Support Vector Machines (SVM) in Text Extraction

By

Nadirah Binti Ghazali

Dissertation in partial fulfillment of
the requirement for the
Bachelor of Technology (Hons)
(Information Communication Technology)

NOV 2006

Universiti Teknologi PETRONAS
Bandar Sri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

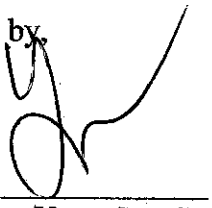
Support Vector Machines (SVM) in Text Extractor

By

Nadirah Binti Ghazali

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,



(Ms. Vivien Yong Suet Peng)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
November 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

NADIRAH BINTI GHAZALI

ABSTRACT

Text categorization is the process of grouping documents or words into predefined categories. Each category consists of documents or words having similar attributes. There exist numerous algorithms to address the need of text categorization including Naïve Bayes, k-nearest-neighbor classifier, and decision trees. In this project, Support Vector Machines (SVM) is studied and experimented by the implementation of a textual extractor. This algorithm is used to extract important points from a lengthy document, by which it classifies each word in the document under its relevant category and constructs the structure of the summary with reference to the categorized words. The performance of the extractor is evaluated using a similar corpus against an existing summarizer, which uses a different kind of approach. Summarization is part of text categorization whereby it is considered an essential part of today's information-led society, and it has been a growing area of research for over 40 years. This project's objective is to create a summarizer, or extractor, based on machine learning algorithms, which are namely SVM and K-Means. Each word in the particular document is processed by both algorithms to determine its actual occurrence in the document by which it will first be clustered or grouped into categories based on parts of speech (verb, noun, adjective) which is done by K-Means, then later processed by SVM to determine the actual occurrence of each word in each of the cluster, taking into account whether the words have similar meanings with other words in the subsequent cluster. The corpus chosen to evaluate the application is the Reuters-21578 dataset comprising of newspaper articles. Evaluation of the applications are carried out against another accompanying system-generated extract which is already in the market, as a means to observe the amount of sentences overlap with the tested applications, in this case, the Text Extractor and also Microsoft Word AutoSummarizer. Results show that the Text Extractor has optimal results at compression rates of 10 - 20% and 35 - 45%

ACKNOWLEDGEMENT

First of all I would like to pay my respects to Allah the Almighty for making me capable of undergoing this project and for showering His blessings upon me throughout the execution of the project and especially at times of need.

The cooperation, help and support provided by Ms. Vivien was a great asset to have during moments of confusion and I truly acknowledge that. She has been a reliable source of guidance throughout the course of this project, and a great supervisor. I would like to thank my close friend Zakiah Anuar, also for her help and guidance in understanding the different aspects of data mining, which I truly needed in the beginning, without her help, I might not have gone half way through this project. I would also like to thank my family for their support and whose love and care enabled me to ease my mind and soul.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
CFS	Cascaded Feature Selection
GUI	Graphical User Interface
IB	Information Bottleneck
IR	Information Retrieval
MS	Microsoft
MSVM	Minimal Support Vector Machines
QP	Quadratic Optimization
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SeqSVM	Sequential bootstrapped support vector machines
SLA	Successive Linear Approximation
SMO	Sequential Minimization Optimization
SRM	Structural Risk Minimization
SVM	Support Vector Machines
SVM-KM	Support Vector Machines-KMeans
VC	Vapnik-Chervonenkis
WMSVM	Weighted Margin Support Vector Machines
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Background

Vital assets of a company is its in-house knowledge that is contained in enormous amounts of documentation, such as reports, process descriptions, drawings, minutes of meetings, and so forth, all of which needs to be organized and archived. As the paperwork grows, problems such as maintaining a clear overview of all the knowledge manually without losing any will gradually occur and can cause serious strategic, legal and profitability issues. To encounter these issues, summarization has been existed to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs, while retaining some of the essential qualities of the original [41] .

In the past, these summaries would have all had to have been produced manually, i.e. by humans. However, with advances in technology and research, especially within the area of Artificial Intelligence, machines have been built and programs have been written which have enabled summaries of varying effectiveness to be created automatically.

Numerous methods have been used to implement text categorization which represents a different machine learning approach: density estimation using a naïve Bayes classifier [18], the Rocchio algorithm [22], a distance weighted k-nearest neighbour classifier [23] [24], and the C4.5 decision tree learner [25]. A method introduced by Bosner, Gayon and Vapnik in 1992, has been widely used for object recognition [26], speaker identification [28], and face detection in images [27], is known as Support Vector Machines (SVM) and will be used to implement text categorization [18] for this project.

1.1.1 Why choose SVM for Text Extraction?

Support Vector Machines is chosen as the algorithm for this project, based on certain features that are stated by Joachims [18] and is listed below:

1. High dimensional input space

   Learning text classifiers deal with huge amounts of features. Although so, SVM is able to handle this situation since it uses overfitting protection that is independent on the number of features.

2. Few irrelevant features

   Irrelevant features can be determined and removed by feature selection. In the case of text categorization, there are only a few irrelevant features which may result in loss of information if aggressive feature selection is used. This is supported by an experiment by [18] indicating that even features ranked the lowest still contain considerable and relevant information.

3. Most text categorization problems are linearly separable

   SVM behave robustly over a variety of different learning tasks and eliminates the need for manual parameter tuning which makes SVM fully automatic.

1.2 Problem Statement

Nowadays, the growth of online technology has made text categorization one of the most important techniques to handle and organize textual information [8]. Text categorization techniques are used for many situations for example correctly categorizing an article from a collection of documents, finding important nuggets of information in voluminous texts, web search implementation, and email categorization.

There have been various methods to employ text categorization such as neural networks [36], regression models [37] and decision trees [38]. However, these methods have their own setbacks that will result in poor development of classifiers due to performance variation using different types of data collections [40]. For example, the Naïve Bayes algorithm has been said to be unsuitable for applications that require smoother class posteriors [39], like text categorization applications. Therefore, numerous researches have been done to further enhance these methods in order to better suit and increase the performance of text categorization.

Sending unsolicited information has become easier with such developments as email and easily-generated mailing lists. Therefore, with the same amount of time, but with more information we have been seeking methods to digest the same amount of information, but at a reduced effort. This is where summarization comes into its realm. There are various reasons why summarization of documents has increased in popularity:

1. The user knows whether to read the document or not and whether it will provide the information they need
2. It allows the user to revise quickly what they have already read
3. The user can seek someone else's opinion on the document or source (e.g. review).

As mentioned above, the greater the increase in information, the greater the need to reduce this information into smaller manageable chunks to see if the full information is relevant to our needs or not. These are the advantages of producing summaries automatically:

1. It might be virtually impossible, due to the medium of the material, to manually summarize all the potentially necessary information, for example, when searching the Internet using a search engine.
2. There is no time to produce an abstract or summary manually
3. No expertise exists at the time to produce a manual summary

4. The material might be of a sensitive or confidential nature
5. The material might all need to be produced to a certain standard or in a certain consistent format
6. A summarization system can tailor-make a summary to a user's exact requirements, for example, to answer a query

1.3 Objectives

The aim of this project is to look at the current situation in the area of summarization research, study the evolvement of SVM, then create a summarization tool which takes into account the existing research. The research technique to be used will be discussed further in the next section, but it is likely that the technique used will be of the feature-identification type, and a sentence-extraction summarizer will be built using this practice. The objectives are to use Support Vector Machines to summarize the test corpus, by classifying each word in the test document with the relevant values for whether it should or should not appear in the summary.

The evaluation method employed within the study will be of an intrinsic nature, with the auto-generated summaries being compared with human-generated extracts. However, due to time and resource constraints, it is likely that comparisons will be made between the Text Extractor with another auto-generated summarizer that has been available in the market.

1.4 Scope of Study

Extraction is considered summarizing a full document into a condensed version. However, there are various types of summaries that can be developed, often depending on the user needs and requirements. Below are some of the types of summaries that can be produced:

1.4.1 Abstract vs. extract

According to Mani [41], the distinction between abstracts and extracts is, "an abstract is a summary at least some of whose material is not present in the input", whereas, "an extract is a summary consisting entirely of material copied from the input."

There is a fundamental limitation to the capabilities of extraction, in that only sentences or phrases in the original document are included, and often there is less semantic or syntactical analysis of the information than with an abstract, whereas abstraction requires knowledge of the meaning of the information, and some ability to make inferences at semantic level.

It has been said that all summaries can created from simple extracts; "A useful first step in the automatic or semi-automatic generation of abstracts from source texts is the selection of a small number of meaningful sentences from the source text and to achieve this, each sentence in the source text is scored according to some measure of importance, and the best-rated sentences are selected [42]."

1.4.2 User-focused vs. Generic

Taking into account of the user's needs is an important consideration for the summary to be generated. A user-focused summary is a tailor-made summary for a particular user, often in response to a query which the user has. Firmin et al [43] discussed the concept of creating the summaries with regard to intent, focus and coverage for a particular user, sometimes in response to a particular question, sometimes for a specialized group. On the other hand, generic summaries have a broader user-base. They are said to be author-focused, as they concentrate on the author's views, as opposed to any particular user.

Another fundamental concept essential to text summarization, is that summaries that are generated should be rational. A lot of research has found that the problem of extracts is due to semantic meaning that is not being sought and since the use of anaphors in language is so productive, yet very unpredictable, the exclusion or manipulation of them has not yet been achieved. Creating a user-focused summary could eliminate some of these difficulties with rationality, as the important details might be more obvious, due to knowing what the user wants.

1.4.3 Compression Rates

The compression rate of a summary is normally calculated as a percentage of the full source's length. Standard compression rates are a summary between 10-30% of the original document, but obviously using any rate between 1% and 99% would be considered a summary, though some of the benefits to summarization might be lost because the summary at 10% compression rate was found to be better than the one at 20% [44].

To provide some guide as to what the compression rate should be, two questions are needed to be considered when creating a summarization technique:

1. How long will a human user be willing to read the summary for?
2. How long does the summary need to be in order to capture all the relevant information?

