categorization, system, automatically, discovers, patterns, applying, different, empirical, learning ,methods
G3	System, relies, modular, flexible, architecture, shows, modules, architecture, information, flow, system, first, trained, obtain, different, classification, models, giving, labeled, sample, documents, divided, groups, training, sample, test
G4	Castillo, M. D. del, Sesmero, P., Perception , Representation, multistrategy, learning, process, learning, 2000, ISBN 84-89315-19-1.

Table 2.1.1: Vocabulary list of terms from Scientific Paper in Fig 2.1.4

After a count of term frequency in each vocabulary, a weight is given to each word depending on its position. For example, in Table 2.1.1, G1 terms are given the following scores; {Multistrategy = 3 x 10 = 30, hybrid = 1 x 10 = 10, text = 1 x 10 = 10, categorization = 1 x 10 = 10 }, Multistrategy get the highest score as it appears three times in the four groups combined and it's a title word. The final scores for some of the terms in the vocabulary list G1 and G2 are shown in Table 2.1.2.

Term	Frequency	Term	Frequency
Multistrategy	30	Develop	1
Hybrid	10	Multistrategy	30
Text,	10	Classifier	2
Categorization	10	System	2
Goal	1	Document	1
Research	1	Categorization	1

Table 2.1.2: A Frequency Count of Terms from G1 and G2 from Table 2.1.1

In order to reduce the size of the vocabulary lists, measures such as information gain, mutual information, document frequency, chi square and cross over entropy are applied in each term in a vocabulary list. For example, the term “multistrategy” has the following measures: Information gain, mutual information and document frequency all have a value of 30, which is taken as the frequency of the term in the document.

Given a category C, with terms that are relevant to that category, for example, if C = {multistrategy , categorization, system}, a string match is performed on the term “multistrategy” from G1 with the terms in C. If the term is present, then the odds ratio is taken as non-zero. The total scores for the term “multistrategy” is at least 90 while that of the term “goal” is just 3 as it is not a title word, and does not match the terms in C, therefore the term “goal” is eliminated. The highest scoring terms are the only ones left for classification {multistrategy, hybrid, text, categorization}.

### **Problem with this representation**

String matching categorization might be helpful in identifying key terms in various documents with the same kind of text context. However, this might not be helpful if a document that contains none of the categories being matched is very domain specific.

## **2.2 Ranking and Weighting Mechanisms**

Redundancy is a huge problem for document summaries. Various researchers have introduced weighting measures and ranking mechanisms to reduce the amount of words being analysed, (Gong and Liu, 2001, Kotcz et. al,2001, McDonald and Chen, 2002).

### **2.2.1 Summarization with Relevance Measure**

Text extraction with ranking was addressed by (Gong and Liu, 2000), they proposed two text summarization methods that ranked sentences extracted from original documents. Some of those ranked sentences were used to create document summaries.

The first method summarizes documents according to relevance measures while the second one used singular value decomposition. Both methods first break documents into individual sentences and then creates a weighted term-frequency vector for each sentence.

We have all heard inspirational stories about people who have faced a cancer diagnosis with courage. Perhaps you have had the privilege of cheering on a cancer survivor taking a victory lap in the *Relay For Life* or witnessed the camaraderie of a dragon boat team as they cross the finish line together. You may have wondered how so many people with cancer cope with this experience. A big part of the equation can be summed up in one word: support. Many will tell you that fighting this disease takes a team effort, a team that includes family and friends.

Figure 2.2.1: A Paragraph of a Text Document

### 2.2.1.1 Relevance Measure

Let there be a document segment  $S_i$ , then  $S_i$  is represented by a term-frequency vector  $T_i = \{t_{1i}, t_{2i}, \dots, t_{ni}\}$  where every  $\{t_{ji}\}$  represents the frequency of term  $j$  in segment  $i$ . Segment  $i$  could be a sentence, a paragraph or even the entire document itself. The term frequency of  $\{t_{ji}\}$  is computed as:  $\{t_{ji}\} = \text{Local weighting of term } j \times \text{Global weighting of term } j$ . Local represents a position of a document while global represents the entire document.

For example, the text segment in Figure 2.2.1. is broken down into sentences  $S$ ;  $S = \{S_1, S_2, S_3, S_4, S_5\}$  as shown in Table 2.2.1. The weighted term frequency of each  $S_i$  in  $S$  is computed and represented in a vector, the vectors for  $S_1$ ,  $S_2$  and  $S_3$  are;

$V_1 = \langle \text{heard } 1, \text{ inspiration } 1, \text{ stories } 1, \text{ about } 1, \text{ people } 1, \text{ faced } 1, \text{ cancer } 1, \text{ diagnosis } 1, \text{ courage } 1 \rangle$ ,  $V_2 = \langle \text{perhaps } 1, \text{ privilege } 1, \text{ cheering } 1, \text{ cancer } 1, \text{ survivor } 1, \text{ taking } 1, \text{ victory } 1, \text{ lap } 1, \text{ relay } 1, \text{ life } 1, \text{ witness } 1, \text{ camaraderie } 1, \text{ dragon } 1, \text{ boat } 1, \text{ team } 1, \text{ cross } 1, \text{ finish } 1, \text{ line } 1, \text{ together } 1 \rangle$  and  $V_3 = \langle \text{wondered } 1, \text{ people } 1, \text{ cancer } 1, \text{ cope } 1, \text{ experience } 1 \rangle$ . The relevance score for each  $S_i$  is computed by taking its corresponding vector and calculating its term frequency in comparison to the document frequency of the same terms. From the text segment in Figure 2.2.1., there are two frequent words, “cancer” has a frequency count of 3, and “people” has a count of 2.

Each of these terms is checked for occurrence in each term vector. In V1, “people” has a frequency of 1 x 2 and “cancer” has 1 x 3, total scores for both words in V1 = 6, V2 = 2, and V3 = 6. If a minimum score of 2 is given, then V2 is eliminated. The k sentences with the highest scoring relevance are added to the summary, if K = 2, then the summary will contain S1 and S3.

Sentence	Contents
S1	We have all heard inspirational stories about people who have faced a cancer diagnosis with courage.
S2	Perhaps you have had the privilege of cheering on a cancer survivor taking a victory lap in the <i>Relay For Life</i> or witnessed the camaraderie of a dragon boat team as they cross the finish line together.
S3	You may have wondered how so many people with cancer cope with this experience.

Table 2.2.1: Individual Text Sentences

### **Problem with this algorithm**

If all the sentences under processing contain the list of the frequent terms, then there would be no sentence elimination, the summary would be exactly the same as the original document. For example is S2 contained the term people, then all three sentences would have the same score, therefore the summary would contain all three sentences. Another challenge comes in determining which sentences to eliminate and which to keep depending on the k value provided by a user. For example, if the number of sentences to include in the summary is just 2, which among the three sentences should be eliminated and why?

#### **2.2.1.2 Singular Value Decomposition**

Singular Value Decomposition (SVD) is a technique well known in theory of matrices to reduce the sizes in frequency of terms in any given matrix, (Nicholson, 2001). The process starts by creating a sentence Matrix  $A = [A_1, A_2, \dots, A_n]$  where each  $A_i$  represents a weighted term-frequency vector of a sentence. A document having M terms and N

sentences can be represented by  $M \times N$  matrix. For example, let there be a set of terms  $T$ ,  $T = \{ t_1, t_2, t_3, t_4 \}$ , a count of the occurrence of these terms in two sentences  $S_1, S_2$  from a document  $d$  can be represented in a  $2 \times 4$  matrix as follows:

2	4	...	$t_1$
1	3	...	
0	0	...	
0	0	...	$t_4$
$S_1$	$S_2$		

2	1	0	0
4	3	0	0

$A^T$

Table 2.2.1.3: Transpose of Matrix A ( $A^T$ )

Table 2.2.1.2: A  $2 \times 4$  Sample Matrix Representation (Matrix A)  
(Adapted from Kuruvilla et al., 2002)

Calculating the singular vector decomposition of the matrix A in Table 2.2.1.2; First eigenvalues and eigenvectors are calculated for matrix A together with its transpose,  $A^T$ . A matrix transpose is the inversion of the columns to be rows and rows to be columns. The transpose of matrix A is shown in Table 2.2.1.3. Any  $n \times n$  matrix W can be represented with scalar numbers called eigenvalues ( $\lambda$ ), and a nonzero column X such that,  $WX = \lambda X$  (Nicholson, 2001). The eigenvalues of  $A^T A$  make up the columns in V and the eigenvalues for  $AA^T$  make up the column for U.

$$AA^T = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 4 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 20 & 14 & 0 & 0 \\ 14 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \leftarrow \text{Matrix W}$$

$$\begin{bmatrix} 20 - \lambda & 14 & 0 & 0 \\ 14 & 10 - \lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} \mathbf{x} = (\mathbf{W} - \lambda \mathbf{I}) \mathbf{x} = 0$$

Four eigenvalues are obtained by performing systems of equations on matrix WX, the results are  $X_1 = -0.58$ , and  $0.82$ ,  $X_2 = .082$  and  $-0.58$ ,  $X_3 = X_4=0$ . These values are used to generate matrix U as shown below as described in (Kuruvilla et al., 2002).

$$U = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad V = \begin{bmatrix} 0.40 & -0.91 \\ 0.91 & 0.40 \end{bmatrix}$$

The same procedure is carried out for  $A^T A$  resulting in matrix V above. Singular values for S are square roots of eigenvalues from  $A^T A$  or  $AA^T$ . The entries in S are arranged diagonally in descending order and are always real numbers.

$$S = \begin{bmatrix} 5.47 & 0 \\ 0 & 0.37 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Matrix S serves as a representation of matrix A in Table 2.2.1.2 (Alter et al., 2002).

### **Problems with this approach**

Although the actual representation of a document can be greatly reduced using a singular matrix, the technique has several drawbacks: (1) A lot of computation time is needed to calculate eigenvalues and eigenvectors that generate the singular matrix, (2) the individual meanings of words are ignored, (3) it is also not clear how a singular matrix can be mapped back to the original contents of a document, and (4) the ordering of values in a matrix could generate different singular matrixes.

### **2.2.2 Summarization as Feature Selection for Text Categorization**

A different approach was taken by (Kotcz et al., 2001). They proposed an algorithm for creating document summaries using only words extracted from the original document.

The algorithm first assigns a weighting measure for each feature extracted, terms identified as unique are evaluated according to their relevance weights. The words with the highest scores are then used to form a document summary.

### 2.2.3 Sentence-Selection Heuristic

A different approach was taken by (McDonald and Chen, 2002) where predefined sets of words and sentence heuristics were used to create summaries. They presented TXTRACTOR, a tool for ranking text segments. Three steps are used in creating a document summary:

#### 1. Sentence Evaluation:

Five heuristics are used to evaluate a sentence:

- (a) Presence of a cue phrase: Predefined set of phrases is checked in each sentence. These cue phrases includes words like, “in summary”, “in conclusion”, “in short” etc.
- (b) Proper Nouns - Checking capitalized words that do not include words beginning a sentence. The total of these words is then averaged for each sentence over the total number of words in a sentence; this is to ensure shorter sentences are treated fairly.
- (c) Word position – words beginning a document or paragraph are given a higher score.
- (d) Sentence length – longer sentences are also given a higher score than short ones.
- (e) TF \* IDF Weighting - The measure of how a term/word occurs in a sentence relative to its occurrence in the entire document.

Let  $t$  be a term in document  $D$ , and  $NOD$  denotes the number of sentences in  $D$ ,  $NOD_t$  denotes the number of sentences in  $D$  containing term  $t$ . The inverse document frequency of  $t$ ,  $IDF(t) = \log(NOD/NOD_t)$ . For example given a text document with four sentences as seen in Table 2.2.2, using the above mentioned heuristics, the results are shown in Table 2.2.3. The entry in sentence S1 is calculated as follows: cue phrases and proper nouns entry are empty as no cue phrases or proper nouns are found in the sentence.



The Term Feature is the entry for word beginning a paragraph and has a position 1 as this is the first sentence in the document. A scan of the entire text segment identifies terms that have a frequency count higher than 1, {problem 2, optimization 3, neurons 2, networks 2} the term “problem” has a frequency count of 2 in S1, therefore,  $TF*IDF(\text{problem}) = 2 \times \log(4/1) = 1.2041$ . This value is then averaged on the sentence length,  $|S1| = 17$ ,  $1/17(TF*IDF(\text{problem})) = 0.071$ . Measures of other sentences are shown in Table 2.2.2.

Sentence	Contents
S1	Feature selection in the context of practical problems such as diagnosis presents a multicriteria optimization problem.
S2	The criteria to be optimized include the classification accuracy, cost, and risk.
S3	Evolutionary algorithms offer a particularly attractive approach to multicriteria optimization because they are effective in high dimensional search spaces
S4	Neural networks are densely interconnected networks of relatively simple computing elements for example, threshold or sigmoid neurons

Table 2.2.2: Breakdown of Text Segment in Figure 2.2.1



S	Cue Phrases	Proper Nouns	Words Position	s	Frequent Terms	TF*IDT	Total TF*IDT
S1	-	-	Feature	17	Problem Optimization	0.0708 0.01771	0.08851
S2	-	-	-	12	Optimization	0.0251	0.0251
S3	Because	-	Evolutionary	18	Optimization	0.01672	0.01672
S4	For example	-	Neural	17	Neural Networks	0.0708 0.0708	0.1416

Table 2.2.3: Term Representation in Table 2.2.2

## 2. Topic Boundary Identification

The TextTiling algorithm in (Hearst, 1997) was used to identify topic boundaries. The algorithm divides text into tokens and then forms blocks of specified length. The aim is to check the similarity of words in each block by calculating the number of times a term occurs in a block. The more similar blocks are, the more the likelihood that the blocks talk of the same topic (Hearst, 1997).

For example, breaking down sentences S1, S2, S3 and S4 into blocks of twenty tokens is shown in Table 2.2.3., similarity between two blocks is calculated by taking the count of how many times a term occurs in the two blocks (McDonald and Chen, 2002). If four blocks are formed from the sentences in Table 2.2.3, S1 = block 1, S2 = block 2, S3 = block 3 and S4 = block 4, then similarity between block 1 and 2 is greater than 0 since the term “optimized” occurs in both blocks, therefore the two blocks are similar. Block 3 and 4 have a zero similarity value, this implies that the two blocks talk of different sub topics, (Hearst, 1997). Evaluating block 1, 2 and 3 shows that they talk of a similar topic and block 4 is identified as having a different topic.