Three major condensation operations which summarizers can carry out in their summarization process have been identified as follows [45]:

1. Selection – the filtering of elements,
2. Aggregation – the merging of elements,

3. Generalization – a substitution of certain specific elements with more general ones.

These condensation operators will invariably alter the target summary size to a greater or lesser extent. For this particular project, the text extraction application using SVM text categorization methods will be able to accommodate documents in English, analyze text files, and perform qualitative measures on the text documents. The type of material which will be summarized within this study will be written documentation or text without any multi-media material. The materials analyzed and summarized will not include any tables, graphs or pictorial information.

CHAPTER 2

LITERATURE REVIEW OR THEORY

2.1 Summarization Approaches

2.1.1 Surface-Level Approach

Surface-Level is an approach which can be statistically calculated by the system in order to determine the important information to be extracted and become the summary. It has been said to be the basis for a lot of summarization research [42]. Examples of features that can be statistically calculated are listed below:

1. Location – this refers to the location of terms or sentences in a sentence, paragraph or the whole document,
2. Keywords – the frequency of terms which can often lead to the thematic meaning of the document [41]. This theory assumes that a word's importance is relative to the frequency of the term in the document, but inversely relative to the total number of documents in the corpus that contain the term. However, this method does not work well with documents of the same type or even with some articles all taken from the same period, as they have terms occurring too frequently for the salience to be considered worthwhile
3. Headings – this includes making use of words in headings and in the title to assist in providing the theme or more information on the salient topics
4. Cue words and phrases – determining certain phrases used in language can advise on the pertinence or redundancy of surrounding words and phrases

2.1.2 Entity-Level Approach

This level of processing involves building an internal representation for the text, by modeling the text entities and the relationships between the entities. The researchers of this approach towards summarization, tried to represent patterns of connectivity between terms in the text in order to show what is important. Such features include:

1. Similarity between terms
2. Words which are related, due to their occurring in common contexts
3. Proximity between text units
4. Logical relations, such as agreements and contradictions
5. Thesaural relationships between words

2.2 Support Vector Machines

Support Vector Machines are learning machines that plots the training data in high-dimensional features space and labels each data by its class. Currently, Support Vector Machines (SVM) is labeled the best approach to the classification of datasets [18]. It was first introduced by Bosner, Gayon and Vapnik in 1992. SVM has been proven to be successful in applications such as bioinformatics, text, handwriting recognition, and image processing [4].

Support vector machines are based on the Structural Risk Minimization principle [20] which finds a hypothesis h for which the lowest true error is guaranteed. The true error of h means the probability that h will make an error on an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis h with the error of h on the training set and the complexity of H (the hypothesis space containing h) measured by VC-dimension [20]. To control the VC-dimension of H, SVM must find the hypothesis h which minimizes the bound of the true error [17].

The learning ability of SVM is independent of the dimensionality of the feature space, meaning that it is able to generalize despite the size of the feature provided that the data is separable with a wide margin using functions from the hypothesis space. Parameter setting for SVM is crucial in producing a hypothesis with the lowest VC-dimension and allowing automatic parameter tuning without expensive cross-validation.

By choosing different kernel functions we can implicitly project the training data from input space X into feature space F for which hyper-planes in F correspond to more complex decision than boundaries in the input space X. There are many types of kernels which two of them are: Polynomial and Radial Basis Function. The computations of two given classes of SVM are Linearly Separable SVM and Linearly Non-separable SVM [14].

2.2.1 Linearly Separable SVM

The Linearly Separable SVM by Gorelick and Friedman [4] learns from training sets to find a classifier in a vector space which best separates the data points into two classes. These are represented in Figure 2.1. The two parallel dashed lines are boundaries in which the solid line can move between them without causing any misclassification. The data points are indicated by both the circles and squares, where the ones lying on the dashed lines are the 'support vectors'.

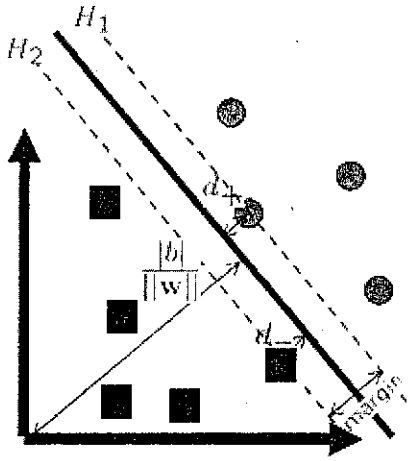


Figure 2.1: Linear Separating Hyperplanes for the Separable Case [4]

Below are the description of **Figure 2.1** and the equations used to obtain a linear separable Support Vector Machine (SVM) with all the variables used, which are as follows:

1. The equation to separate hyper-plane H in figure 1 is: $\mathrm{w} \cdot \mathrm{x} + b = 0$
   - **w** is normal to hyper-plane H
   - x is an arbitrary feature vector
   - w and b are learned from a training set of linearly separable data
2. |b| / ||w|| is the perpendicular distance from hyper-plane H to the origin
3. (d+) + (d-) is the margin or shortest distance from hyper-plane H to the closest positive (negative) point.
4. If H is a separating hyper-plane, then

   $\mathrm{x}_i \cdot \mathrm{w} + b \geq d_+$ for $y_i = +1$

   $\mathrm{x}_i \cdot \mathrm{w} + b \leq d_-$ for $y_i = -1$

   - x_i and y_i are points on the feature space where:

     $\mathrm{x}_i \in \mathbb{R}^n, i = 1, \ldots, l$

     $y_i \in \{-1, 1\}$
5. No training points should fall between hyper-plane H_1 and hyper-plane H_2
6. A simpler derivation of all the variable from above results in the equation below:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \forall i$$

2.2.2 Linearly Non-separable SVM

The Linearly Separable SVM algorithm which was explained in section 2.1.1 as above cannot handle non-separable data [4]. Therefore, an addition to the previous algorithm is made for this situation which relaxes the constraints that is by introducing positive slack variables to aid for the occurrence of training errors.

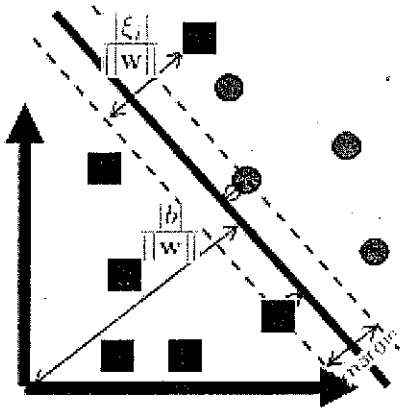


Figure 2.2: Linear Separating Hyperplanes for the Non-separable Case [4]

Figure 2.2 is the graphical depiction of the Linearly Non-separable Support Vector Machines. The equations and variables that are necessary for this case are as follow:

1. Positive slack variables (ξ_i) are introduced to relax constraints. These variables are not present in the equation for the linear separable case explained previously. The equation is as follow:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i$$

2. The equation $\frac{\|\mathbf{w}\|^2}{2} + C(\sum_i \xi_i)^k$ needs to be minimized
   - C is a penalty parameter chosen by the user

- ||w|| is the Euclidean norm of w, while $\sum_i \xi_i$ is an upper bound on the number of errors

2.3 Minimal Support Vector Machines (MSVM)

The Minimal Support Vector Machines uses smaller sets of data points compared to a 1-norm or 1-type SVM classifier. It is suitable as an incremental algorithm that maintains a small portion of a large dataset before merging and processing it with new incoming data which allows the maintenance of a small fraction of a large dataset.

Wu and Srihari [3] says that MSVM is a method for selecting a small set of support vectors (also termed as *minimal,* which are data points corresponding to constraints with positive multipliers) which completely determines a separating plane classifier, and has never been done before in previous projects. It has improved testing set accuracy over one that is chosen by a standard SVM, as well as faster execution in terms of computing time since redundant data are removed in early stages.

Classifications of datasets are achieved by either a linear or a non-linear separating surface in the input space of the dataset using Successive Linear Approximation (SLA) that leads to an improved generalization classifier.

The linear SVM works as a separating plane classifier midway between and parallel to two bounding planes, with maximum distance (or margin) between them. The bounding planes attempt to place the two classes of a given dataset on opposite sides. To produce the separating plane, a quadratic program or a linear program is solved, depending on the standard used in measuring the distance between bounding planes. There are two cases in linear SVM, that is, the linearly separable case and the linearly non-separable case. As the name implies, the latter produces some errors between the distances of the two bounding planes.

2.4 Weighted Margin Support Vector Machines

Based on Wu and Srihari [3], the Weighted Margin Support Vector Machines compensates for lack of information in existing datasets. It permits the incorporation of prior knowledge with the right kernel function chosen. To train the data, a Sequential Minimization Optimization (SMO) is used to generalize on imperfectly labeled training dataset because each pattern in the dataset associates not only with a category dataset but also a confidence value that varies from 0.0 and 1.0.

Like always, a set of vectors are given along with their corresponding labels where the SVM defines a hyper-plane in kernel mapped feature space that separates the training data by a maximal margin.

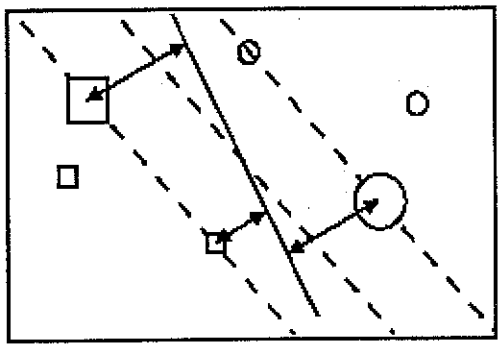


Figure 2.3: Weighted Margin Support Vector Machines [3]

In **Figure 2.3**, the positive class or data are depicted as circles and negative as squares. The size of the squares/circles represents their associated confidence value. The dashed line in the middle is the hyper-plane derived based on the standard SVM training, and the solid line is the solution to the transductive SVM learning.

There are two types of WMSVM which are: Weighted Hard Margin Support Vector Machines and Weighted Soft Margin Support Vector Machines.

2.4.1 Weighted Hard Margin Support Vector Machines

According to Wu and Srihari [3], the simplest model and the easiest algorithm of support vector machine is the maximal weighted hard margin classifier. It only works on a data set that is linearly separable in feature space. Thus, it cannot be used in many real-world situations.

When each label is associated with a confidence value, intuitively one wants support vectors that are labeled with higher confidence to assert more force on the decision plane. So, to train a maximal weighted hard margin classifier, the effective weighted functional margin is fixed instead of the functional margin of support vectors. Then the norm of weight vector is minimized.

2.4.2 Weighted Soft Margin Support Vector Machines

To solve problems for linearly non-separable datasets, the weighted soft margin SVM was introduced by Wu and Srihari [3]. The soft margin classifier is typically the solution that generalizes or simplifies the soft margin classifier to a weighted soft margin classifier by a weighted version of slack variable. Here the effective weighted margin slack variable is used so that the final decision plane will be more tolerant on these margin violating samples with low confidence than others. Therefore producing samples with high confidence label to contribute more to the final decision plane.

2.5 Applying Cascaded Feature Selection (CFS) to SVM text categorization

Text categorization has the increased need for a high-precision system instead of a high recall but with low-precision system [15]. A high-precision system means that the system has the capability to categorize documents according to the actual category based on the contents of the document.

Problems exist in text categorization where the input feature for the document dataset has high dimensionality. This indicates that the occurrence of words in documents is very high which results in malfunction of categorization techniques and in order to encounter this, the use of a cascaded feature selection (CFS) using Support Vector Machines (SVM) [10] is needed.

According to Masuyama and Nakagawa [10], the two steps of CFS are: first, classifying test documents either into a positive or negative set, and secondly, further classifying the positive sets from the first step into a positive or negative set. By doing this, the characteristics of each feature can be used and negatively categorized documents in the first step can be removed resulting in the expectations of a higher precision of this method.

2.6 Sequential bootstrapped support vector machines (SeqSVM) - a SVM accelerator

Since SVM is required to solve a quadratic optimization (QP) problem which needs resources that are at least quadratic on the number of training samples, training time and memory increase dramatically with the increase of the training set. In order to solve this problem, a method which uses a SVM accelerator called sequential bootstrapped SVM (SeqSVM) is proposed by Li et al. [4].

The key principle in this method is to help the SVM pick the convex hull sample that is wrongly classified by the current SVM and furthest from the current SVM solution [4]. The convex hull sample, which disagrees most with the SVM solution, will lie on the convex hull of each class distribution and all support vectors lie on the convex hull in the case of linearly separable classes. Two difficulties that needs to be overcome is the SeqSVM's iterations will take too many if there are too many support vectors, and when the class distributions are not separable, it is not easy to pick convex hull samples.

These difficulties were overcome using artificial database and benchmark databases, demonstrating the effectiveness of the proposed algorithm to reduce the learning time of SVM on the whole training set [6].

The principle of SeqSVM is simple and is not straight forward. Hence, it is necessary to refine it to take care of computational efficiencies and non-separable cases. This algorithm runs on top of any techniques, such as Chunking, decomposition and SMO, to handle large scale problems. The algorithm can be described as follows:

1. Separate the large training set randomly into two disjoint subsets: *TrainActual* and *TrainPool*
2. Train a SVM classifier on the current *TrainActual* set
3. With the derived SVM, select the convex hull sample in the *TrainPool* set that is furthest away from and on the wrong side of the current SVM hyperplane
4. Transfer the convex hull sample to the *TrainActual* set
5. Retrain the SVM again

Support vectors determine the final decision boundary of SVM. Consequently, if a SVM classifier could be trained with only the support vectors in the large training set, less training samples can be used to obtain the same generalization performance as using all the training samples and therefore decrease training time greatly. Therefore, in the SeqSVM, training is focused on the samples which have more probability to be support vectors and pay less attention to the samples which have less probability in the large training set.

2.7 On Feature Distributional Clustering for Text Categorization

This approach is one that was introduced by Bekkerman et al. [16], which combines feature distributional clusters with a Support Vector Machine (SVM) classifier. A more

efficient word-cluster representation of documents is produced by employing distributional clustering of words using the information bottleneck method. Moreover, it further yields high performance text categorization that can generate categorization accuracy and representation efficiency.

The combination of the Information Bottleneck (IB) clustering framework with SVM has produced a high performance categorization of the well known 20 Newsgroups (20NG) dataset [31]. IB clustering is actually used to represent documents in a feature cluster space instead of a feature space, where each cluster is a distribution over document classes.

Bekkerman et al. [16] states the three advantages of using distributional word clusters which are as follows: A dimensionality reduction is performed which implicitly considers correlations between the various features rather than considering each feature individually. The clustering achieved by the IB method provides a good solution to the statistical sparseness problem that is important to the straightforward word-based document representation. Finally, the clustering of words allows extremely compact representations that allow the use of strong but computationally intensive classifiers.

The scheme of this method is based on IB distributional clustering whereby the words of the documents are clustered into k clusters using the deterministic annealing implementation of the information bottleneck method, and then the articles are projected onto pseudowords and later trained by the SVM classifier. Below is a brief explanation of information bottleneck and distributional clustering and distributional clustering via deterministic annealing.

2.7.1 Information bottleneck and distributional clustering

Relevant encoding of the random variable X relies on partitioning of X into domains that preserve the mutual information between X and another given variable, Y [16]. The

resulting partition or clusters of X constitute an approximate sufficient partition that enables the construction of an optimal code over X that provides all the information that X has on Y. The problem has a simple variational formulation which finds the optimal trade-off between the minimal partition of X and the maximum preserved information on Y.

2.7.2 Distributional clustering via deterministic annealing

Deterministic annealing is a top-down hierarchical clustering procedure introduced by Salton and McGill [32] that requires the use of an appropriate annealing rate in order to identify "phase transitions" which corresponds to cluster splits.

2.8 Boosting SVM for Text Classification through parameter-free Threshold Relaxation

There have been many algorithms proposed to address the need to improve the recall of an information retrieval system without affecting its precision using Support Vector Machines since it has been discovered that the performance of SVM for text classification is not competitive from a recall perspective [30]. Text classifications have unevenly distributed and poorly represented classes that can lead to an over fitting of more frequent classes which eventually leads to the reduction of recall [29]. One of the proposed algorithms includes a parameter-free threshold relaxation which relies on a process of retrofitting and cross validation to set algorithm parameters (beta and gamma) empirically.

2.8.1 Utility Models

Utility models are used to measure the degree of satisfaction of a user's expectations on how well the text classification system makes independent decisions, whether a document belongs to a given class or not. According to [29], incorporating utility

models into SVMs can be accomplished heuristically through embedded learning strategies such as beta-gamma threshold adjusting algorithm to determine how far the threshold satisfies the utility measures.

2.8.2 Beta-gamma Threshold Adjusting Algorithm for SVM

The core beta-gamma threshold relaxation strategy consists of two procedures: *TrainSVM* and *SetThresholdUsingBetaGamma* [29]. Both procedures use the following as input:

C: a category label

T: a labeled dataset of documents consisting of both positive and negative examples of *C*

UtilityMeasure: a utility measure that models the user

n: the number of folds that will be used in parameter selection

M: a model that models the category *C*

β *and* γ: the threshold adjustment parameters

p : denotes the number of positive documents in the thresholding dataset, *T*

Procedure 1: *SetThresholdUsingBetaGamma (C, T, M, β, UtilityMeasure)*

1. Rank the thresholding dataset, *T*, using the SVM, *M*, as a scoring function, thereby yielding a ranked document list *R* consisting of tuples <Document$_i$, Score$_i$>
2. Generate the cumulative utility curve for *R*, i.e., for each document in the ranked list *R* compute the cumulative utility using the utility measure *UtilityMeasure*.
3. Determine the rank or index of the maximum utility point on the cumulative curve and of the first zero utility point following the maximum utility point. Denote these as i_{Max}, and i_{Zero} respectively. Assign the variables θ_{max} and θ_{Zero}

with the output scores of the model, M, for the documents associated with the maximum and zero utility points respectively.

4. Return the threshold, θ, which is calculated as follows:

$$\theta = \acute{\alpha}\theta_{Zero} + (1 - \acute{\alpha})\theta_{max}$$

$$\acute{\alpha} = \beta + (1 - \beta)e^{-py}$$

Procedure 2: *TrainSVM (C, T, UtilityMeasure, n)*

1. Train an SVM, M, using T and an SVM training algorithm.
2. Partition the data T into n non-overlapping subsets of the data ensuring that both positive and negative documents are present in each fold or subset. In particular, if P denotes the number of positive documents in T, then each fold is forced to have approximately P/n positive documents.
3. $\beta_{sum} = 0$
4. Foreach fold n
   i. Set T_n to the remaining n-1 folds.
   ii. Call *SetThresholdUsingBetaGamma (C, T_n, M, void, void, UtilityMeasure)* and assign the variables θ_{max} and θ_{Zero} the corresponding values in Step 3 of the *SetThresholdUsingBetaGamma* routine.
   iii. Rank the documents in the held-out fold n using model M and generate the cumulative utility curve for this ranked list R, i.e., for each document in the ranked list R, compute the cumulative utility using the utility measure *UtilityMeasure*.
   iv. Determine the rank or index of the maximum utility point on the cumulative curve, denoted as i_{Max}. Assign the variable θ_n the output score of the SVM, M, for the document associated with the maximum utility point i_{Max}.
   v. Retrofit β_n as follows:

$$\beta_n = \frac{\theta_n - \theta_{Max}}{\theta_{Zero} - \theta_{Max}}$$

vi. $\beta_{sum} = \beta_{sum} + \beta_n$

5. End Foreach
6. Set β to the average over the βs determined for each fold as follows: $\beta = \beta_{sum} / n$
7. Calculate the optimal threshold, θ_{Opt}, using β (that was determined using the previous steps) as follows: *SetThresholdUsingBetaGamma (C, T, M, β, γ, UtilityMeasure)*
8. Alter the SVM classification rule slightly as follows to accommodate the adjusted threshold:

$$\textbf{Class } (X) = \textbf{Sign}((W, X) + b + \theta_{Opt})$$

An advantage of this approach is that it does not require the user to provide a list of possible values for beta and gamma given that the value of β is determined using retrofitting, thereby, alleviating the need for a cross validation exploration of alternative β values.