## 3. Sentence Ranking

A ranking is done on the sentences based on the different scoring measures assigned from the heuristics as seen in Table, 2.2.3 last column. The highest ranking sentence from each topic is picked to represent that topic. S1 scores higher than either S2 or S3 and is picked. The final summary contains sentences S1 and S4.

### **Problems with this algorithm**

The weighting and ranking of terms seems to bring out good document representation as claimed by (McDonald and Chen, 2002). However, no two generated summaries from two different documents can be linked together to infer any kind of patterns or associations. The algorithm deals with one document at a time, similar to the summarizer in Microsoft Word, (Hahn and Mani, 2000).

## 2.3 Lexical Analysis

For a document to represent an original document, the semantics in the summary should be a component of the original document. Terms that are closer in meaning should be grouped in the same summary. Various studies have been done on summarization using lexical analysis in (Silber and McCoy, 2000), using semantic orientation in document segments in (Turney and Littman, 2003) and Discovery of rules by understanding the lexical knowledge in the document by (Sakurai and Suyama, 2004).

### 2.3.1 Efficient Text Summarization with Lexical Chains

Two main decomposition strategies are introduced in (Salton et. al, 1996) including: a chronological decomposition of text into segments, and semantic decomposition into text themes.

A text represented by a vector of weighted terms of the form  $D_i = (d_{i1}, d_{i2}, \dots, d_{it})$  where  $d_{ik}$  represents an importance weight for the term  $T_k$  attached to document  $d_i$ . The terms attached to documents for content representation purposes may be words or phrases derived from the document texts by an automatic indexing procedure. The term weights are computed by taking into account the occurrence characteristics of the terms in individual documents.

Assuming text is represented in vector form as a set of weighted terms, it is possible to compute pair wise similarity values showing the similarity between pairs of texts. This is based on a coincidence in terms assignments to the respective items.

The vector similarities are computed as the inner product between corresponding vector elements, defined as:  $\text{Sim}(D_i, D_j) = \sum_{k=1}^t d_{ik} \cdot d_{jk}$ , where  $\text{Sim} = 0$  for sets that are disjoint and 1 for complete identical sets. The documents are represented as nodes called vertices in a graph and an edge between 2 nodes represents the similarity between two texts as sufficiently large. A minimum threshold of 0.01 is taken as the minimum threshold to calculate similarity.

Suppose we have the following documents. 22387 → Thermometer Fusion, 19199 → Radioactive Fallout, 17016 → nuclear weapons, 17012 → nuclear energy, 11830 → hydrogen bomb and 8907 → fission nuclear, with a pre-computed threshold, see Figure 2.3.1, page 33.

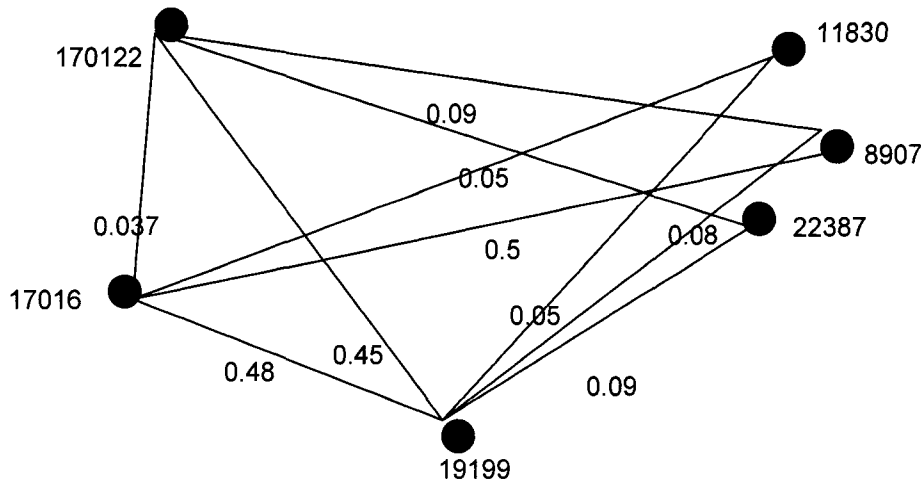


Figure 2.3.1: Lexical Centroids Adapted from (Salton et. al, 1996)

Each of these edges represents a pre-calculated threshold of 0.01 or higher. However, there can be refinement as to what is considered similar between 2 nodes. A central node means that it has the highest number of similarity between nodes.

A triangular path in the graph represents three mutually related paragraphs in various text documents. Each triangle can then be represented as a vector. The three sides of a triangle are the three elements that make up the vector and their average is the center of the vector; centroid vector. Similar triangles can be merged when the similarity between corresponding centroid pair exceeds a given threshold. Figure 2.3.1 shows three documents, 17012, 17016 and 8907 represent a triangle. These three documents can be seen as mutually related as they all contain the word “nuclear”.

## **Problems with this representation**

This is a good representation for small sets of related documents. However, as the volume of documents increases, this representation adapts the same problem as cluster analysis, where more documents become less closer to a centroid assigned earlier and no new cluster is present to accommodate the changes.

### **2.3.2 Text Summarization with Lexical Chains**

The primary goal in (Silber and McCoy, 2000) is to create an efficient tool that should be able to summarize huge documents. A linear time algorithm is presented for calculating lexical cohesion among an arbitrary number of related words also known as lexical chains. The algorithm first creates a WorldNet lexical database as defined by (Miller, 1995). Three steps are involved:

(1) For each noun in the source document, all possible lexical chains are formed by looking up all relation information which includes, synonyms, hyponyms, hyponyms and siblings This information is then stored in an array indexed on the index position of the word from WorldNet for linear time retrieval.

(2) For each noun in the source document use the information collected in step 1 to insert the word in each meta chain. Each meta chain contains a score and a data structure.

The score is computed as each word is computed into the chain. Two words can be connected if they have the same semantics. The algorithm continues to find the best interpretation of the lexical chains.

The algorithm first creates the best set of graphs from the lexical chains. The algorithm then deletes nodes from each graph so that no two graphs share a node and the score of all the meta chains is maximal. In computation of the best chain, the algorithm carries out the following steps:

For each word in the document, for each chain that the word belongs to, (1) find the chain whose score will be affected most by removing this word from it. (2) Set the score component of this word in each of the chains to which it belongs to and update the score of

all chains to reflect the words removal. This is done in linear time since the interpretation of the text can be extracted without actually having to construct any interpretation. This is a big step to overcome the repetition in (Nomoto and Matsumoto, 2001). Lexical chains are used to detect correlation of noun phrase in a text document.

For example a text contains, “A friend just bought a new computer. The machine is a powerful and fast computer”. From the lexical dictionary, “friend”, “computer”, “machine” and “computer” are extracted as nouns. These four words are then represented by four dots. Lexical chains are then formed from these dots, one with the word “friend”, another with the word “machine” and a third with two dots and a chain connecting the two similar words “computer”→ “computer”. Suppose we pick the word “computer” and for each chain that has this word, we remove the word, it can be found out that the third chain will be the most affected by removing the word “computer” as the other two chains do not contain the word.

### **Problem with this approach**

This research did not give a clear scoring mechanism assigned to a lexical chain; they based the score on intuition.

### **2.3.3 Semantic Orientation**

A method for inferring the semantics of a word based on its statistical association was introduced by (Turney and Littman, 2003). Their focus was on identifying positive or negative measure of words, distinguishing antonyms from synonyms of a given word.

#### **Calculating Semantic orientation of a word**

Let  $P_{word}$  =set of words that are positively oriented and,  $N_{words}$ =set of words that are negatively oriented. Given two words word1 and word2,  $A(word1,word2)$  is defined as an association measure between the two words. This was referred to as Point Wise Mutual Information (PMI) in ( Chunks and Hanks, 1989).  $PMI(word1, word2) = \log(P(word1 \& word2) / P(word1) P(word2))$ , where  $P(word1\&word2)$  is the probability that word1 and word2 occur together. Having a document corpus, (Church and Hanks, 1990) takes the

count of word1 and word2, this is taken as the terms probability,  $p(\text{word1})$  and  $p(\text{word2})$  respectively.

The probability that word1 and word2 occur together,  $p(\text{word1}\&\text{word2})$ , is calculated by dividing text segments into small windows, then a count is done on the number of times word1 and word2 occur together in each window. The ratio of the two probabilities gives the degree of dependence between the two words. The log of this ratio gives the correlation measure. If the log is positive then the words tend to occur together, if it's negative then the presence of a word indicates the absence of the other word. The drawback in this approach is that the semantic orientation is very domain specific. For example, positively identified words in entertainment world may be seen as negative in health related issues.

String matching is also used in identifying semantic orientation of words (Hu and Liu, 2004). For example, assume there is a set of words labeled as positive words (pw);  $\text{pw} = \{\text{good, nice, excellent, positive, fortunate, correct, superior}\}$  and another set of words labeled as negative words (nw),  $\text{nw} = \{\text{bad, nasty, worst, poor, negative, unfortunate, wrong, inferior}\}$ . These two groups of words can be used to determine the semantic orientation of a sentence depending on a term presence in the sentence (Hu and Liu, 2004).

In reviewing customer opinions, an opinion sentence  $S = \text{"This is the worst camera I've ever bought, it has a nasty picture quality, I would recommend it to nobody"}$ , can be identified as either positive or negative using either the pw or nw term list. The presence of the terms "worst" and "nasty" in S makes S a negatively oriented sentence as the two terms are present in set nw. This approach suffers greatly when dealing with presence of two or more words in a review and these words are opposites of each other. This might lead to placing reviews in the wrong class.

#### **2.4 Association Rule Approach**

In traditional databases, the Apriori approach in (Agrawal and Srikant, 1994) was the foundation of many rule generation algorithms. It was originally designed for identifying

frequent patterns in structured data. Apriori requires several scans of the entire database thereby taking up much memory. This leads to much in efficiency. Improvements to the Apriori approach could lead to generation of frequent patterns in text documents (Holt and Chung, 1991, Zaiane and Antonia, 2002, Kongthon, 2004).

#### **2.4.1 Multi-pass Apriori (M-Apriori) and Multi-pass-Direct Hashing and Pruning (M-DHP).**

The problem of mining association rules from words in text documents was addressed by (Holt and Chung, 1999). Two algorithms were proposed, Multi-Pass Apriori (M-Apriori) based on the original Apriori in (Agrawal and Srikant, 1994) and the Multi-Pass Direct Hashing and Pruning (M-DHP) advancement on the Direct Hashing and Pruning (DHP) in (Park, 1997).

##### **(a) Apriori Algorithm**

This algorithm generates association rules among frequently occurring itemsets in any given relation database. Frequent itemsets are discovered in a level wise manner. If  $k$  itemsets are found to be frequent, then a self join is performed on this set to obtain a candidate set  $(k + 1)$  which is then evaluated to generate frequent  $(k+1)$  itemset. To improve the level-wise generation of frequent  $k$  itemsets, the apriori property is used to reduce the search space.

The Apriori Property states that “all non-empty subsets of a frequent itemset must also be a frequent itemset”, (Han and Kamber, 2001). For example, given a minimum support ( $ms$ ), if an itemset  $B$  does not satisfy  $ms$ , then if an item  $A$  is added to  $B$  forming a new set  $AB = \{\{B\},A\}$ , then the Apriori Property regards  $AB$  as an infrequent itemset as it does not satisfy  $ms$ , therefore it must be eliminated in creating a candidate set.

The Apriori algorithm starts by counting the frequency of each item in a transaction set, the resulting data forms candidate set 1,  $C_1$ . The transaction data in Table 2.4.1 generates the candidate set in table, 2.4.2. If the minimum support is given as 2, then all the itemsets in  $C_1$  are maintained and form the frequent 1 itemset,  $L_1$ , see Table 2.4.2.



TID	List of Item IDS
T100	111,112,115
T200	112,114
T300	112,113
T400	111,112,114
T500	111,113
T600	112,113,115

Table 2.4.1: A Sample Transaction Data Set

$C_1$  →

Itemset	Support Count
{111}	3
{112}	5
{113}	3
{114}	2
{115}	2

Table 2.4.2: Candidate 1 Itemset,  $C_1$

$L_1$  →

Itemset	Support count
{111}	3
{112}	5
{113}	3
{114}	2
{115}	2

Table 2.4.3: Frequent 1 Itemset,  $L_1$

To discover the frequent 2 itemsets, a join is performed on  $L_1$  with itself,  $L_1$  join  $L_1$ , this generates the candidate 2 itemset as shown in table Table 2.4.4. A scan is done on the transaction original data in Table 2.4.1, thereby counting the number of times the 2 items in each set occurs together. Those sets with support lower than minimum support are eliminated in generating frequent 2 itemset as shown in Table 2.4.5. The algorithm continues generating frequent sets until no more candidate sets are found.

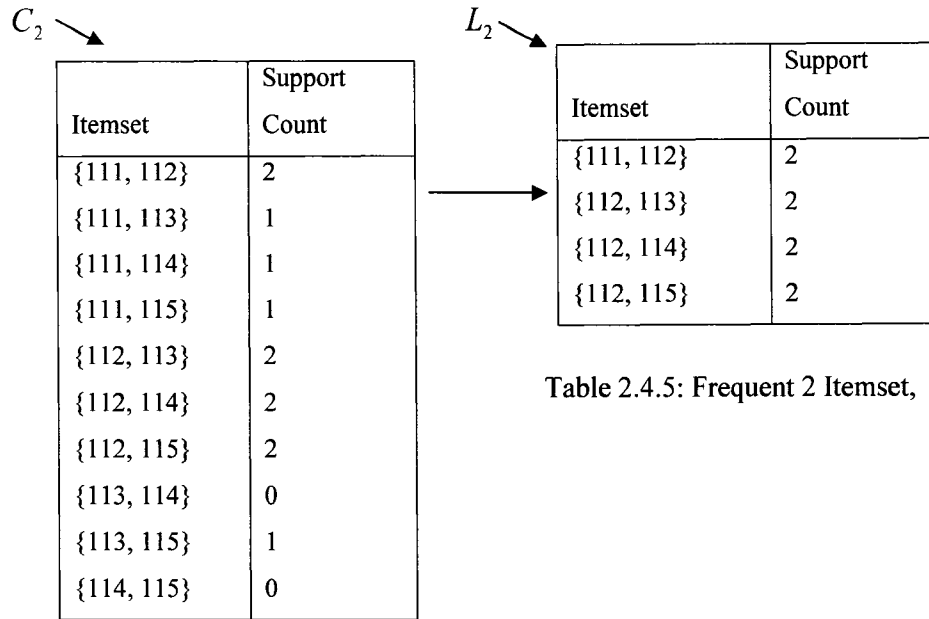


Table 2.4.4: Candidate 2 Itemset,  $C_2$

Table 2.4.5: Frequent 2 Itemset,  $L_2$

The resulting set of all frequent itemset is the union of each  $L_i$  sets,  $L = \{L_1, L_2, \dots, L_m\}$ , where  $m$  is the last database scan.