2.9 K-Means

K-Means algorithm groups objects based on its attributes into K number of groups, where K is a positive number. These groupings are based on the measurements of the minimum distance between the objects and the cluster's centroid [35]. Two procedures are available to search for the optimum set of clusters. The first assigns each object to a cluster and the second sets initial positions for the cluster centroids.

In the first procedure, the objects are randomly assigned to one of the K clusters. Once this is done, the positions of the K centroids are determined, as is the value of the metric to minimize. A global optimization method is then used to reassign some of the objects to different clusters. New centroids are determined, as is the metric to minimize. This procedure is continued until the optimum assignment of objects to clusters is found.

Figures that show the graphical process of K-Means are included in Appendix A, **Figure 2.4**.

Various metrics to the centroids that can be minimized include:

1. Maximum distance to its centroid for any object.
2. Sum of the average distance to the centroids over all clusters.
3. Sum of the variance over all clusters.
4. Total distance between all objects and their centroids.

2.10 Multi-document Biography Summarization using SVM

In this paper, a system that uses Information Retrieval (IR) and text categorization techniques was described to provide summary-length answers to biographical questions by extracting biography related information from large volumes of news texts and composing them into fluent, concise, multi-document summaries. The summaries generated by the system address the question about the person, though not listing the chronological events occurring in this person's life due to the lack of background information in the news articles themselves.

Support Vector Machines (SVM) was used in this study, to classify sentences into one of the two biography categories whereby sentences are categorized based on their biographical saliency and their non-biographical saliency, both quantified in percentage. It was proven that SVM produced the second highest percentage precision and recall indicating that SVM has the capability of classifying sentences [47].

2.11 SVM-KM: Speeding SVMs learning with a priori cluster selection and k-means

The author combined SVM and K-Means to accelerate the training of Support Vector Machine by first grouping the training vector in many clusters whereby clusters that are

formed only by a vector that belongs to the same class label can be discarded and only cluster centers are used. On the other hand, clusters with more than one class label are unchanged and all training vectors belonging to them are considered. Cluster with mixed composition are likely to happen near the separation margins and they may hold some support vectors. Consequently, the number of vectors in SVM training is smaller and the training time can be decreased without compromising the generalization capability of the SVM.

According to the results obtained from the experiment, it was concluded that the combination of SVM and K-Means reduced the total training time by reducing the training set size and therefore decreasing the SMO's training time [48]. SVM-KM was efficient when dealing with low dimensional and dense training sets. However, performance was affected whenever the dimension becomes larger since several distance evaluations are performed during the k-means execution.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In the preceding chapter, a lot was discussed about the theory of Support Vector Machines, the different kinds of enhancement that was made to produce an accurate categorization, the implementations that were necessary for the development of text categorization applications, and the benefits and drawbacks of each implementation. This chapter focuses on the methodology of this project.

3.2 Planning

There exist various applications that involve processing data especially textual classification, such as search engines and of course, text summarizers. But do people know what happens behind the interface of these applications? Initially, the intention of this project is to study the effectiveness of data mining algorithms, particularly Support Vector Machines (SVM). This study would be able to realize a better way of implementing text categorization applications in a way that it produces a more precise and convincing outcome.

What better way to implement this algorithm other than constructing a text extractor, which will serve as a means to display the outcomes of each textual processing, enabling the measurement of the algorithm's efficiency and performance computation using a particular testing data. These computations will eventually be compared to with the measurements from other existing text summarizers or extractors. Comparing the SVM-based text extractor with other algorithms will prove how this algorithm performs.

3.2.1 Aims of Text Extractor

The system developed will be called Text Extractor. The aim of the project is to develop a text summarizer that reads in a text document and automatically generates a text summary based on sentence extraction. It is proposed that the summary will be indicative and generic.

The project will be divided into a training element, the auto-generated summarizer, and a method for evaluating the resulting summaries. The features and structure of these programs will be discussed in greater detail in the Design section.

3.2.2 Requirements

Since the requirements are based on establishing a workable and satisfactory summarization tool, the emphasis is more on the equation being correct and producing auto-generated summaries, which are comparable to human-generated summaries, and with a satisfactory evaluation method, than creating a pleasing interface with a lot of functionality. Therefore, the requirements are stated as follow:

1. The system can read in individual text documents from the corpus
2. The system can statistically analyze the source document
3. The system can output a summary to a specified length
4. The user selects the length of the summary required
5. The corpus is in a machine-readable format
6. The corpus contains source documents with an accompanying human-generated extract
7. The summary is generic and indicative
8. The user interaction is through a GUI (Graphical User Interface)

9. The system is able to print out summary generated by the application

3.3 Analysis

Recognizing the need to understand about the theory and background of the chosen algorithm was the vital step after realizing the proposed topic. Studying the concept and the underlying algorithm of Support Vector Machines was carried out after rigorous attempts of gathering ample information on SVM through the internet. Professional and trusted articles were obtained from websites which gave propositions to the enhanced versions of SVM.

Present nowadays are various tools that cater for text analysis in the market today, some of which are open source software. The search for these tools over the internet was carried out and a study in terms of the user interface was conducted on these tools. Most of these tools incorporate many functions in one application, for example, the combination of text summarizer, vocabulary editing, and query searching.

Additionally, this project requires the understanding of text classification whereby documents are classified into groups with similar traits or attributes. In most articles, many have agreed that SVM makes the best algorithm for classifying text [17]. However, since SVM has existed a decade ago [18], there are some setbacks that needs to be encountered for, especially in terms of its performance. Therefore, many researchers have come up with numerous ways to increase the performance of SVM in text categorization.

In this case, to accelerate the performance of SVM, it was decided to unite SVM with the usage of another data mining algorithm, called K-Means clustering. This collaboration is useful since K-Means clustering differentiates every data from each other and groups them into clusters of similar traits. For example, words that form verbs are put into one cluster, while words forming adjectives are put into another cluster.

Since words are already in order, further classifications of words are made easy resulting in the reduction of the SVM algorithm cycles producing an accelerated performance.

Another effort to enhance the overall performance of this project is to increase the generalization capability of SVM-KM since it has been proven to have had a lower percentage compared to SVM itself from an experiment conducted by Almeida M. B. et al. [33]. To encounter this situation, new kernel parameters should be specified to the SVM-KM algorithm instead of using similar kernel parameters.

3.4 Design

3.4.1 Design of the Text Extractor program

The overall design of the programs and their relationships can be seen diagrammatically in **Figure 3.1** below. There are four stages to the overall extraction system engine which is shown in the diagram below accompanied by the explanation of each element in the system architecture.

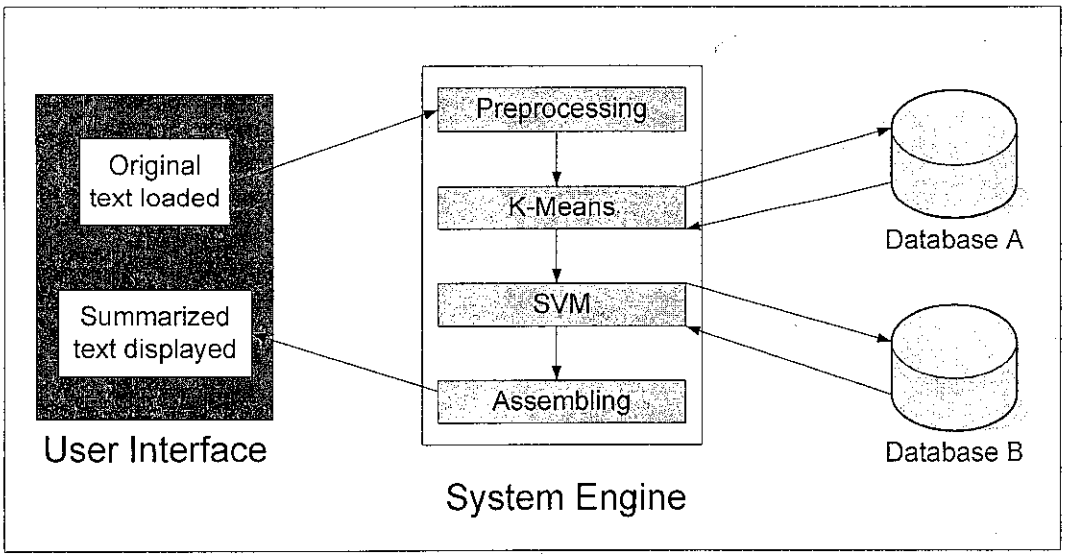


Figure 3.1: System Architecture of the Text Extractor System

The system architecture of the overall project would be as in **Figure 3.1**. The user will first load the original document using the user interface. When the extract button is pressed, the whole document will go through the system engine, which consists of four stages; namely preprocessing, K-Means algorithm, SVM algorithm, and assembling.

At the preprocessing stage, the whole document will be processed to remove stop words, then being split up into individual words and sentences and stored in separate structures. This is done in order to make it easier for the machine learning algorithms to process each word.

K-Means algorithm will basically take each word and group them into their respected categories. This is done with reference to the database that is provided for the algorithm to compare each word with the categories available in the database. There are eight categories altogether, which are noun, verb, adjective, adverb, pronoun, conjunction, interjection and preposition.

After completely obtaining the eight different clusters of words, it is passed on to the next algorithm which is Support Vector Machines. Here, the words contained in each cluster are processed by finding and comparing the synonyms of the words that are stored in another database. Comparisons made will allow accurate word occurrence to be determined. For example, the words 'important' and 'crucial' is present in one particular cluster. Both are of different words and each word will hold the word occurrence of 1. However in this case, by considering the synonym of both words, the word occurrence will become 2 instead of the latter scenario because the meaning of both words is the same.