### (b) Direct Hashing and Pruning (DHP)

This is an advancement of the Apriori, a hashing technique for filtering out item-sets that may be unnecessary for the generation of the next  $(k+1)$  set of candidate item-sets.

#### Hashing Procedure

A hash table is created where each column in the table is a bucket slot, the aim is to count the entries in each bucket and eliminate those entries with support less than a given minimum support. To determine which bucket slot an itemset belongs to, a hash function  $h$  is defined as,  $h(x,y) = ((\text{order of } X) \times 10 + (\text{order } y)) \bmod n$ , where  $n$  is the number of bucket slots to be generated.

For example, a hash table for candidate 2 itemset in Table 2.4.4 is populated as follows: if  $(x,y) = (111,114)$ , then  $h(111,114) = ((111*10) + 114) \% 7 = 6$ ,  $h(111,113) = ((111*10) + 113) \% 7 = 5$ . Thus, itemset  $(111,114)$  is placed in bucket slot 5 and  $(111,113)$  is placed in slot 3. A count of all entries in each slot is taken and those slots with support less than 2 are eliminated. A database scan will only look for frequency of the remaining itemsets when generating frequent 2 itemset. This procedure continues until no more frequent candidate sets are generated.

Bucket Address	0	1	2	3	4	5	6
Bucket Count	1	1	2	1	1	2	2
Bucket Contents	{111,115}	{112,113}	{112,114} {113,115}	{112,115}	{111,112}	{111,113} {113,114}	{111,114} {113,115}

Table 2.5: A HashTable Representation of Candidate 2 Itemsets

Using the above Apriori and the Direct Hashing and Pruning algorithms, (Holt and Chung, 1999) proposed two algorithms for discovering association rules from text data. First text documents are broken down into individual words, stop words are removed then stemming using the Porter Stemmer in (Porter, 1984). The remaining set of words is ordered alphabetically. Partitions of the resulting ordered words are then generated; all words with a common start letter are placed in the same partition. Now, each partition is treated as a database and taken as input to the Apriori algorithm and the direct hashing and pruning algorithm. If there are  $n$  number of partitions generated, then  $n$  number of inputs are supplied to the two algorithms, the Multi-Pass-Apriori and Multipass Direct Hashing and Pruning.

A certain ordering is performed on how the partitions are supplied to the two algorithms, if  $n$  number of partitions is generated, then  $P_n$  is processed first followed by partition  $P_{n-1}$  and this continues up to partition  $P_1$ . An assumption is made that if items in  $P_n$  are

found to be frequent, then after processing  $P_{n-1}$ , the resulting frequent itemset is merged with those from  $P_n$  forming a new set that is assumed to be frequent.

### **Problem with this approach**

Partitions generated could contain so many words that might be irrelevant in the mining step as there is no special treatment given to terms appearing in strategic document positions, e.g. subheadings, titles or pronouns. Giving priority to such terms could eliminate redundant words leading to improvements of the mining algorithm.

### **2.4.2 Associating Terms with Text Categories**

The Apriori algorithm is also used in (Zaiane and Osmar, 2002). They use predefined sets of categories to identify keywords in documents. Two algorithms are proposed.

#### **Association-Rule Based Categorizer by Category (ARC-BC).**

Documents are represented as a transaction with several words as an itemset, Let  $D$  be a document, then  $D$  is assigned a distinct ID with several terms as an itemset,  $D = \{t_1, t_2, \dots, t_m\}$ ,  $m$  is the number of terms chosen to represent the document. A predefined set of categories  $C = \{C_1, C_2, \dots, C_n\}$ . Having documents in a set  $DB$ ,  $DB = \{D_1, D_2, \dots, D_n\}$ , the algorithm uses Apriori to assign categories to those documents. One category is passed to the entire set  $DB$  in each iteration of the Apriori. The aim is to develop rules of the form,  $t_1 \wedge t_2 \wedge t_3, \dots, \wedge t_n \rightarrow C_i$ . A document  $i$  that contains the terms in the rule is represented as  $D_i = \{C_i, t_1 \wedge t_2 \wedge t_3, \dots, \wedge t_n\}$ .

For example, given a predefined set of category terms  $C$ ,  $C = \{\text{health, cancer, diagnosis}\}$ . Each  $c_i$  in  $C$  is provided as input together with the set of transaction data set  $ts$ ,  $ts$  is the set of terms chosen to represent documents in transaction like manner. The goal is to generate frequent itemset that contains the category provided as input. If after a frequent itemset generation,  $L = \{\text{health, hospitals, medical, doctors, nurses, drugs}\}$ , since the term  $\text{health}$  is matched to category  $C_1$ , then a rule of the form  $\{\text{hospitals} \wedge \text{Medical} \wedge \text{doctors} \wedge \text{nurses} \wedge \text{drugs} \rightarrow \text{health}\}$  is created. A collection of all the rules generated is then termed as the classifier.

### **Association-Rule-base Categorizer for All Categories (ARC-AC)**

Instead of passing a category  $C_1$  alone at a time, all the categories are passed to the Apriori as a set plus the transaction data set. The algorithm iterates on all the categories and checks to find a match on terms in the category set from the frequent items generated. In this approach a document  $D_i$  can be represented by more than one category,

$D_i = \{C_1, C_2, \dots, C_m, t_1 \wedge t_2 \wedge t_3, \dots, \wedge t_n\}$ . For example, if a frequent I itemset contains {health, medical, insurance, hospital, drugs, nursing, homes}, if the category set C contains  $C_1, C_2$ ;  $C = \{\text{medical, insurance}\}$  then two rules are generated  $r_1, r_2$ ;

$$r_1 = \{\text{health} \wedge \text{insurance} \wedge \text{hospital} \wedge \text{drugs} \wedge \text{nursing} \wedge \text{home} \rightarrow \text{medical}\}.$$

$$r_2 = \{\text{health} \wedge \text{medical} \wedge \text{hospital} \wedge \text{drugs} \wedge \text{nursing} \wedge \text{home} \rightarrow \text{insurance}\}.$$

A model for document d1 is,  $d1 = \{c1, t1, t2, t3\}$ , each term in  $d_i$  is a term picked from the frequent itemset generated. For example, a document d1 is categorized as {health, hospitals, Medical, doctors, nurses, drugs}. A collection of the rules generated forms the classification model.

### **Document Classification.**

A new document is categorized using either of the two developed classification models, a document d is assigned to a category if the terms in a category rule are present in the d.

For example, a document  $d_1$  with several terms,  $d_1 = \{\text{medical, insurance, health, insurance, hospital, drugs, nursing, home}\}$  can be categorized as  $\{c_1, c_2, t_1, t_2, t_3, t_4, t_5, t_6\}$ .

If too many rules assigns a document to too many categories, a dominance factor is introduced, all categories are ordered according to how many number of rules have the category as the antecedent. The category with the highest rules is seen as the dominant factor and therefore it is assigned to the document being classified.

### **Problems with this approach**

These two algorithms suffer from several drawbacks; (1) the amounts of words that are used to represent the document in a transaction set is too huge, (2) for one to give appropriate predefined category terms, a lot of document context must be known, this is not feasible especially when dealing with huge volumes of textual data, and (3) if a predefined category is not found in a document, then a document might be assigned to the wrong category.

### **2.4.3 Discovering Technological Intelligence**

In her dissertation, (Kongthon, 2004), association rules were also in gathering related terms in text data. This was advancement in the Technologies Opportunities Analysis (TOA) development in Technology Policy Assessment at Georgia Institute of Technology, USA. She developed two algorithms the first algorithm is tree-like network capturing the important themes of a hierarchical structure, the second groups concepts together to form a thesaurus for data preprocessing.

## **Chapter 3: PROPOSED ALGORITHMS FOR TEXT INFORMATION MINING**

This chapter gives the details of the proposed algorithms for text information mining. The problem of linking together related documents is addressed, while picking the right terms to represent a document. The aim is to combine feature extraction mechanisms, summarization and association rules to generate a system that acts as a classifier for text documents. The thesis claim is that if key features are extracted from any given document based on their semantic orientation, they could act as a document representation otherwise known as summary. When a collection of such summaries is placed in partitions, mining algorithms can then be applied to the partitions thereby generating concept hierarchies of related documents. The concept hierarchies can then be used to classify a new document by identifying the level in the hierarchy that best categorizes the document.

As mention in section 2, there are various techniques for acquiring knowledge from text documents. The feature selection technique in (Chuang et. al, 2000) was able to extract certain sentence segments from text documents based on the occurrence of predefined bonus words in the sentences. This approach is limited to a certain domain of the supplied key words and does not scale well to documents with varying topics.

A multi strategy classification system in (Castillo and Serrano, 2004) was used to classify scientific and hypertext documents. Term frequency was the deciding factor on which terms to be extracted. Given predefined categories, a string match was then applied on the extracted words. If a match was found, then the document was assigned the matched category. This approach also does not scale well to varying topics.

A different approach is taken in (Hu and Liu, 2004), they take the semantic orientation of words found in a sentence then classify the sentences that are identified as semantically related. For example, if two customers, c1 and c2 have commented on a certain product,

adjectives in the review sentences are first identified with a part of speech tagger (Brill, 1992).

A predefined set of words is provided in (Hu and Liu), positive oriented (pw), and negative oriented (nw). If the adjectives in c1 and c2 appear in set pw, then the two customers have given positive reviews about the product otherwise the reviews are taken as negatively oriented. This approach suffers when adjectives that are semantically opposites appear in one sentence, then the reviews might be assigned to the wrong class.

As described in section 2.4.2 a, categorization scheme in (Zaiane and Antonie, 2002) gives a bigger range of categories for which to classify documents. Their approach uses association rules in (Agrawal and Srikant, 1994) to define categorization rules. However their approach suffers from the following drawbacks, (1) the approach considers only term frequency, (2) the number of words that are taken as document representation in a transaction set is too huge, (3) as predefined categories are still a deciding factor, if none of the terms in generated category rules exist in a document to be classified, then the document might be assigned the wrong class or not be classified at all.

This thesis proposes a semantic partition based document classification model. An observation has been made that if a lot of words in a text document tend to have the same or related meanings (semantics), it can be inferred that the words refer to the same thing. Words with related meanings can be grouped together forming semantic partitions. Semantic partitions formed from each document can act as the representation of the document in a transaction set and can act as input to a mining algorithm like Apriori, (Agrawal and Srikant, 1994). It can be assumed that words that tend to have no relation with any other words in a document do not convey meaning of a document and can be eliminated. This elimination can greatly reduce the amount of words used to represent a document.

Given a set of text documents, two algorithms for understanding the contents and relations in these documents have been proposed. The first algorithm will use Semantic Partitions,(SEM-P). The second algorithm will be an enhancement to the Semantic



Partitions, (ESEM-P). Using two thirds of a text collection, a classification model is generated using either SEM-P or ESEM-P. The remaining third of the text collection is to be classified using the developed model. The two algorithms differ in the amount of terms used to represent a document in a transaction dataset. Details of both algorithms are discussed below.

Figure 3.1 shows a sample text collection of 5 documents, {D1, D2, D3, D4, D5} that will be used to build a classification model.

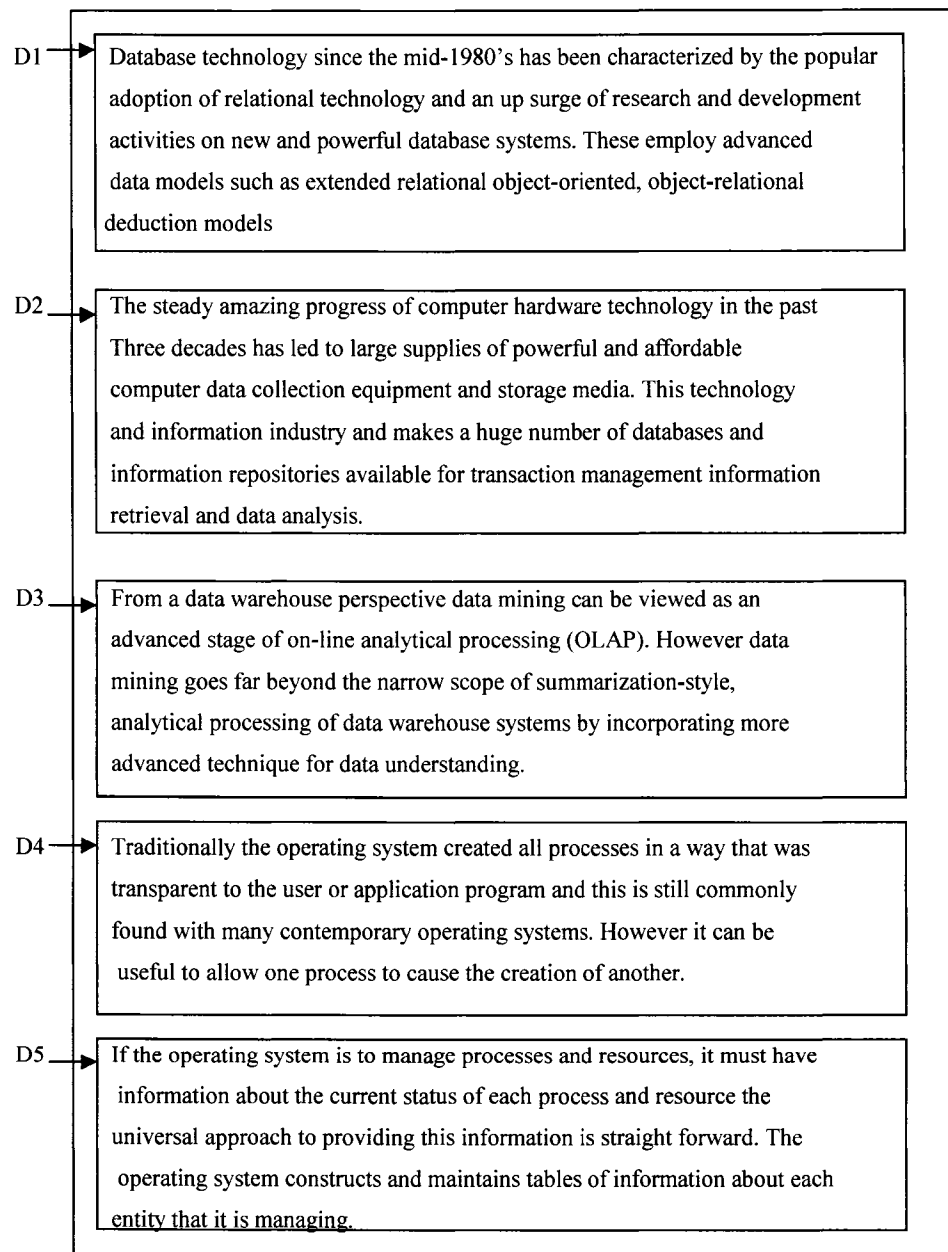


Figure 3.1: A Collection of 5 Documents {D1, D2, D3, D4, D5}.

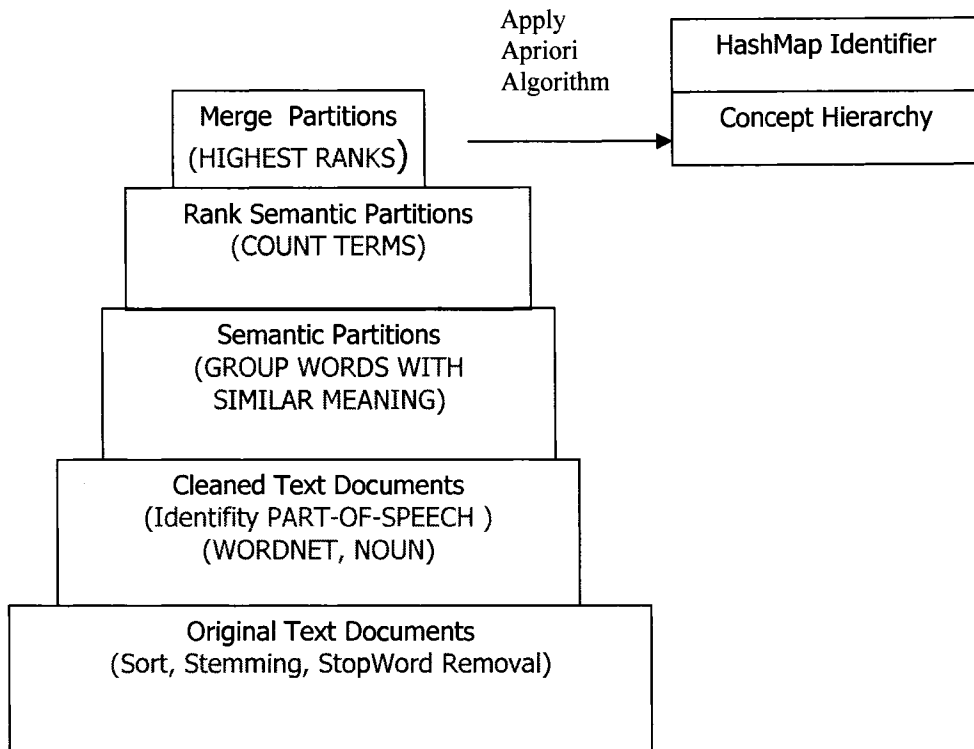


Figure 3.1.1: Overall process of text information mining

### 3.1 Description of Semantic Partition (SEM-P)

The main algorithm is divided into six main modules. These modules are presently as separate algorithms in the following few pages.

#### 3.1.1 Semantic-Partition Algorithm (SEM-P)

#### Algorithm 3.1 SEM\_P: Semantic Partitions

```
■ //An algorithm to generate a classification model for text documents
■ Input: A set T containing n documents
■ Output: A concept Hierarchy (CH) and a HashMap Identifier (HI)
Begin
1. For each Document d in set T do:

    1.1 Extract tokens  $k^d$  with Algorithm Feature Extraction
         $k^d = \text{FeatureExtraction}(d)$  (presented in Figure 3.1.3)
    1.2 Find semantically related tokens in  $k^d$  as semantic partitions, SEM_P
        SEM-P = GenerateSemanticPartitions( $k^d$ ) (presented in Figure 3.1.3.2).
    1.3 For each semantic partitions s in SEM-P assign a rank using number of tokens
        RSEM-P = RankSemanticPartition(SEM-P) (presented in Figure 3.1.4.1).
    1.4 Merge all the semantic partitions in RSEM-P depending on their ranks
        MSEM-P = MergeSemanticPartitions(RSEM-P) (presented in Figure 3.1.4.2a
        and Figure 3.1.4.2.b)
2. Generate a concept hierarchy (text file) of frequent items by applying the Apriori Algorithm
    CH = GenerateFrequentPatterns(MSEM-P) (presented in Figure 3.1.5.1)
3. Create a table with 0/1 as entries from the CH levels, this is the HashMapIdentifier table
    HMI = GenerateHashMap Identifier(CH) (presented in Figure 3.1.5.2)

End
```

Figure 3.1.2 The Semantic Partition Algorithm, (SEM\_P)

#### Step 1: Feature Extraction

Each document is broken down into single words. These words are then passed to a stop word removal method where a list containing popular words like {the, a, an, of, etc.} is used. If any of the words in the list are found in the document, then these are eliminated. Lexical sorting of the remaining words will be done to enhance the stemming step since terms with common prefix will be close together after sorting.

## What is Stemming?

The presence of words that have a common prefix but different suffices only add up to the massive amount of data to be processed. For example, set  $S = \{\text{consider, considered, consideration, considering}\}$  has words that share the common prefix “consider”, when the stemming algorithm is applied to  $S$ , all the suffixes “ered”, ”ation” and “ing” will be removed leaving the common prefix “consider”, only one copy of this word will be kept. The popular Porter’s Stemming algorithm in (Rijsbergen et. al., 1980) will be used.

After preprocessing a document as described above, the documents in Figure 3.1.1 are represented as shown in Figure 3.1.3.1. Note that a comma is used for visualization purposes only and is not present during actual feature extraction stage. The FeatureExtraction algorithm is shown in Figure 3.1.3.

### Algorithm 3.1.1 (Feature Extraction)

```
Algorithm FeatureExtraction( )  
Input: A set of Text Documents  $T$   
Output: A set of Cleaned Text Documents  $T_C$   
Variables: prefix_List // to hold the tokens with similar prefix  
word_prefix // to hold the prefix of a particular word  
Begin  
  (1). for each text Document  $d$  in Document set  $T$ , do  
    extract all tokens  $t\{t_1, t_2, \dots, t_m\}$  from document  $d$  and store them in set  $T_C$   
  (2) Sort all the tokens in set  $T_C$  in ascending order.  
  //stop word removal  
  (3). For each token  $t$  in sorted list  $T_C$ , do  
    if token  $t$  is found in stop_word_list, then remove  $t$  from list  $T_C$   
  //stemming  
  (3). For each token  $t$  in the sorted list of tokens  $T_C$ , do  
    (3.1). Get the prefix of a token  $t$  and store it in word_prefix.  
    (3.2). Identify all tokens with prefix similar to word_prefix and store these  
    in Prefix_list. /*Prefix_list contains tokens  $\{t_1, t_2, \dots, t_k\}$  with similar prefix*/.  
    (3.3). Remove all tokens  $t_2$  to  $t_k$  from sorted list  $T_C$ , token  $t_1$  is left to  
    represent the entire prefix_list in the sorted list  $T_C$ .  
end
```

Figure 3.1.3: Feature Extraction Algorithm

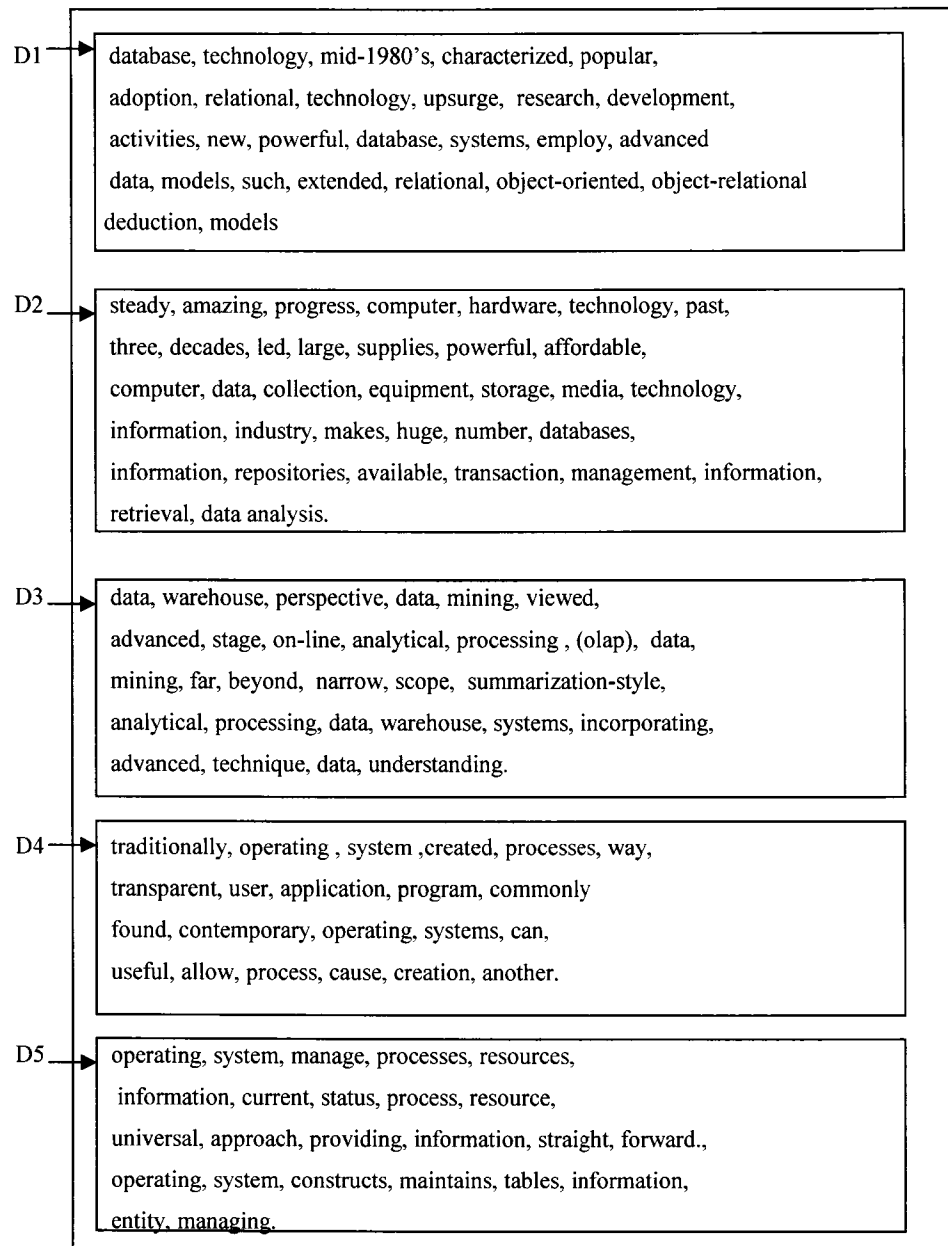


Figure 3.1.3.1: Documents D1 to D5 after Feature Extraction

## **Step 2: Semantic Orientation and Partitioning**

In order to get the semantic meaning of words, two mechanisms can be applied. One can create an ontology of words where related words are assigned the same class, this approach was taken in (Sakurai and Suyama, 2004), however, this approach is domain specific and one need to do several updates to the ontology to accommodate new terms. To avoid this problem, an online lexical ontology, WordNet in (Miller, 1995) will be used.

### **Why Semantic Orientation?**

Most words in English vocabulary can be referring to the same entity but spelled differently, for example “caretaker” and “janitor”, these two words refer to the same kind of occupation, using WordNet, they are placed in the same word group. In text information mining, given any domain, one would like to group words that are similar in meanings together. Suppose a health document collection is to be examined, then all words related to a particular topic i.e., diseases should be in one group.

### **How is Semantic Orientation identified?**

In WordNet dictionary, words are arranged according to their part of speech tags (Brill, 1992). All nouns are stored together with their meanings, these are referred to as senses. For example to get the senses of a word ”computer”, one has to pass a query to the WordNet dictionary with the word ‘computer” as the search key. If the word is found, then WordNet returns all the stored meanings (senses) of the word.

Unlike the semantic orientation in (Hu and Liu) where adjectives are considered in grouping related sentences together, in this thesis only nouns will be considered in identifying semantics of a document. First, a word is tagged using the part of speech tagger described in (Brill, 1992). If a word is identified as a noun, then the word will be passed to WordNet as the search key. WordNet retrieves all the stored senses of the word. The senses are words that give the search key a meaning. The idea is to do a string match of all the words contained in the senses with the words in a document.

If any word  $w$  in the senses is found in the document then we remove  $w$  from the document and place it together with the search key in a new data structure. This newly created data structure is the semantic partition (SEM-P) and all words that are related in meaning will be in the same partition. Each document will have its own semantic partitions. For example in Figure 3.1.4, eight Semantic Partitions,  $SEM-P = \{SEM-P1, SEM-P2, SEM-P3, SEM-P4, SEM-P5, SEM-P6, SEM-P7, SEM-P8\}$  for document D1 are generated as follows;

Starting from the beginning of the D1 the search key  $sk$  is taken in sequence, if  $sk =$  “database”, then a call to WordNet is done with  $sk$  as the parameter to search for. WordNet returns 1 sense  $s$  with a description of the passed word,  $s =$  “organized body of related information”. After tokenizing and removal of stop words from  $s$ , we have a token set  $s'$ ,  $s' = \{organized, body, related, information\}$ . Now do a search for every word  $w$  in  $s'$  in the entire document D1. The word “relational” is found in D1, note that “related” and “relational” share the same prefix “relate” so they are taken as the same word. Now the first Semantic partition of D1,  $SEM-P1 = \{database, relational\}$ .