The word occurrence frequency obtained from the SVM algorithm will be used to score sentences which are done in the assembling stage. The sentence having the highest score will be put at the beginning of the extract followed by the sentences after it. Following that, the extract is generated and is displayed onto the user interface.

3.4.2 Database

Both databases used for this project are of Microsoft Access format. Database A contains a corpus of words that are categorized under different parts of speech including verb, preposition, noun, interjection by which each category has its own column while database B contains a corpus of words that are organized into synonym sets. Both databases consist of approximately about 135,000 and 500,000 words of running English text, respectively.

3.4.3 Stop word list

The stop word list is a list of terms to be excluded from consideration when, for example, weighting terms or sentences. Excluding certain words is an important part of any summarization research, because it avoids unnecessary bias towards words which bring little benefit to an analysis of a technique, for example, the keywords frequency count technique. In this program, any stop words will have to be removed before analyzing each sentence for the presence or absence of the rest of the more relevant terms.

The stop word list used for this summarization project based on the list used by Ovid Technologies [46] by which the stop word list used in this project is a modified version of this particular list, as some of the words were moved into the bonus or stigma word sets instead.

3.4.4 Graphical User Interface

Designing the interface of the application was solely dependant on the element that comprises the overall application. Among the elements needed are text areas to display the original document, the summary of the document and the details of the documents

that have been summarized. Apart from that, a function such as the execution button to execute the summarizer is also taken into consideration.

Another important part constituting to the design of the application is associated with the layout of the elements in the application. It is crucial to design the layout of the application systematically to cater for the ease of its usage or its user friendliness.

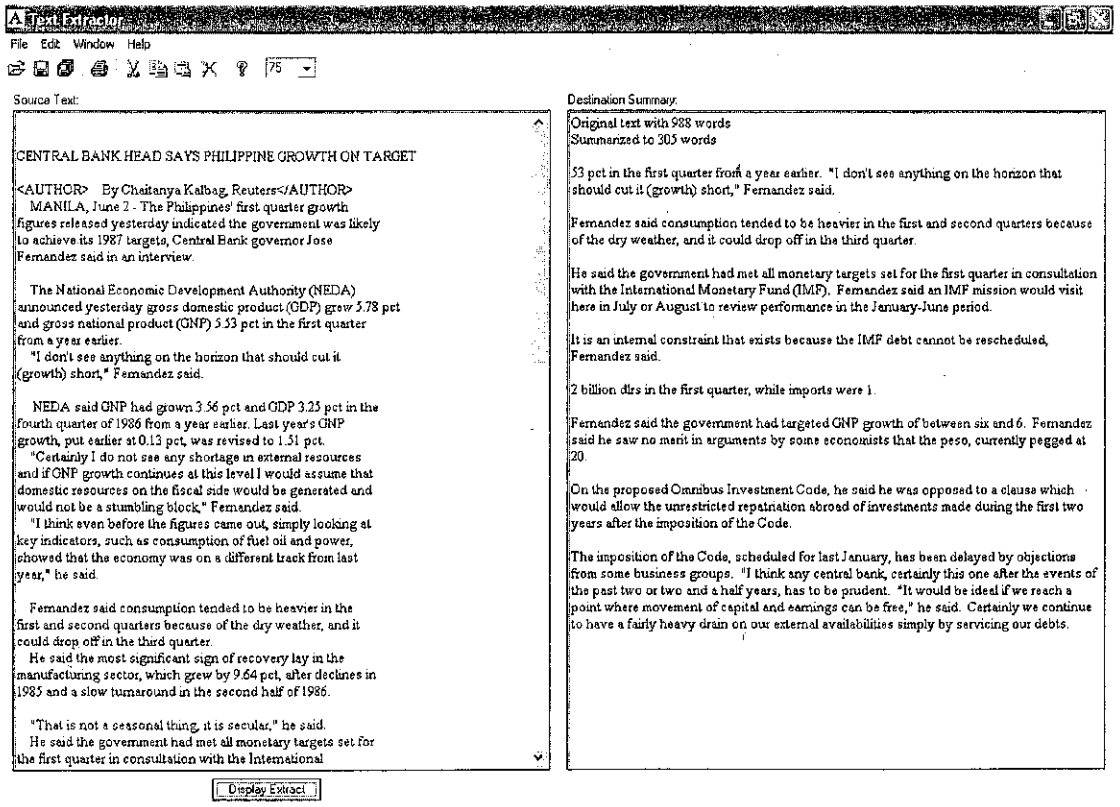


Figure 3.2 The Graphical User Interface of the Text Extractor

The user interface is where documents to be processed are loaded. Users have the ability to determine the length of the summary to be generated. After rigorous processing by the system engine, the summary will be outputted onto the display screen below the original document.

3.4.5 Structure of the Text Extractor program

It is decided that the Text Extractor program will be constructed in a way each word in the document will be classified by the machine learning algorithm. All of the summaries generated are intended to be indicative and generic, which means that they should assist the reader or user in whether they want to read the full source text or not, and that they are not aimed at a specific user-audience, but suitable for all potential readers.

There will only need to be one data structure present in the Text Extractor program, as no evaluation between the extracts and source documents needs to be done at this stage. It is proposed that the Text Extractor data structure will hold the following information:

1. The full document text
2. The document by sentences
3. The document by word
4. The proposed summary lengths in percentage

3.5 Implementation

The following is the algorithm of the Text Extractor program. The program basically consists of four parts, which are explained in the sections below:

3.5.1 Preprocessing Algorithm

Before the classification begins, the loaded text document is preprocessed during the preprocessing stage to remove unnecessary objects from the dataset such as initial white spaces and store words individually as a collection. The algorithm for the preprocessing stage is shown in the box below.

```
Initialize stop word list, hot word lists
Initialize constant variables with probability values
Define collections
Initialize variables
Load source document files, read into buffer
        Contents of buffer changed to lowercase characters
Skip initial white space
        Stop words removed
If a real word
        Text split into sentences collection
        Text split into words collection
If new paragraph
        Increment number of paragraphs
Calculate document's sentence count
        Total added to total real words in source
```

3.5.2 K-Means Algorithm

The K-Means algorithm is called upon to cluster the word collection into groups which have the same attributes. The clusters obtained will then be classified by the SVM algorithm.

Equation 3.1 shows the mathematical function of K-Means algorithm [49]. n represents the total number of clusters whereby in this application, the number of clusters is set to 8 in correspondence to the parts of speech categories. μ is the clusters where the words will be assigned to according to their category while w represents the word.

$$\sum_{k=1}^{n} \mu_k (w) = 1 \quad , \text{where } n = 8 \qquad \textbf{Equation 3.1}$$

```
Initialize database
If word present in the database
        If "VERB", assign word to verb cluster
        If "NOUN", assign word to noun cluster
        If "ADVERB", assign word to adverb cluster
        If "ADJECTIVE", assign word to adjective cluster
        If "PRONOUN", assign word to pronoun cluster
        If "PREPOSITION", assign word to preposition cluster
        If "INTERJECTION", assign word to interjection cluster
        If "CONJUNCTION", assign word to conjunction cluster
Else discard
```

3.5.3 Support Vector Machines (SVM) Algorithm

The SVM algorithm basically takes the clusters generated by the K-Means algorithm and classifies each word in the clusters into corresponding synonyms, which will determine the actual occurrence frequency of each word.

The occurrence frequency (f) of the each word is shown by **Equation 3.2** [50] below:

$$f(w) = \sum_{k=1}^{n} \mu_k [f(w)] = 1 \text{ , where } n = 8 \qquad \textbf{Equation 3.2}$$

```
Initialize database
Initialize variable
Initialize wordcluster collection, μk
For each cluster
    Do until end of each element in wordcluster
            If Data Is Present In database
                    Increment occurrence frequency of word
```

3.5.4 Assembling Algorithm

After obtaining the frequency of each word, the sentences are scored and based on the score calculated; the system will output the extract onto the interface.

```
Initialize variables
Initialize collection
For total number of words
    If word should be acknowledged
        Do until finish word count
            Store word in new collection
For word occurance frequency count
        If word in new collection is within the frequency score range
            Retain word
        Else discard
For each sentence
        Store in new collection
            If word in new collection is within the frequency score range
            Increment sentence score
For each sentence
        Initialize score to 0
```

```
        Store sentences to Sentence collection
        For words contained in the sentence .
                Increment score with Word occurance frequency
                If words contain hot words
                        Increment Score with word sequence count * 0.05
                If Sentence Total Score > 0 Then
                        Score = Score / (Sentence total score * 1)

                If Sentence in Paragraph 1
                        Multiply score by 4
                Else If Sentence in Paragraph 2
                        Multiply score by 3
        Else If Sentence in Paragraph = 3
                        Multiply score by 2
                Else If Sentence in Paragraph > 3
                        Multiply score by 1
        Else set score to 0
For each sentence
        If Sentence Score >= Maximum Score Then
                Increment the Total Real Words in Summary
                Add sentence to summary
Display summary
```

3.6 Tools

3.6.1 Software

Microsoft Visual Basic 6.0 is chosen as the developing tool since it provides the user with the uncomplicated creations of the user interface. Microsoft XP is used as the platform that conforms to the compatibility of Visual Basic.

Text categorization requires that words in the particular document to be summarized, need to be represented independently. To do this, it is best to store the words of document in a database; in columns and rows. The best database to use is Microsoft Access.

3.6.1 Hardware

Executing the SVM algorithms will involve a lot processing power and computer memory. Therefore, a computer with high specifications is needed. Since this application does not involve any connection to the internet, only one computer is needed. Below are the specifications of the computer:

1. Intel Pentium 4 Processor
2. 512MB of RAM
3. 100GB of hard disk capacity

CHAPTER 4

RESULTS AND DISCUSSION

This section of the report contains findings related to the subject matter throughout the product development process until its completion. By identifying the rationalization of the application, a research has been done to gather data from previous results of the implementations of SVM. From previous research, it shows that SVM has a tendency to produce good results in classifying data.