The same procedure continues for all words in D1 until no more semantic partitions can be generated. The final semantic partitions for D1 are;  $SEM-P1 = \{database, technology, relational, research, systems, data, models, object-oriented, object-relational\}$ ,  $SEM-P2 = \{mid-1980s\}$ ,  $SEM-P3 = \{employ\}$ ,  $SEM-P4 = \{development\}$ ,  $SEM-P5 = \{characterized\}$ ,  $SEM-P6 = \{popular, powerful\}$ ,  $SEM-P7 = \{advanced\}$ ,  $SEM-P8 = \{activities\}$ . Figure 3.1.4 shows the semantic partitions for all the original documents in Figure 3.1.1. The GenerateSemanticPartition algorithm is shown in Figure 3.1.3.2.

### Algorithm 3.2 : (Generate Semantic Partitions)

**Algorithm** GenerateSemanticPartitions( );

**Input:** Set of Cleaned Text Document,  $T_C$

**Output:** Semantic Partitions, SEM-P for each  $T_{C_i}$  in  $T_C$

**Variables:**

- set containing partitions of semantically related words,  
semantic partitions, SEM\_P.

- Part\_of\_speech\_value and part of speed for any given word.
- word\_senses //this holds the meanings for a word as retrieved though wordnet
- word\_senses' holds the word\_senses after stopword removal

**Begin:**

- (1). For each document d in set of cleaned documents  $T_C$ ,
  - (1.2). for each word w in document d, do
    - identify the part\_of\_speech\_value for word w
    - (1.2.1)if( part\_of\_speech\_value is NOUN) then,
      - extract the senses of word w from WORDNET and store these in word\_senses list.
      - (1.2.1a) for each word w1 in word\_senses list, do
        - if word w1 is in stop\_word\_list remove word w1 from word\_senses list
      - (1.2.1b) for each word w2 in cleaned document d , do
        - identify all the words w from cleaned document d that are found in word\_sense list and group these words together. /\* these words are seen as having similar meanings, semantically related. These groups of semantically related words are called semantic partitions, SEM\_P.\*/

**End**

Figure 3.1.3.2: The Generate Semantic Partition Algorithm



D1	Semantically Related Terms
SEM-P1	Database,technology,relational Research,systems,data,models,object-oriented,object-relational
SEM-P2	Mid-1980's
SEM-P3	Employ
SEM-P4	Development
SEM-P5	Characterize
SEM-P6	Popular, powerful
SEM-P7	Advance
SEM-P7	Activity

D2	Semantically Related Terms
SEM-P1	Steady
SEM-P2	Amazing
SEM-P3	Progress
SEM-P4	Computer,hardware,technology,data, equipment,storage,media,information, repositories, transactions, retrieval, analysis
SEM-P5	Management
SEM-P6	Decades
SEM-P7	Supplies
SEM-P8	Powerful, great

D3	Semantically Related Terms
SEM-P1	Data, warehouse, mining, analytical, processing, summarization, systems
SEM-P2	advanced
SEM-P3	techniques
SEM-P4	understanding

D4	Semantically Related Terms
SEM-P1	Operating,processes,program
SEM-P2	traditionally
SEM-P3	created
SEM-P4	still
SEM-P5	commonly
SEM-P6	found
SEM-P7	Useful

D5	Semantically Related Terms
SEM-P1	Operating, system, processes, information
SEM-P2	manage
SEM-P3	resources
SEM-P4	current
SEM-P5	status
SEM-P6	universal
SEM-P7	approach

Figure 3.1.4: Semantic Partitions on words in Documents D1, D2, D3, D4 and D5

### Step 3: Semantic Partition Ranking

The number of terms in each partition is then recorded to give priority to longer partitions. If a document tends to have a lot of words that are semantically related, then it can be said that these words can be a representation of a document context. The count of terms in a partition is also taken as the rank of the partition. For example, the semantic partitions for document D1 have the following term count, SEM-P1 = 9, SEM-P6 = 2 and SEM-P2, SEM-P3, SEM-P4, SEM-P5, SEM-P7 each has a count of 1. Therefore, D1 is then seen as having three ranks, with SEM-P1 as the highest ranked partition, therefore it can be taken as a representation for D1. Similar ranking is done for all the other documents as seen in Table 3.1.5. The RankSemanticPartition algorithm is shown in Figure 3.1.4.1.

Di	Rank1	Rank 2	Rank 3
D1	SEM-P1  = 9	SEM-P6  = 2	SEM-P2 ,  SEM-P3 ,  SEM-P4 ,  SEM-P5 ,  SEM-P7 ,  SEM-P8  = 1
D2	SEM-P4  = 11	SEM-P8  = 2	SEM-P1 , SEM-P2 , SEM-P3 , SEM-P5 , SEM-P7  = 1
D3	SEM-P1  = 7	SEM-P2 ,  SEM-P3 , SEM-P4  = 1	
D4	SEM-P1  = 3	SEM-P2 , SEM-P3 , SEM-P4 , SEM-P5 , SEM-P6  ,  SEM-P7 = 1	
D5	SEM-P1  = 4	SEM-P2 , SEM-P3 , SEM-P4 , SEM-P5 , SEM-P6  ,  SEM-P7 = 1	

Table 3.1.1: A Ranking of Semantic Partitions from Figure 3.1.4.

### Algorithm 3.3: Ranking Semantic Partitions( )

<p><b>Algorithm RankSemanticPartition(SEM-P);</b>  <b>Input:</b> Semantic Partitions SEM-P for each Document T  <b>Output:</b> Ranked Semantic Partitions (RSEM-P)  <b>Variable:</b> partition_rank /*number of terms in each semantic partition, SEM – P  <b>Begin</b>              For each set of semantic partition, SEM_P do                  count the number of terms in a partition, SEM_P and save it                  in partition_rank.              return(partition_rank);  <b>end.</b></p>
--

Figure 3.1.4.1: Ranking Semantic Partition Algorithm

#### Step 4: Merging Partitions

The highest ranked partition in each document are then placed in one group, the second highest in another group and so on until all ranks are grouped resulting to frequency partitions. If a collection D has several documents,  $D = \{D_1, D_2, \dots, D_n\}$ , each  $D_i$  in D has a highest rank semantic partition  $j$ ,  $D_i\text{SEM-}P_j$ . Let  $D_1$  and  $D_2$  be elements in D,  $D_1$  and  $D_2$  has highest ranked semantic partitions,  $D_1\text{SEM-}P_i$ ,  $D_2\text{SEM-}P_i$ , and second highest partitions  $D_1\text{SEM-}P_j$ ,  $D_2\text{SEM-}P_j$ . Several groups can then be created from these ranks; group1 =  $\{D_1\text{SEM-}P_i, D_2\text{SEM-}P_i\}$ , group 2 =  $\{ D_1\text{SEM-}P_j, D_2\text{SEM-}P_j\}$ , ... group n =  $\{ D_1\text{SEM-}P_k, D_2\text{SEM-}P_k\}$ , where n is the number of ranks and k is the number of semantic partitions for each document.

For example, using the ranked semantic partitions generated in Table 3.1.1 the semantic partitions in each rank are merged to form one group as shown in Table 3.1.2. Group 1 in Table 3.1.5 contains SEM-P1 from D1, SEM-P4 from D2, SEM-P1 from D3, SEM-P1 from D4 and SEM-P1 from D5.

Groups	Semantic Partitions Representing each Document
Group 1	D1SEM-P1, D2SEM-P4, D3SEM-P1, D4SEM-P1, D5SEM-P1
Group 2	D1SEM-P6, D2SEM-P8
Group 3	D1SEM-P2, D1SEM-P3, D1SEM-P4, D1SEM-P5, D1SEM-P7, D1SEM-P8 D2SEM-P1, D2SEM-P2, D2SEM-P3, D2SEM-P5, D2SEM-P7 D3SEM-P2, D3SEM-P3, D3SEM-P4 D4SEM-P2, D4SEM-P3, D4SEM-P4, D4SEM-P5, D4SEM-P7, D4SEM-P8, D5SEM-P2, D5SEM-P3, D5SEM-P4, D5SEM-P5, D5SEM-P7,

Table 3.1.2: Groups Formed from Ranked Semantic Partitions

#### (i) Partition Pruning

Merging low ranking partitions from different groups can sometimes generate a huge volume of terms that is not seen as relevant to a document context. For example, as seen if Table 3.1.1, group 3 contains all those partitions that were having a rank = 1. The terms in these partitions were seen as having little relevance to the main documents context, therefore they can be eliminated. Having a predefined threshold  $t$ , all those groups containing elements with ranks less  $t$  are therefore pruned out. From Table 3.1.2, if  $t = 2$

then Group 2 and Group 3 are pruned out leaving only Group1 for further analysis as seen in Figure 3.1.4.2.

Document ID	Terms form Highest Ranked Semantics
D1	Database,technology,relational,Research,systems,data,models,object-oriented,object-relational
D2	Computer,hardware,technology,data,equipment,storage,media,information,repositories,transactions, retrieval, analysis
D3	Data, warehouse, mining, analytical, processing, summarization, systems
D4	Operating,processes,program
D5	Operating, system, processes, information

Figure 3.1.4.2: Representation of Group 1 from Table 3.1.2

### Algorithm 3.4(a): (Merging Semantic Partitions)

```

Algorithm 3.4(a): MergeSemanticPartitions(RSEM-P, Rank)
Input: Ranked Semantic Partitions, RSEM-P for every Processed Text Document in T, Rank
Output: Merged Semantic Partitions , MSEM-P
variable: 1. semantic_partition_count - /*This will hold the semantic partitions
with rank greater than 2. */
2. Merge_Semantic_partition_list, MSEM_P. /* to hold merged semantic partitions */
Begin
1. For every set of ranked semantic partition, RSEM_P, do
1.1 Get the highest ranked semantic partitions and put them in a set of merged
semantic partitions, MSEM_P. /* MSEM_P hold all the highest ranked semantic
partitions for all the text documents processed*/
2. Merge the remaining semantic partitions according to their ranks
2.1 Pruning of all partitions with ranks less than a given threshold.
/*Given a threshold of 2, then all semantic partitions with rank < 2 are
eliminated.*/
End

```

Figure 3.1.4.2a: The Merging Partition Algorithm

#### (ii) Merging Partitions Depending on Semantic Partition Distribution

As further evaluation of various text documents has been done, several semantic partitions may be generated whereby each partition contains no more than two elements. In this situation a different kind of merging has been implemented. The goal is to completely represent any given text document with any kind of semantic partition

distribution. First a count  $c$  of how many semantic partitions contain more than two elements is done.

Given a total of  $p$  semantic partitions for each text document evaluated, count  $c$  can fall into three cases. Case 1, count  $c$  is greater than half of  $p$ , ( $c > \frac{1}{2}p$ ) This means the documents have a lot of terms with similar meanings. In this case, all the semantic partitions with rank greater than two are merged together. Case 2, count  $c$  is less or equal to half of  $p$  but greater than a quarter of  $p$ , ( $\frac{1}{4}p \leq c \leq \frac{1}{2}p$ ). In this case all the semantic partitions with rank greater than two are merged together and a random selection of half of the remaining semantic partitions is added to the merged partitions. Case 3, count  $c$  is much less than a quarter of  $p$ , ( $c < \frac{1}{4}p$ ). This indicates that the document has very few words with similar meanings. In this case all the semantic partitions with rank greater two are merged together including a random selection of three quarters of the remaining semantic partitions. The improved MergeSemanticPartition algorithm is shown in Figure 3.1.4.2b.

### Algorithm 3.4(b): (Merging Semantic Partitions)

```
Algorithm 3.4(b): MergeSemanticPartitions(RSEM-P, Rank)  
Input: Ranked Semantic Partitions, RSEM-P for every Processed Text Document in T, Rank  
Output: Merged Semantic Partitions , MSEM-P  
variable: 1. semantic_partition_count /*This will hold the number of semantic partitions  
with rank greater than 2. */  
2. Merge_Semantic_partition_list, MSEM_P. /* to hold merged semantic partitions */  
Begin  
1. For every ranked semantic partition in set RSEM-P, do  
1.1 if ( rank_for_semantic_partition is greater than 2)  
increment the value of semantic_partition_count /*this value is  
used to identify the distribution of the generated semantic partition*/  
2. /*semantic_partition_count can have 3 range of values as described in the  
following three cases*/  
Case 1: /* semantic_partition_count has a value more than half of the overall  
semantic partitions. This is seen as a normal distribution, implying that a  
document is well represented by the generated semantic partitions. */  
Merge all the semantic partitions with rank greater than 2 and store these  
partitions into merge_semantic_partition list, MSEM_P.  
  
Case 2: /*Semantic_partitions_count has a value less than half but greater than a  
quarter of the overall semantic partitions. This is seen as a skewed distribution  
implying that a document has few words related in meanings. */  
Merge all the semantic partition with rank greater than two and randomly select  
partitions from the rest of the remaining 1/2 of semantic partitions. Store all the  
selected semantic partitions into list, MSEM-P.  
  
Case 3: /*semantic_partitions_count has a value less than a quarter of the overall  
semantic_partitions. In this case the document is seen as having very little  
or no semantically related words. However, we must still select words to  
represent this document. */  
Merge all the semantic partitions with elements more than 2 and randomly select  
partitions from the remaining 3/4 of the semantic partitions. Store these merged  
partitions into list, MSEM_P.  
  
return (merged_semantic_partition list, MSEM_P);  
end
```

Figure 3.1.4.2b: The Merging Partition Algorithm

### Step 5: Mining Partition

Each of the resulting groups in step 4 is seen as a transaction set and can be processed using data mining algorithms, the Apriori algorithm (Agarwal and Srikant, 2000) will be used to identify frequent terms in each group. Each semantic partition in the group is taken as a transaction representing its original document in the transaction set. A minimum support of 2 is given to mine frequent patterns in the transaction set shown in Table 3.1.3.

Candidate 1 Items	Support Count	Candidate 1 Items	Support Count	Items (Terms)	Support Count
Database	1	Equipment	1	Technology	2
Technology	2	Storage	1	Systems	3
Relational	1	Media	1	Data	3
Research	1	Information	2	Information	2
Systems	3	Transaction	1	Analysis	2
Data	3	Management	1	Processing	3
Models	1	Analysis	2	operating	2
Object-oriented	1	Warehouse	1		
Object-relational	1	Processing	3		
Computer	1	Summarization	1		
Hardware	1	Operating program	2 1		

Table 3.1.3: Candidate 1 itemset,  $C_1$

Table 3.1.4 Frequent 1 itemset,  $L_1$

Candidate 2 Itemsets	Support count
technology, systems	1
technology, data	2
technology, information	1
technology, analysis	1
technology, processing	0
technology, operating	0
systems, data	1
systems, information	1
systems, analysis	0
systems, processing	3
systems, operating	2
data, information	1
data, analysis	1
data, processing	1
data, operating	0
information, analysis	1
information, processing	1
information, operating	1
analysis, processing	0
analysis, operating	0
processing, operating	2

Table 3.1.5: Candidate 2 itemset,  $C_2$

$L_2$  →

Frequent 2 itemsets	Support Count
Technology, data	2
Systems, procesing	3
System, operating	2
Processing, operating	2

Table 3.1.6: Frequent 2 Itemset,  $L_2$  in  $C_2$

$C_3$  →

Candidate 3 Itemsets	Support Count
Technology, data, system	1
Technology, data, processing	0
Technology, data, operating	0
System, processing, operating	0

Table 3.1.7: Candidate 3 Itemset,  $C_3$



None of the itemsets in Table 3.1.7,  $C_3$ , has the desired minimum support therefore the Apriori algorithm terminates. There are two frequent item sets generated from the transaction set in Table 3.1.4,  $L_1$  and  $L_2$  as shown in Table 3.1.6.

**Step 6: Generating Concept Hierarchies and HashMap Identifiers**

All the frequent identified terms will be placed in a concept hierarchy where the highest scoring term is the root or hierarchy header. Scoring is calculated depending on the support of a term in the transaction set. Support (item  $i$  in  $L_i$ ) = Count of item  $i$  in the transactions/ Total number of transactions. For example, the support for all items in  $L_1$  is done as follows; Support(technology) =  $2/5 = 0.4 \times 100 = 40\%$ , Support(data) =  $3/5 = 0.6 \times 100 = 60\%$ , Support(system) =  $4/5 = 0.8 \times 100 = 80\%$ , Support(processing) =  $3/5 = 0.6 \times 100 = 60\%$ , Support(technology) =  $2/5 = 0.4 \times 100 = 40\%$ .

Building the concept hierarchy is done by ordering the support count in levels, for example, there are three levels using the support count calculated for items in  $L_1$ , {80%, 60% 40%}. The final concept hierarchy is shown in Figure 3.1.5.

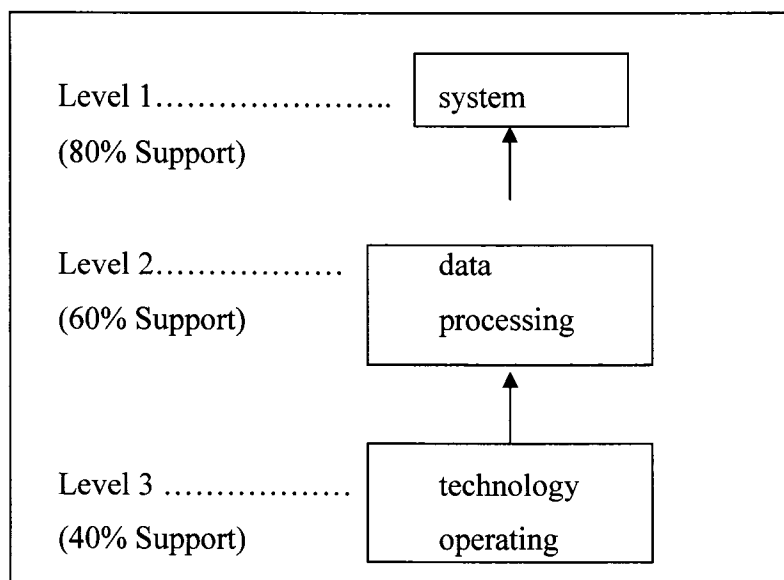


Figure 3.1.5: A Concept Hierarchy for Frequent 1 items,  $L_1$

The algorithm for generating Concept Hierarchy is shown in Figure 3.1.5.1.

### Algorithm 3.5: (Generate ConceptHierarchy)

```
Algorithm 3.5: GenerateConceptHierarchy(MSEM-P)  
Input: Merged Semantic Partitions (MSEM-P)  
Output: Concept Hierarchy CH  
Variables: MinSup//minimum support  
              set L of frequent patterns  
              set Level to hold terms with similar support  
Begin  
1. Generate sets of frequent patterns, L by passing the set of merged semantic partitions,  
   MSEM_P into the Apriori algorithm./* A call to Apriori contains the semantic partitions  
   and a minimum support, MinSup; Apriori (MSEM_P, MinSup)*/  
2. For each set Li of identified frequent patterns in set L, do  
   2a. Sort frequent set  $L_i$  according to their support;  
   2b. Group all items in set Li with similar support starting from highest support to the  
       lowest. /*This grouping forms the levels known as concept hierarchy, CH*/  
  
      return (concept hierarchy, CH);  
End
```

Figure 3.1.5.1: The algorithm to Generate Concept Hierarchies

A last scan of the database will be done, this scan will be used to populate a binary HashMap which will be used to identify the documents that contains the words in the concept hierarchy. This portion is lacking in data mining algorithms, normally after frequent terms are generated, there is no data structures that links the rules generate to the original database. This is very important in text information mining, since only a portion of a text document is used in identifying frequent patterns; one would like to be able to track down represented documents.

## Step 6: Generate HashMap Identifier

**Algorithm 3.6 GenerateHashMapIdentifier()**  
**Input:** The concept hierarchy CH with different levels L  
**Output:** A HashMapIdentifier, HMI  
**Variables:** Table HashMapIdentifier /\* all entries initialized to 0\*/  
**Begin**

1. For every Level L in the concept hierarchy CH,
  - 1a. For every word w in Level L
  - 1b. For every token t in cleaned text document ,TC  
 If (word w in Level L is similar to token t)
    - 1c. Update the entry for word w in HashMapIdentifier table from 0 to 1

**End**

Figure 3.1.5.2: The algorithm for Generating HashMap Identifier

Every column in the Binary HashTable will represent a unique document, see Table 3.1.8, every row represents a term in the concept hierarchy. Each entry  $ij$  represents the presence of concept  $i$  in document  $j$ , that is if term  $i$  is in  $j$  then the  $\{ij\} = 1$  otherwise  $= 0$ . A tally of all elements in a column  $j$  with 1 entry represents the total number of concepts in the hierarchy that are in document  $d_j$ , columnsum. A tally of all elements in a row that have a 1 entry represents the total number of documents that a level represents, rowsum.

Levels	D1	D2	D3	D4	D5	RowSum
Level 1	1	0	1	1	1	4
Level 2	1	1	1	1	1	5
Level 3	1	1	0	1	1	4
ColumnSum	3	2	2	3	3	

Table 3.1.8: A HashMap Binary Identifier for the Concept Hierarchy in Figure 3.1.4.2

### Classifying a New Document with the Generated Concept Hierarchy

The concept hierarchy in Figure 3.1.5 is the main model for classifying a new document. First, a document to be classified goes through the same stages of identifying the semantics partitions. Only the highest ranked semantic partitions are retained, these will be the deciding factor on which class a document belongs to. For example Figure 3.1.6 contains a document D6 that we wish to classify. After preprocessing and identifying the

semantics of the words in D6, semantic partitions are formed as shown in Table 3.1.9. The highest ranked partition is SEM-P8 = {processes, application, programs, language, operating, system}. Now a string match is done on the terms in the concept hierarchy in Figure 3.1.6 with the words in SEM-P8. It can be observed that SEM-P8 is represented in every level of the concept hierarchy since words in SEM-P8 are present in levels 1, 2 and 3 in Figure. It can be inferred that the concept hierarchy in Figure 3.1.6 is a good representation for D6.

There are a number of ways in which the requirements for mutual exclusion can be satisfied. One way is to leave the responsibility with the process that wish to execute concurrently. Thus processes, whether they are system programs or application programs would be required to coordinate with one another to enforce mutual exclusion with no support from the programming language or the operating system.

Figure 3.1.6: Document D6 to be classified



Semantic Partitions		Semantic Partitions	
SEM-P1	number		
SEM-P2	ways	SEM-P9	execute
SEM-P3	requirement	SEM-P10	required
SEM-P4	mutual	SEM-P11	coordinate
SEM-P5	exclusion	SEM-P12	another
SEM-P6	satisfied	SEM-P13	enforce
SEM-P7	responsibility	SEM-P14	Support
SEM-P8	processes, application, programs, language, operating, system.		

Table 3.1.9: The Identified Semantic Partitions for Document D6

A modification of the HashMap in Table 3.1.8 is done as a new document, D6 is to be added to the hashMap. The resulting updated HashMap is shown in Table 3.1.10.

Levels	D1	D2	D3	D4	D5	D6	RowSum
Level 1	1	0	1	1	1	1	5
Level 2	1	1	1	1	1	1	6
Level 3	1	1	0	1	1	1	5
ColumnSum	3	2	2	3	3	3	

Table 3.1.10: An Updated HashMap after Adding Document D6.

### 3.2 Description of Enhanced Semantic Partitions (ESEM-P)

With a huge volume of text documents to be processed where the average number of words in each document could grow to thousands, this would also generate a huge amount of terms in each semantic partition. To reduce this huge volume of terms, weighting and ranking mechanism will be used for each term in a semantic partition.

Four measures will be used to rank each term in a semantic partition; these measures will be represented in a vector, a semantic partition SEM-P will be represented as; SEM-P =  $\{t_1 \langle m_1, m_2, m_3, m_4 \rangle, t_2 \langle m_1, m_2, m_3, m_4 \rangle, \dots, t_n \langle m_1, m_2, m_3, m_4 \rangle\}$ , where n is the number terms in the partition. An entry  $m_i$  in a vector represents the value of measure 1.

Measures to Evaluate Each Term:

1. Term Frequency in the Document:

The number of times a term occurs in the document will be recorded in m1.

2. Term Frequency in entire Document set:

The number of times a term occurs in the entire document will be recorded in m2, excluding the count of the term in the document being processed.

3. Proper Noun Terms:

If a term begins with a capital letter and its not beginning a sentence will be considered and recorded in m3.

4. Position of a Term:

Terms that begin a document, a topic or a subtopic will be recorded in m5. A three term window is used to determine if a term begins a document. This will give preference to compound words, for example, if “Database technology” begins a document the term technology is also given the same weight as the term database.

Each of the measures will then be multiplied by 10, this will provide a huge margin between terms thereby giving a clear elimination strategy. If Cut-off threshold (COT) will then be provided, all term that do not meet the COT will be eliminated from their semantic partitions.

The algorithm has eight modules just like the Semantic partition algorithm however; Module 5 of SEM-P algorithm is enhanced using a vector measure for each term in a partition. All the highest ranked partitions in Figure 3.1.5 are represented as follows;

D1: {Database <2x10,1x10,0,1x10>, technology<2x10,2x10,0,1x10>, relational<2x10,0,0,0>, Research<1x10,0,0,0>, systems<1x10,5x10,0,0>, data<1x10,6x10,0,0>, models<1x10,0,0,0>, object-oriented<1x10,0,0,0>, object-relational<1x10,0,0,0>. After a sum of each term vector, D1={database<40>, technology<50>, relational<20>, research<10>, systems<60>, data<70>, models<10>, object-oriented<10>, object-oriented<10>}. If the cut-off threshold is given as 20, then all the terms that are less than the cut-off threshold are eliminated. Now D1 = {database, technology, relational, systems, data}. This same procedure is done for the documents d2, d3, d4 and d5 and results are shown in Table 3.1.9.

Documents	Terms Left form Each Document after Pruning
D1	Database, technology, relational, systems, data
D2	Computer, technology, data, information, analysis
D3	Data, mining, analytical, processing, systems
D4	Operating, systems, processes
D5	Operating, systems, processes, information

Table 3.1.11: A result of Pruning the Contents of Figure 3.1.5

Now, Table 3.1.11 is provided to the Apriori algorithm in step 6, the rest of the ESEM-P is carried out the same as in SEM-P. The complete algorithm for ESEM-P is shown in Figure 3.1.9.

### 3.2.1 Enhanced Semantic-Partitions Algorithm (ESEM-P)

Algorithm 3.2.1 ESEM\_P: (Enhanced Semantic Partitions)

```

■ //An algorithm to generate a classification model for text documents
■ Input: A set T containing n documents
■ Output: A concept Hierarchy (CH) and a HashMap Identifier (HI)
Begin
1. For each Document d in set T do:
    1.1 Extract tokens  $k^d$  with Algorithm Feature Extraction
         $k^d = \text{FeatureExtraction}(d)$  (presented in Figure 3.1.3)
    1.2 Find semantically related tokens in  $k^d$  as semantic partitions, SEM_P
        SEM-P = GenerateSemanticPartitions( $k^d$ ) (presented in Figure 3.1.3.2).
    1.3 For each semantic partitions s in SEM-P assign a rank using number of tokens
        RSEM-P = RankSemanticPartition(SEM-P) (presented in Figure 3.1.4.1).
    1.4 Merge all the semantic partitions in RSEM-P depending on their ranks
        MSEM-P = MergeSemanticPartitions(RSEM-P) (presented in Figure
        3.1.4.2a and Figure 3.1.4.2.b)
        1.4.1a. For each word w in MSEM-P create a vector with four measures, a
        predefined cut_of_threshold.
        1.4.1b. Eliminate all words w in MSEM-P with total vector measure less
        than a given threshold.
2. Generate a concept hierarchy (text file) of frequent items by applying the Apriori
Algorithm.
    CH = GenerateFrequentPatterns(MSEM-P) (presented in Figure 3.1.5.1)
3. Create a table with 0/1 as entries from the CH levels, this is the HashMapIdentifier
table
    HMI = GenerateHashMap Identifier(CH) (presented in Figure3.1.5.2)
End

```

Figure 3.1.7: Enhanced Semantic-Partitions Algorithm (ESEM-P)

## **Chapter 4: PERFORMANCE ANALYSIS**

### **4.1 Implementation Environments**

The performance of the proposed SEM\_P and ESEM\_P algorithms will be compared to the Categorizer in (Zaiane and Antonie, 2002). The experiments consists of two parts; the first part contains two sections. Section one will test the time to preprocess text documents and section two will test the time to develop a classifier using the processed texts. The two times will then be combined and will be taken as the total time to build a classification model. The second kind of experiment will test the effectiveness of the developed classification model using a separate set of text documents. All the experiments are to be performed on a PC with 3.00 GHz Xeon(TM) CPU and 1.00 GB of RAM, running on Windows XP Professional Version 2002 Service Park 2. All algorithms are implemented using Java.