4.1 Evaluation

Since the aim of the evaluator program is to evaluate the effectiveness of the Text Extractor program at creating summaries, it needs measurements to rate itself against. These will be a reference extract, taken from an existing summarizer that is already in the market, the Text Extractor and the Microsoft Word AutoSummarizer. The AutoSummarizer will act as a comparison of a summary in an auto-generated format; one which has already evolved and enhanced from time to time.

Both auto-generated summaries (Text Extractor and AutoSummarizer) will be evaluated against the "gold-standard" reference extract, to produce an overlap calculation, of how many sentences appear in both the auto-generated summary and the reference extract. This will be presented as a percentage. This method of evaluation is an intrinsic method, as it aims to test the quality of the summaries against other summaries or extracts.

Evaluation for the application is carried out by presenting some computational results which are achieved by testing using common datasets that have been used widely for text categorization, and finally, these results are compared to some baseline summarization procedures or reference summary (manually produced summaries by an

expert) that will determine the application's qualitative measure and how well it performs. The evaluation will consider the following information to calculate the overall performance of the Text Extractor:

1. The "gold standard" or reference extract's selected sentences
2. The Text Extractor's selected sentences
3. The AutoSummarizer's selected sentences
4. The overlap between the Text Extractor summary and the reference extract
5. The overlap between the AutoSummarizer summary and the human-generated extract

The sections below address the criteria that are taken into account for the evaluation task.

4.1.1 Performance Measure

Below is the performance measures used for the evaluation of the abstraction application [1]. In simple terms, precision is a measure of the usefulness of the extractor while recall is a measure of the completeness of the extractor. Basically, recall is a measure of how well the engine performs in finding relevant sentences to be included in the abstract whereby it is 100% when every relevant sentence is retrieved. On the other hand, precision is a measure of how well the engine performs in not returning non-relevant sentences and is 100% when every document returned to the user is relevant to the abstract.

$$\text{Precision} = \frac{|\{\text{Relevant sentences}\} \cap \{\text{Retrieved sentences}\}|}{|\{\text{Retrieved sentences}\}|}$$

$$\text{Recall} = \frac{|\{\text{Relevant sentences}\} \cap \{\text{Retrieved sentences}\}|}{|\{\text{Relevant sentences}\}|}$$

$$\text{F-Score} = 2 \text{ x precision x recall x } / \text{ (precision + recall)}$$

4.1.2 Using different compression rates

Different compression rates are used in order to determine at what level of compression does each summarizer performs best with. Each application will have to produce summaries with percentages of 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, and 70.

4.1.3 Evaluating against another text summarizer

The evaluation is done against a currently used application which is the MS Word Summarizer. MS Word Summarizer is a summarizer that is integrated in Microsoft Word which cuts words by counting words and ranking sentences. It identifies the most common words, gives each sentence a score based on the frequency of the words in the document, and finally averages the sentence by dividing the total value of the sentence by the number of words within it. The top scoring sentences are later compiled to become the summary of the desired number of words or as a percentage of the original document length, set by the corresponding user.

4.1.4 Copernic Text Summarizer

To obtain the results of all performance measures, a reference output should be at hand. Previous evaluations had used a human-generated summary, specifically by a language expert, as a reference in obtaining the number of relevant sentences in a particular summary. However, each human-generated summary produced by different experts should yield different results.

Therefore, instead of using the human capability and perhaps address the time constraint of this project, this experiment will resort in another method, whereby

another summarization application that already exists in the market will be used as a reference. Copernic uses both statistical and linguistic algorithms which pinpoints the key concepts and extracts the most relevant sentences, resulting in a document summary that is a condensed version of the original text.

4.1.5 Using a standardized dataset: Reuters-21578

The Reuters-21578 dataset is currently the most widely used test collection for text categorization research and serves as a standard real-world benchmarking corpus. The corpus contains two types of datasets: test and train. Both datasets contain an array of newspaper articles ranging from many sectors in the industry including trade, gold and soy-oil. However, in this experiment, 20 articles will be chosen randomly from 20 different sectors and from these articles, 4 will be used for testing. This is to test how well both applications perform on different data.

4.1.6 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Rouge is an application that has been used by numerous researchers to cut down on testing time. It is Unix-based and it basically compares the output or generated summary of two applications and produces the recall, precision, and the F-Score of both applications.

4.2 Results

The results produced by the Text Extractor and MS Word AutoSummarizer were obtained by summing the number of sentences for all of the extracts and summaries generated and comparing them to the Copernic Text Summarizer, the reference

summarizer. All sentence counts for summaries generated by both applications were conducted in precisely the same way.

4.2.1 Tabular data

Reuters-21578: 0009757						
Compression rate (%)	**Text Extractor**			**AutoSummarizer**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.6658	0.7959	0.7321	0.7525	0.7446	0.7412
10	0.6928	0.6033	0.6419	0.7213	0.6546	0.6621
15	0.683	0.6482	0.6492	0.6785	0.6191	0.6584
20	0.6443	0.6527	0.6128	0.6493	0.6703	0.6316
25	0.5235	0.6269	0.5313	0.672	0.6559	0.5698
30	0.6135	0.5914	0.5068	0.6375	0.6179	0.5381
35	0.5723	0.6421	0.4921	0.6413	0.6466	0.5334
40	0.5723	0.679	0.4511	0.6163	0.6111	0.5042
45	0.552	0.6123	0.4603	0.5128	0.6320	0.496
50	0.5473	0.5886	0.4512	0.5888	0.5888	0.4591
55	0.535	0.59	0.4402	0.5838	0.6023	0.4587
60	0.5168	0.5557	0.42	0.5593	0.5572	0.4158
65	0.4893	0.5648	0.4167	0.546	0.5752	0.4161
70	0.4768	0.5263	0.3733	0.5228	0.5256	0.3786

Table 4.1: The average precision, recall, and F-score for Text Extraction and AutoSummarizer using Reuters-21578 articles

4.2.2 ROC Curve

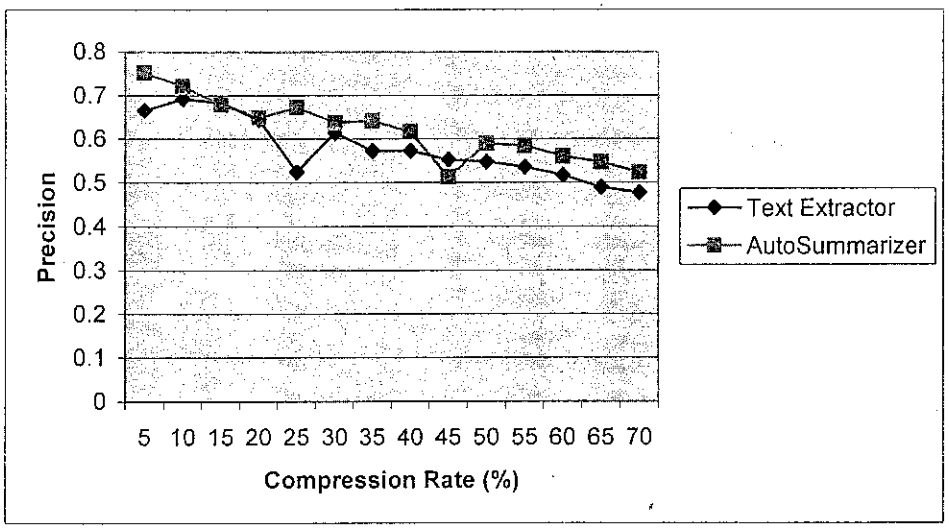


Figure 4.1: The average precision graph for Text Extractor and AutoSummarizer using Reuters-21578 articles

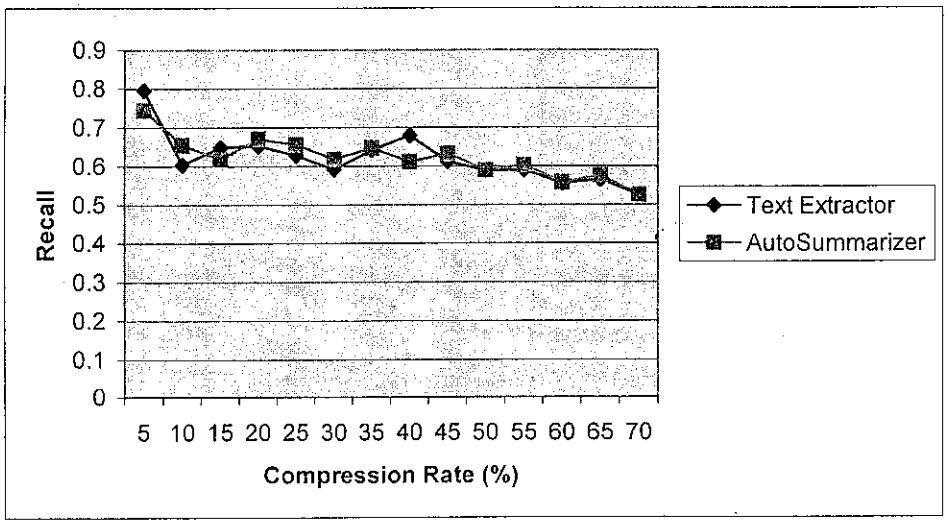


Figure 4.2: The average recall graph for Text Extractor and AutoSummarizer using Reuters-21578 articles