### **4.2 Performance Measures with Existing Text Collections**

In order to effectively evaluate our proposed systems, the same Reuters-21578 text collection used in (Zaiane and Osmar, 2002) has been used. The Reuters-21578 corpus is a collection of news articles that appeared in Reuter's newswire in 1987. This corpus consists of 22 data files all saved in SGML file format. Each of the first 21 files contains approximately 1000 text documents. The 22nd file contains 578 documents that are specifically used by information retrieval researchers for testing purposes as mentioned in (Zaiane and Antonie, 2002). There is also a separate file containing 132 categories assigned to the text documents. To effectively utilize the Reuters-21578 corpus, a Java file has been implemented to extract the contents in each of the 22 data files and results saved into individual text file formats.

The performance of the proposed SEM\_P and ESEM\_P algorithms has been measured in two parts. The first part measures the amount of disc space used to hold the original text documents and the resulting disc space after processing text using the Categorizer in (Zaiane and Antonie, 2002), the proposed SEM\_P and ESEM\_P algorithms. The execution time is also recorded for all three algorithms is also recorded.



The second part of evaluation involves the amount of time it take for the three algorithms, Categorize, SEM\_P and ESEM\_P takes in generating association rules and building a classification model with those rules. The apriori algorithm implemented on WEKA (REF) for generating association rules has been used. WEKA is a system developed in Java at University of Waikato, New Zealand. This system contains implementation of various Machine Learning Algorithms such as, Apriori, Decision Trees and Naives Bayes. The output of the Apriori serves as input to build a classification model. Each of the three algorithms Categorizer in (Zaiane and Antonie, 2002), SEM\_P and ESEM\_P has its own classification model.

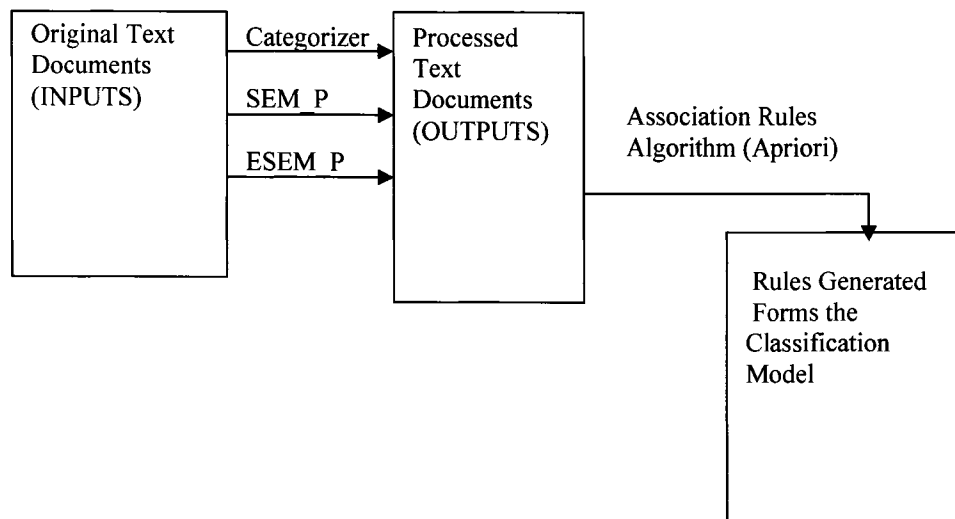


Figure 4.2: Overall process in the experiments

### 4.3 Experiments on Memory Usage and Text Preprocessing Time

Due to the limitations of the java virtual machine, the amount of documents processed has been reduced to a maximum of 525 documents, and experiments performed on 6 different groups of text as shown on Table 4.3.1.

	Number of Documents	Original Documents Sizes	Categorizer	SEM_P	ESEM_P
1.	25	42.4 KB	40.1KB	14.6KB	12.5KB
2.	50	145KB	139KB	63.0KB	54.4KB
3.	75	306KB	279KB	127KB	75.2KB
4.	100	526KB	446KB	210KB	102.4KB
5.	125	804KB	639KB	245KB	213KB
6	150	1.1MB	854KB	275KB	262KB
Total	525	2.9234MB	2.3971MB	934.6KB	719.5KB

Table 4.3.1: Amount of Disc Space Before and After Processing Text

Number of Documents	Categorizer	SEM_P	ESEM_P
25	0.797secs	107.282secs	268.939secs
50	5.328secs	446.767secs	671.798secs
100	54.125secs	836.509secs	1256.609secs

Table 4.3.2: Execution Time to PreProcess Text Documents

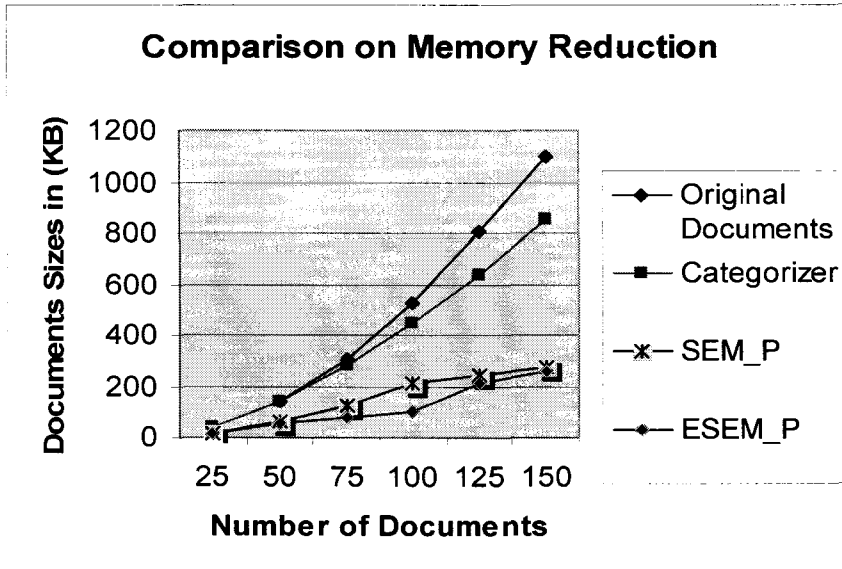


Figure 4.3.1: Memory Reduction after Preprocessing Text

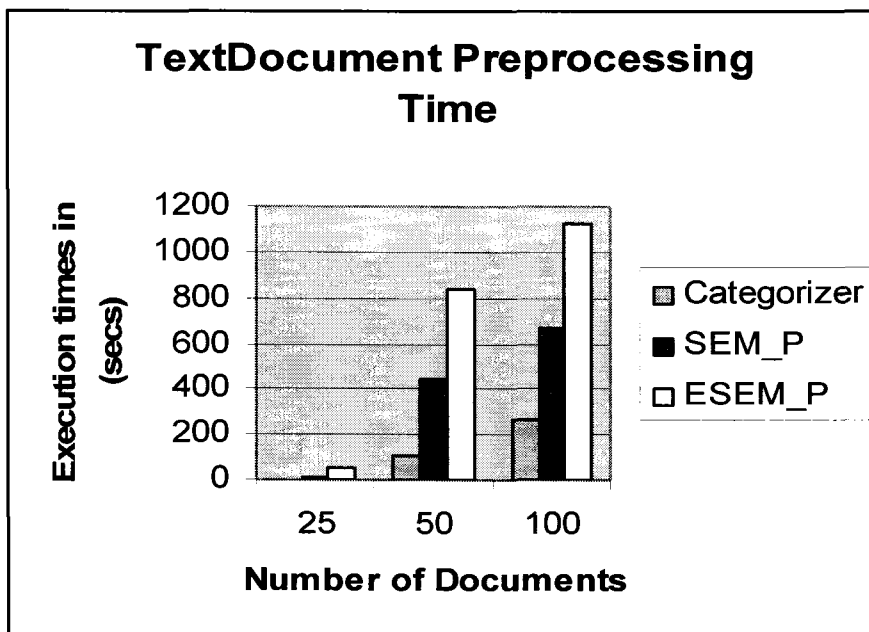


Figure 4.3.2: Comparison on Text Document Preprocessing Times

The amount of time it takes to preprocess text in both SEM\_P and ESEM\_P is 75% higher than the amount of time taken by the Categorizer algorithm. This extra time is taken to identify the semantic orientation of each word in each document using WORDNET ontology. Each document processed by either the SEM\_P or the ESEM\_P

algorithm is reduced to its semantic content resulting to fewer amounts of words than in the Categorizer algorithm.

#### 4.4 Experiments on Building Classification Models

The same approach used in (Zaiane and Antonie, 2002) has been used to build the proposed classification models. A total of ten categories from the Reuters-21578 corpus haven picked as classes to generate association rules. The rules having a certain category as a consequence have been picked to serve as a document classifier. The ten categories are; acq, corn, crude, earn, grain, interest, money-fx, ship, trade and wheat as described in Reuters-21578 documentation.

Each category is passed together with the outputs from each of the three algorithms, the Categorizer algorithm in (Zaiane and Antonie, 2002), the proposed SEM\_P and ESEM\_P. A combination of all the rules generated using each of the categories from each algorithm is done, this is the resulting classification model. an evaluation on how much time it takes to identify frequent itemsets and generate association rules is tested. The time to build a classification model using the outputs from the three algorithms is shown in Table 4.4.1.

	Number of Documents	Categorizer	SEM_P	ESEM_P
1.	25	2.043secs	0.992secs	0.681secs
2.	50	3.2736secs	1.637secs	1.0912secs
3.	75	5.106secs	2.043secs	1.702secs
4.	100	10.242secs	4.121secs	3.414secs
5.	125	20.212secs	8.559secs	5.706secs
6	150	35.424secs	10.212secs	6.808secs
<b>Total</b>	<b>525</b>	<b>76.3006secs</b>	<b>26.6712</b>	<b>18.7722secs</b>

Table 4.4.1: Execution Time in Seconds for Building Classification Models

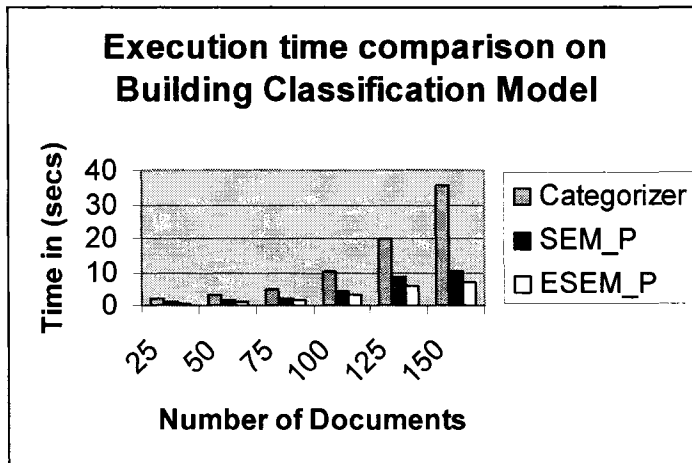


Figure 4.4.1 Execution time for Building Classification Model with (Apriori)

It's clear from Table 4.4.1 that the time to build a classification model using association rules with output generated by the Categorizer algorithm is much slower compared to both the SEM\_P and ESEM\_P algorithms.

#### 4.5 Analysis of Experimental Results

The result in Table 4.4.1 shows a trade off between time and space complexities between the three algorithms, Categorizer, SEM\_P and ESEM\_P. The Categorizer takes a faster approach in preprocessing original text documents but the resulting documents from this algorithm is still very huge. Building a classification model from the Categorizer algorithm takes more time than either the SEM\_P or the ESEM\_P algorithms as in Table 4.4.1 and Figure 4.4.1.

The SEM\_P algorithm takes an original text document, reduces the document to semantically related words, these words forms semantic partitions which are then merged to form the final processed document. This processing takes more time that the Categorizer algorithm, but, the output from the SEM\_P is much smaller in size and takes faster time than the Categorizer in generating association rules and building the classification model. The ESEM\_P algorithm takes more time than either the Categorizer or the SEM\_P algorithms. However, a further reduction of terms in the merged semantic partitions generated with SEM\_P is done by adding a weighted vector to each of the

words. The words that have a vector weight lower than a given threshold are eliminated from the semantic partitions resulting to reduced memory space.

The outputs from both the SEM\_P and ESEM\_P represents the semantic meaning of each processed documents, and can therefore serve as a document summary. These summaries are less bulky than the output from the Categorizer algorithm resulting to faster classification time and can also be easily sent over networks and shared among various processes. The main advantage of the SEM\_P and ESEM\_P algorithms is that the generated semantic partitions serves as structured data for popular data mining algorithms like the Apriori in (Agrawal and Srikant, 1994) providing faster association rule generation and building classification models. These semantic partitions also serve as document summaries and can be shared among different processes as they are less bulky and do not overload any given network.

The two algorithms SEM\_P and ESEM\_P identifies nouns from any given text document, any one can use these algorithms and make adjustments as to which part of speech they wish to process. This adjustment involves just changing the “NOUN” as a part of speech to a users choice, this change is only in one step. Other adjustments one can make, a different ontology can be used instead of WordNET which has been used in the two algorithms to identify the semantic orientations of nouns.

#### **4.5.1 Suggestions to Improve the Preprocessing Time of Both the SEMP and ESEM\_P**

Due to the number of iterations involved in generating all the semantic partitions of any given document, a single process takes a lot of time to process hundreds of documents. To reduce this time, an observation has been made that original text documents can be divided into several groups and have several processes preprocessing these groups of text in parallel and merging the outputs from all the processes to form the classification model. Assigning only a partial segment of the entire text collection to be processed by a single processor can greatly improve the time to identify the words semantics.

## **Chapter 5: CONCLUSIONS AND FUTURE WORK**

This thesis discusses the problem of finding hidden patterns or relations between stored text documents. Related literature includes techniques for text clustering, text summarization, feature extraction, information retrieval and association rules.