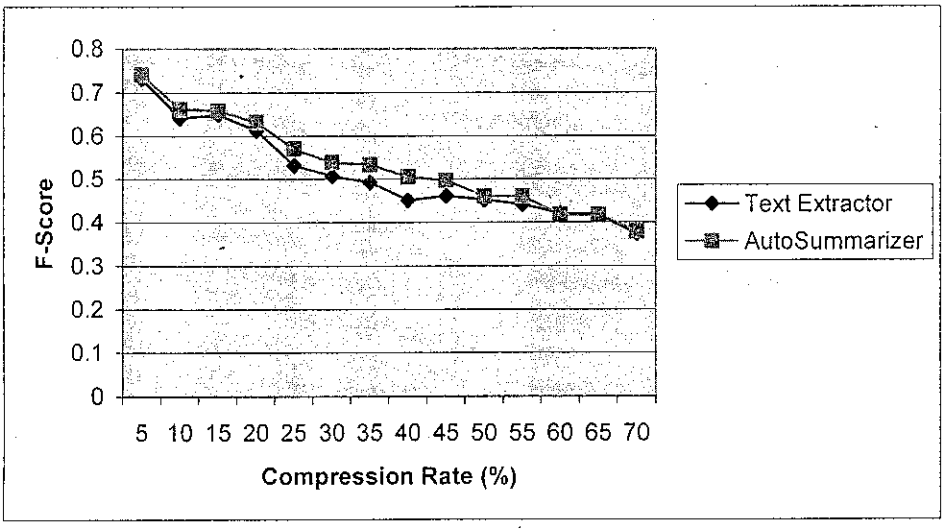


Figure 4.3: The average F-Score graph for Text Extractor and AutoSummarizer using Reuters-21578 articles

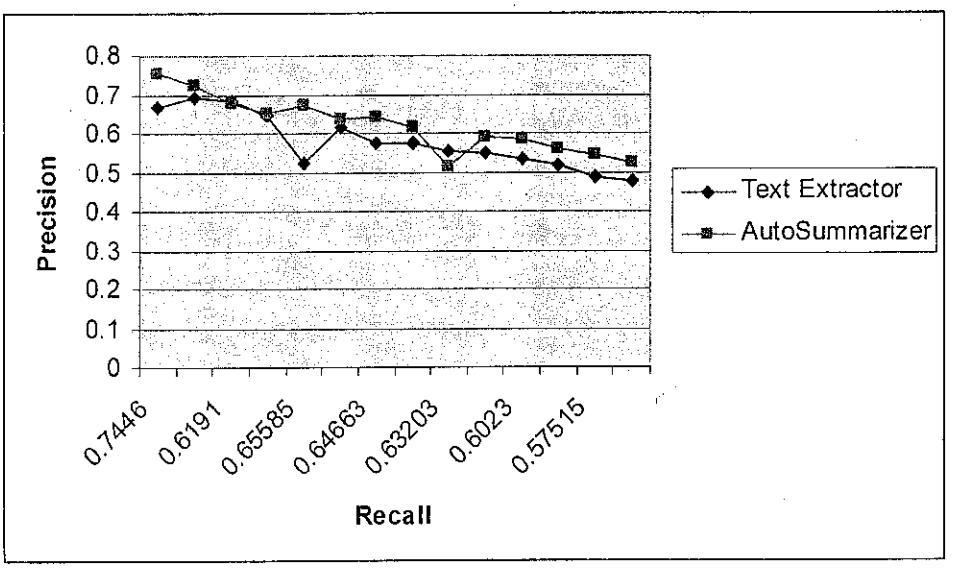


Figure 4.4: The average precision vs. recall graph for Text Extractor and AutoSummarizer using Reuters-21578 articles

The average precision, recall, and f-score shown in **Table 4.1**, indicates that the auto-generated summary nearest to the ideal extract produced by the Copernic Summarizer is the AutoSummarizer. However, the Text Extractor has been noticed to perform better for certain compression rates shown by the ROC Curves in **Figures 4.1, 4.2, 4.3** and

4.4. The rest of the results in tabular form and ROC curves are located in the appendices summary for further references. This is quite a disappointing result, which suggests further improvements to be carried out in order to produce a better performing extractor.

Though there was no other evaluation methods used – the evaluation of all the auto-generated summaries has been based on an overlap of sentences when compared with another established auto-generated extract. This method of evaluation is very effective, but if the Text Extractor summary was aimed at fulfilling an information need, for example to answer a query, then this auto-generated summary might still be able to achieve a purpose.

4.3 Discussion

The development of the application started of with the implementation of the front-end or the user interface which went relatively smooth since the developing tool that was used is Visual Basic 6.0. The next step was the construction of user commands such as the loading of the text document and other operations. The most challenging part was the implementation of the overall algorithm which was quite confusing due to the many processes that needed to be address especially in processing textual data.

A separate program was created in order to test each the sub-functions. It is believed that all the essential sub-functions for this extractor have been met except the specific algorithms that implement SVM. Since it was mentioned at the start of the implementation that code efficiency determination was the major goal, development of the code design will continue until the result of this project is finally obtained.

4.3.1 Evaluation of results

There are definitely explanations as to why these results have been obtained. The first reason is that the combination of features in the system might not have been suitable for this corpus of newspaper articles. Other research using this approach obtained excellent results due to the use of a corpus consisting of scientific papers, presumably with the average length of each paper a lot longer than 20 sentences, which is the average for this corpus [45]. Using a corpus with an average longer document length, is likely to affect the analysis, because the overall structure of the document is likely to be different, and therefore more emphasis can be paid especially to the location factors.

Cue words that were declared in the application to extract important information from the corpus did not quite do the trick. It is recognized that the selection of cue words (bonus terms and stigma terms) used in this summarizer is very poor, and if more time was permitted, an analysis of the overall corpus's word frequencies might have produced a wider set of terms. On the other hand, it might be that a newspaper corpus makes little use of bonus and stigma words; especially since the corpus will be covering a huge spectrum of topics, even if it is all of the same document-type.

Looking at the sample data, as a comparison to the results obtained, it seems that some of the terms used in the newspaper clippings were abbreviated or alternate versions of the way the term would be written in the body of the article. For example, 'Govt' would appear in the headline, whereas 'Government' would appear in the text. Since no synonyms for any terms were provided, this problem could not be overcome. A further detailed study of the corpus would be required to identify these irregularities.

It is concluded that the mixture of trying to identify important sentences for a summary from documents in a newspaper corpus by using machine learning algorithm, did not produce satisfactory results. However, the conclusions which have arisen from the results, suggest that this technique is not suitable for a newspaper corpus and still have a lot of improvements to be made to it especially in terms of research.

4.3.2 Evaluation of Reuters-21578

The Reuters-21578 corpus was designed to be used for the evaluation of both single and multiple document summarization systems and is widely used by numerous researchers for text categorization evaluation. Since this project emphasizes on English-based single-document extracts made it seem like the perfect choice as the corpus for this research project. However, based on the results generated by the Text Extractor summarizer, it appears that the corpus might not have been suitable for the summarization technique used.

The documents and extracts had been encoded in XML, which was not much of a problem since the extractor managed to filter the tags. Sentence splitting had already been carried out; the title had been separated, and the paragraph and relative sentence position of each sentence had been identified. However, in a manual inspection of a sample of the extract generated, it was found that the sentence splitting did not necessarily fall on a full-stop.

Other punctuation symbols were used, such as the colon, and sometimes, due to the structure of the article a sentence could be comprised of a single word. The corpus should not be blamed for the underperformance of the extractor, as it is believed that if the corpus contains constraints or limits, then it must be overcome by the researcher himself.

Having put more time and effort to study the corpus in more depth and conduct a fuller analysis of the corpus, would have believingly identified some constraints which have unsuspectingly affected the performance of the application.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The necessity of having a text extractor nowadays is increasing. This is due to information overload, and methods must be researched in order to manage this. Some commercial summarization tools are available, for example, Microsoft Office products including AutoSummarize, a built-in summarizer. However, there is still room for improvements to be made to produce a better application since there are still no answers to what makes a good summarizer or extractor. The measurement of the effectiveness of extraction applications is itself a large area of research.

Basically, an extractor is an application that reads in a textual document, quantifies and classifies important words, removes unnecessary contents, summarizes it using a certain technique within the chosen summary length, and evaluates its effectiveness against some pre-defined criteria. The research of the effectiveness of text summarization still has a long way to go to ensure better uses of technology in the near future.

The reason why the AutoSummarizer produced evidently better summaries, i.e. nearer to the ideal standard of the human-generated extract than the application developed, could be due to the corpus which was entirely consisting of newspaper articles whereby with newspaper articles it has been found that the most important information is placed nearer to the start of the document. This can be confirmed by the fact that the best individual feature was the one which identified sentences in the first third of the document. It can be suggested that the poor performance of the Text Extractor was due to the combination of the features selected to extract the values with the type of document used.

5.2 Recommendation

There is obviously room for development and improvement within this research project to further enhance and obtain the expected results, as opposed to the unsatisfactory results obtained from this particular project. Below are some recommendations for future developments.

The crucial part of an extractor is to intelligently select the best sentences that will comprise the summary. Therefore, by introducing other features into the project might help extract important sentences. For instance, features such as uppercase words, as included in a research by Kupiec, et al (1995). This project can also be improved by taking into consideration ending of each sentence, whether it is really a full stop or contrariwise.

Another way for improvements is to exclude sentences of fewer than five words. An experiment done by Kupiec, et al [46] showed that short sentences are not used in summaries. Therefore, this suggests that an extractor should ignore and discard sentences having less than five words at the beginning that is during preprocessing. It is obvious that this project focuses more on generic rather than query based summarization. Hence, by adding the use of the Text Extractor summary in answering a query would move into a different area of evaluation which is of extrinsic evaluation, where the summarization system's output is tested in relation to another task, in this case; answering a query, which would further enhance the usability of this application.

It is known that the Text Extractor uses Support Vector Machines (SVM) as the machine learning algorithm. It would be a good idea to consider other machine learning techniques such as the decision tree algorithm and the neural network algorithm. This is to determine whether other algorithms might be suitable for the features chosen and the corpus used for the evaluation.

REFERENCES

[1] Fung, G., and Mangasarian, O.L., "Data selection for support vector machine classifiers", In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (Boston, Massachusetts, United States), ACM Press, 2000, pp. 64 - 70.

[2] Tong, S., and Koller, D., " Support vector machine active learning with applications to text classification", The Journal of Machine Learning Research archive, Vol 2, The MIT Press, March 2002, pp. 45 - 66.