A new model called Semantic-Partition for document classification is proposed in this thesis. It aims at capturing the semantic meanings of words in a text document. Words related in meanings tend to refer to the same thing. By grouping these related words, semantic partitions are formed for each text document. Having several documents represented by their semantic partitions, relations between these documents can be retrieved using a data mining algorithm, Apriori (Agrawal and Srikant, 1994). Each document's semantic partitions act as its representation in a transaction dataset. Ranking the semantic partitions reduces the amount of words chosen to represent a document. Low ranked partitions are eliminated from the transactions set. This elimination improves the performance of the Apriori as fewer itemsets are processed.

Given a set of text documents, two algorithms for understanding the contents and relations in these documents have been proposed. The first algorithm is Semantic Partitions,(SEM-P). The second algorithm is an enhancement to the Semantic Partitions, (ESEM-P). Using the reuters-21578 text collection, a classification model has been generated using the SEM-P and ESEM-P algorithms. A weighting and ranking heuristic measure for each term in a partition is used in ESEM-P to prune low ranked terms resulting to improved performance on the ESEM-P over the SEM-P. Each of the two algorithms contain eight steps: feature extraction, semantic orientation, semantic partitions and ranking, merging partitions, pruning partitions, association rule mining of the partitions, forming concept hierarchy of frequent identified terms and generating a hashmap identifier that shows the documents that contain the frequent identified terms.

The theoretical and practical implementations of the proposed two algorithms shows better performance than the categorizer algorithm in (Zaiane and Antonie, 2002) in terms of storage and classification time. The outputs from the SEM\_P and ESEM\_P can serve as input to various machine learning algorithms such as NaivesBayes, Decision Trees, Clustering and Association Rule Mining. The outputs also serve as document summaries that can be shared easily among different networks and processes as they require shorter transfer time compared to the original text documents. The two algorithms can also be scaled to different uses, in this thesis only nouns are considered in generating semantic partitions, any one who wishes to use different part of speech needs only change the “NOUN” entry in the algorithms to the speech of their own choice.

The two proposed algorithms have been developed using Java programming language due to Java’s ability to interact with WordNet ontology that is used in identifying semantic orientation of words in text documents.

There are a number of issues to be addressed in the future.

1. Instead of using only document stored as text, the system could be adjusted to handle different types of documents, e.g. stream mining of emails and multimedia files.
2. The system can also be improved for use in portable devices such as cell phones thereby serving as an organizer for text messages, personal notes and promotions from phone companies.
3. To improve the computation time to develop semantic partitions, parallel processing could be implemented where by different processes process separate text documents and then merging their outputs to form the final product.



## Bibliography

- Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Proceedings of the 20<sup>th</sup> VLDB Conference*. pp.487-499.
- Alter O, Brown PO, Botstein D. (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proc Natl Acad Sci U S A*, 97, 10101-6.
- Baker, L. D. and McCallum, A.K. (1998) Distributional clustering of words for text classification. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval table of contents*. Melbourne, Australia. pp 96 - 103
- Bekkerman,R., El-Yaniv, R, Tishby, N. and Winter, Y. (2003) Distributional Word Clusters vs. Words for Text Categorization. *Journal of Machine Learning Research* 3: 1183-1208.
- Brandow, R., Mitze, K., and Rau, L. F. (1995.) Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675-685.
- Brill, E. (1992). A simple rule-based part of speech tagger. *In Third Conference on Applied Natural Language Processing*, pages 152-155, Trento, Italy.
- Califf, M.E. and Mooney, R.J. (2003) Bottom relational learning of pattern matching for information extraction. *Journal of Machine Learning Research*, 4 pp122-210.
- Calvo, R. A., Lee J.M and Li X. (2004) Managing Content with Automatic Document Classification. *Journal of Digital Information*, Volume 5 Issue 2 Article No. 282.
- Castillo, M. D. D. and Serrano, J. I. (2004). Special issue on learning from imbalanced datasets: A Multistrategy Approach for Digital Text Categorization from Imbalanced Documents. *ACM SIGKIDD Exploration Newsletter*. 6(1), 70-79.
- Chickering D., Heckerman D., and Meek, C. (1997) A Bayesian Approach for Learning Bayesian Networks with local structure. *In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*.
- Chuang, W.T. & Yang, J (2000). Extracting Sentences for Text Summarization: A Machine Learning Approach. *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens Greece. pp. 125-159.
- Church K.W. and Hanks P.(1989). Word association norms, mutual information and lexicography. *In Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

- Dakova, M. (2006) Strategies for Seeking of Information n the WWW. *Academic Internet Journal*, Volume 6, <http://www.acadjournal.com/2002/v6/Part5/p2/>.
- Eiron, N and McCurley, K. S.l (2003) Analysis of anchor text for web search *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval table of contents* Toronto, Canada. pp 459 - 460
- Forman, G. (2003) An Extensible Empirical Study of Feature Selection Metrics for Text Categorization. *Journal of the Machine Learning Research*. 3, 1289-1305.
- Good, I.L. (1965) The Estimation of Probabilities: An Essay on Modern Bayesian Methods, *MIT Press*.
- Gong, Y. and Liu, X. (2001). Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. *Proceedings of the ACM SIGIR*, New Orleans, USA. pp. 19-25.
- Hahn, U. and Mani, I. (2000). The challenges of Automatic Summarization. *Article in IT Professional*, 33(11), 29-36.
- Han, H, Manavoglu, E., Giles, C. L and Zha H. (2003) Rule-based word clustering for text classification. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*. Toronto, Canada pp 445 - 446
- Hearst, M. (1997). Untangling Text Data Mining *Proceedings of ACL 37th Annual Meeting of the Association for Computational Linguistics*. Marlyland, USA. pp. 3-10.
- Holt, J. D and Chung S.M. (1999) Efficient Mining of Association Rules in Text Databases. *Proceedings of the eighth international conference on Information and knowledge management* Kansas City, MO USA. pp. 234-242.
- Hu, M. & Liu, B. (2004). Mining and Summarizing Customer Reviews. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery in Dat Mining*, Seattle, Washington, USA. pp 168-177.
- Huang C., Chuang, S. and Chien, L. (2004). LiveClassifier: Creating Hierarchical Text Classifiers through Web Corpora. *Proceedings of the 13th International Conference on World Wideweb*, New York, USA. pp. 184-192.
- Kongthon, A. (2004). A Text Mining Framework for Discovering Technological Intelligence to Support Science & Technology Management. *PhD Dissertation, Georgia Institute of Technology*, Georgia, USA. pp.105-112.
- Kotcz, A., Prabakarmurthi, V., Kalita, J. (2001). Summarization as Feature Selection for Text Cateorizatio. *Proceedings of the CIKM*, Atlanta GA, USA. pp. 365-370.
- Krishna, K. and Krishnapuram, R. (2001). A Clustering Algorithm for Asymmetrically

Related Data With Applications to Text Mining, *Proceedings of the CKIM*, Atlanta, GA, USA. pp. 571-573.

Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. *Proceedings of the 18th ACM SIGIR Conference* Seattle, Washington, pp. 68-73.

Kuruville, F. G, Park, P.J. and Schreiber, S. L. (2002) Vector algebra in the analysis of genome-wide expression data. <sup>1</sup>*Howard Hughes Medical Institute, Bauer Center for Genomics Research, Department of Chemistry and Chemical Biology, Harvard University, Cambridge, MA 02138, USA* <sup>2</sup>*Department of Biostatistics, Harvard School of Public Health, Informatics Program, Children's Hospital, Harvard Medical School, Boston, MA 02115, USA*

Lin, D. and Pantel, P. (2001). DIRT - Discovery of Inference Rules from Text. *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, USA. pp.323-328.

Liu, B., Hu, M. and Cheng. J. (2005). Opinion Observer: Analyzing and Comparing Opinions on the Web. *Proceedings of the 11th International Conference on World Wide Web*, Chiba, Japan. pp. 342-351.

Marcu, D. (1997). The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts. *PhD Thesis, Department of Computer Science, University of Toronto*, Toronto, Canada. 374 pages.

Mann, W.C. and S.A. Thompson, (1988) Rhetorical Structure Theory: Toward a functional theory of text organization. 8(3): pp. 243-281.

Mann, W.C. (1984) Discourse Structure for Text Generation. *Proceedings of the 22nd annual meeting on Association for Computational Linguistics*, Stanford, California , pp 367 - 375

McDonald, D. and Chen, H (2002). Using Sentence-Selection Heuristics to Rank Text Segments in TXTRACTOR. *Proceedings of the 2nd ACM/IEE CS- Joint Conference on Digital Libraries*, Portland ,USA. pp. 28-35.

Mei, Q. and Zhai, C. (2005) Discovering Evolutionary Theme Patterns from Text - An Exploration of Temporal Text Mining. *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, Chicago, Illinois, USA. pp. 198-207.

Meir, R and Zhang, T (2003) Generalization Error Bounds for Bayesian Mixture Algorithm. *Journal of Machine Learning Research*. 4, 839-560.

Miller G. A. (1995). "WordNet: A Lexical Database for English", *Communication of the ACM*, 4(11), 39-41.

Mooney, R. J and Bunescu R. (2005). Mining Knowledge from Text Using Information Extraction. *ACM SIGKDD Exploration Newsletter*. 7(1), 3-10.

Moniroga, S., Arimura, H., Ikeda, T., Sakao, Y. and Akamine, S. (2005) Key Semantic Extraction Dependency Tree Mining. *Proceeding of the 11<sup>th</sup> ACM SIGKDD international conference on Knowledge discovery in data mining KDD '05*. pp. 666-671.

Nahm, U. Y. and Mooney, R. J. (2001) Mining Soft-Matching Rules from Textual Data. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence(IJCAI-01)*, , Seattle,WA, pp 979-984.

Nicholson, W. K. (2001) Elementary Linear Algebra, *McGraw-Hill Ryerson Limited, 1st Edition*.

Nomoto, T and Matsumoto, Y. (2001) A new approach to unsupervised text summarization. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval table of contents*. New Orleans, Louisiana, United States. pp 26 - 34

Park, J. S., Chen, M. S. And Yu, P. S. (1997). Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Trans. On Knowledge and Data Engineering*, (9)5, pp 813-825.

Patman, F. and Thompson, P. (2003) A New Frontier in Text Mining Intelligence and Security Informatics. *Lecture Notes in Computer Science*. (2665) pp 27 – 38.

Phan, X.H., Ho, T.B., Nguyen, L. M., and Horiguchi, S. (2005). Improving Discriminative Sequential Learning with Rare-but-Important Associations. *Proceedings of the ACM Symposium on Applied Computing*, Chicago, Illinois, USA. pp. 304-313.

Riloff, E. (1999) "Information Extraction as a Stepping Stone toward Story Understanding", In *Computational Models of Reading and Understanding*, Ashwin Ram and Kenneth Moorman, eds., The MIT Press.

Sakurai, S. S. and Suyama, A. (2004). Rule Discovery from Textual Data based on Key Phrase Patterns. *Proceedings of the ACM Symposium on Applied Computing March 2004*, Nicosia Cyprus. pp. 606-612.

Salton, G., Singhal, A., Buckley, C. and Mitra, M. (1996). Automatic Text Decomposition Using Text Segments and Text Themes. *Proceedings of the Seventh ACM Conference on Hypertext*. Washington DC, USA. pp. 53-65.

Sengupta, A. Dalkilic, M and Costello, J. (2004). Semantic Thumbnails - A Novel Method for Summarization Document Collections. *Proceedings of the 22nd International Conference on Design of Communication: The Engineering Quality Documentation*,

- Memphis, Tennessee, USA. pp. 45-51.
- Silber, H. G. and McCoy, K.F. (2000). Efficient Text Summarization Using Lexical Chains. *Proceedings of the 5th International Conference on Intelligent User Interfaces*, New Orleans, USA. pp. 252-255.
- Stairmand, M. A., (1997) "Textual Context Analysis for Information Retrieval" in *Proceedings of ACM SIGIR*, pp140-147.
- Sukhahuta, S and Smith, D. (2001). Information Extraction Strategies for {Thai} Documents, *International Journal of Computer Processing of Oriental Languages (IJCPOL)*, 14(2), pp 153-172.
- Turney, P.D. (2002). Mining the Web for Lexical Knowledge to Improve Keyphrase Extraction: Learning from Labeled and Unlabeled Data. [www.arxiv.org/](http://www.arxiv.org/).
- Turney, P.D. and Littman, M. L. (2003). Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transaction on Information Systems*. 21(4), 315-346.
- Vapnik, V., (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag.
- Yang, J. and Hanovar, V. (1998) Feature Subset Selection Using a Genetic Algorithm. *IEEE Intelligent Systems and their Applications*. 13(2) pp 44-49.
- Yonatan, R. F., Libetson, L., Ankori, K. Schler, J. and Rosenfeld, B. (2001). A Domain Independent Environment for Creating Information Extraction Modules. *Proceedings of the 10th International Conference on Information and Knowledge Management*, Atlanta, Georgia, USA. pp. 586-588.
- Zaiane, O. and Antonie, M.L. (2002). Classifying Text Documents by Association Terms with Text Categories. *Proceedings of the 13th AustralAsian Conference on Database Technologies*. pp. 215-222.
- Zhai C.X., Velivelli, A. and Yu, B. (2004) A cross-collection mixture model for comparative text mining Full text . *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* .Seattle, WA, USA pp.743 - 748
- The reuters-21578 text categorization test collection,  
<http://www.research.att.com/~lewis/reuters21578.html>  
[http://web.mit.edu/be.400/www/SVD/Singular\\_Value\\_Decomposition.htm](http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm)  
[http://en.wikipedia.org/wiki/Stop\\_words](http://en.wikipedia.org/wiki/Stop_words)  
<http://nlp.stanford.edu/software/tagger.shtml>  
<http://www.daviddlewis.com/resources/testcollections/reuters21578/>  
<http://www.cs.cmu.edu/~mccallum/bow/>

### Vita Auctoris

NAME	Catherine Inibhunu
PLACE OF BIRTH	Central Province, KENYA
YEAR OF BIRTH	1974
EDUCATION	School of Computer Science University of Windsor, Windsor, Ontario, Canada 2001-2004  BCS. Computer Science (General) B.Sc. (H) Computer Science with Specialization in Software Engineering and Minor in Mathematics 2001-2004  School of Computer Science University of Windsor, Windsor, Ontario, Canada M.Sc. Computer Science. 2005-2006