[3] Wu, X., and Srihari, R., " Incorporating prior knowledge with weighted margin support vector machines", In Proceedings of tenth ACM SIGKDD international conference on Knowledge discovery and data mining (Seattle, WA, United States), ACM Press, 2004, pp. 326 - 333.

[4] Gorelick, L., Friedman, J. (July, 2003). "Kernel Methods", Retrieved April 5, 2006, from The Weizmann Institute of Science, Faculty of Mathematics and Computer Science Web Site: http://www.wisdom.weizmann.ac.il/~hassner/cv03/KernelBasedMethods.ppt

[5] Chu, W., Ong, C.J., and Keerthi, S.S., " An improved conjugate gradient scheme to the solution of least squares SVM", IEEE Transactions on Neural Networks, Vol 16, Issue 2, March 2005, pp. 498 - 501.

[6] Li, X., Zhu, Y., and Sung, E., " Sequential bootstrapped support vector machines - a SVM accelerator", In Proceedings of 2005 IEEE International Joint Conference on Neural Networks (IJCNN 2005), Vol 3, July 31-August 4, 2005, pp. 1437 – 1442.

[7] Jia, H., Murphey, Y.L., and Gutchess, D., Chang, T.S., "Identifying knowledge domain and incremental new class learning in SVM", In Proceedings of 2005 IEEE International Joint Conference on Neural Networks (IJCNN 2005), Vol 5, July 31-August 4, 2005, pp. 2742 - 2747.

[8] Namburu, S.M., Tu, H., and Luo, J., Pattipati, K.R., "Experiments on Supervised Learning Algorithms for Text Categorization", In Proceedings of 2005 IEEE Conference on Aerospace, March 5-12, 2005, pp. 1 - 8.

[9] Jimmy, L., Mohamed, Q., "A new method for query generation applied to learning text classifiers", In Proceedings of 2003 IEEE/WIC International Conference on Web Intelligence (WI 2003), October 13-17, 2003, pp. 633 - 636.

[10] Masuyama, T., Nakagawa, H., "Applying cascaded feature selection to SVM text categorization", In Proceedings of the 13th International Workshop on Database and Expert Systems Applications, 2002, September 2-6, 2002, pp. 241 - 245.

[11] Yang, Q., Li, F.M., "Support Vector Machine for Customized Email Filtering based on Improving Latent Semantic Indexing", In Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Vol 6, August 18-21, 2005, pp. 3787 - 3791.

[12] Cai, L., Hofmann, T., "Hierarchical Document Categorization with Support Vector Machines", In Proceedings of the 13th ACM International Conference on Information and Knowledge Management (Washington, D.C., USA), ACM Press, 2004, pp. 78 - 87.

[13] Liu, T.Y., Yang, Y., Wan, H., Zeng, H.J., Chen, Z., Ma, W.Y., "Support Vector Machines Classification with a Very Large-Scale Taxonomy", The source of ACM SIGKDD Explorations Newsletter archive on Natural language processing and text mining, Vol 7, Issue 1, ACM Press, June 2005, pp. 36 - 43.

[14] Burges, C.J.C., "A Tutorial on Support Vector Machines for Pattern Recognition", The source of Data Mining and Knowledge Discovery archive, Vol 2, Issue 2, Kluwer Academic Publishers, June 1998, pp. 121 - 167.

[15] Yang Y. and Liu X., 1999, "A re-examination of text categorization methods". In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999), pp. 42-49.

[16] Bekkerman, R., El-Yaniv, R., Tishby, N., Winter, Y., "On Feature Distributional Clustering for Text Categorization", In Proceedings of the 24th Annual International

ACM SIGIR Conference on Research and Development in Information Retrieval (New Orleans, Louisiana, USA), ACM Press, 2001, pp. 146-153.

[17] Joachims, T., "Text Categorization with Support Vector Machines: Learning with many Relevant Features", In Proceedings of the 10th European Conference on Machine Learning (ECML), Springer Verlag, 1998. http://www.ai.cs.uni-dortmund.de/DOKIMENTE/Joachims_97a.ps.gz

[18] Joachims, J., "A Probabilistic Analysis of the Rocchio Algorithm with tfidf for Text Categorization", In Proceedings of the International Conference on Machine Learning (ICML), 1997.

[19] Salton, G., Buckley, C., "Term Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, 1998, 24(5): pp. 513-523

[20] Vapnik V. N., "The Nature of Statistical Learning Theory", Springer, New York, 1995

[21] Yang, Y., Pedersen, J., "A Comparative Study on Feature Selection in Text Categorization", In Proceedings of the International Conference on Machine Learning (ICML), 1997

[22] Rocchio, J., Relevance Feedback in Information Retrieval, In Salton, G., editor, "The SMART Retrieval System: Experiments in Automatic Document Processing", Prentice-Hall Inc., 1971, pp. 313-323.

[23] Mitchell, T., "Machine Learning", McGraw-Hill, 1997

[24] Yang, Y., "An Evaluation of Statistical Approaches to Text Categorization", Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997

[25] Quinlan, J.R., "C4.5: Programs for Machine Learning", Morgan Kaufmann, 1993

[26] Blanz, V., Scholkopf, B., Bulthoff, H., Burges, C., Vapnik, V., Vetter, T., "Comparison of View-based Object Recognition Algorithms Using Realistic 3D Models", In C. von der Malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff, editors, Artificial Neural Networks-ICANN 1996, Berlin, 1996, Springer Lecture Notes in Computer Science, Vol. 1112, pp. 251-256.

[27] Osuna, E., Freund, R., and Girosi, F., "Training Support Vector Machines: An Application to Face Detection", In Proceedings with IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 130-136.

[28] Schmidt, M., "Identifying Speaker with Support Vector Networks", In Interface 1996 Proceedings, Sydney, 1996.

[29] Shanahan, J.G., Roma, N., "Boosting support vector machines for text classification through parameter-free threshold relaxation", In Proceedings of the twelfth International Conference on Information and Knowledge Management (New Orleans, LA, USA), ACM Press, 2003, pp. 247 - 254.

[30] Robertson, S.E., Soboroff, I., "The TREC 2001 Filtering Track Report", The 10th Text Retrieval Conference (TREC-2001), 2002, pp. 26 – 37

[31] Rose, K., "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems", In Proceedings of the IEEE 86 (1998), no. 11, pp. 2210-2238.

[32] Salton, G., and McGill, "Introduction to Modern Information Retrieval", McGraw Hill, 1983.

[33] Marcelo Barros de Almeida, Antonio de Padua Braga, Joao Pedro Braga, "SVM-KM: Speeding SVMs Learning with a priori Cluster Selection and k-Means," *sbrn*, p. 162, VI Brazilian Symposium on Neural Networks (SBRN'00), 2000

[34] V. Guralnik and G. Karypis, "Workshop on Data Mining in Bioinformatics", pp. 73-80, 2001

[35] Andrew W.M, "K-Means and Hierarchical Clustering", 2004

[36] E. Wiener, J.O. Pedersen, and A.S. Weigend, "A neural network approach to topic spotting", In Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95), 1995

[37] Y. Yang, C.G. Chute, "An example-based mapping method for text categorization and retrieval", ACM Transaction on Information Systems (TOIS), 12(3):252{277, 1994

[38] Joachims, T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", In European Conference on Machine Learning (ECML), 1998

[39] Tsuruoka, Y., Kawaguchi-shi, Tsujii, J., "Journal of Biomedical Informatics archive", Vol. 37(6), pp. 461-470, 2004

[40] Yang, Y., Liu, X., "A re-examination of text categorization methods" In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and development in Information Retrieval, pp. 42 – 49, 1999

[41] I. Mani and M.T. Maybury (eds.), "Advances in automatic text summarization", pp. 111-121. Cambridge, Massachusetts: MIT Press.

[42] Teufel, S. and Moens, M. 1997. "Sentence extraction as a classification task", In ACL/EACL-97 Workshop on Intelligent and Scalable Text Summarization, Madrid, Spain, 1997

[43] Firmin, T. and Chrzanowski, M.J., "An evaluation of automatic text summarization systems", Advances in automatic text summarization, I. Mani and M.T. Maybury (eds.), 325-336. 1999, Cambridge, Massachusetts: MIT Press

[44] Jing, H., Barzilay, R., McKeown, K., and Elhadad, M. 1998. "Summarization evaluation methods: Experiments and analysis", Working Notes of the Workshop on Intelligent Text Summarization, 60-68, Menlo Park, California: American Association for Artificial Intelligence Spring Symposium Series.

[45] Paice, C.D. 1981. "The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases", Information Retrieval Research, R.N. Oddy (eds.) London, Butterworths

[46] Kupiec, J., Pedersen, J. and Chen, F. 1995. "A trainable document summarizer", Advances in automatic text summarization, I. Mani and M.T. Maybury (eds.), 55-60. Cambridge, Massachusetts: MIT Press.

[47] Zhou L., Ticrea M., Hovy E., "Multi-document Biography Summarization using SVM", Information Sciences Institute, Marina Del Ray

[48] Marcelo Barros de Almeida, Antonio de Padua Braga, Joao Pedro Braga, "SVM-KM: Speeding SVMs Learning with a priori Cluster Selection and k-Means", pp. 162, VI Brazilian Symposium on Neural Networks (SBRN'00), 2000.

[49] Can, F., Ozkarahan, E. A. (1990) "Concepts and effectiveness of the cover coefficient-based clustering methodology for text databases." ACM Transactions on Database Systems, 15 (4) 483-517

[50] Neto, J. L., Santos, A. D., Kaestner, C. A. A., and Freitas, A. A. (2000). "Document clustering and text summarization", In Proceedings of the 4th International Conference on Practical Applications of Knowledge Discovery and Data Mining (PADD-2000), 41-55, London: The Practical Application Company.

APPENDICES

APPENDIX A

GRAPHICAL INTERFACE OF THE PROCESS OF K-MEANS ALGORITHM

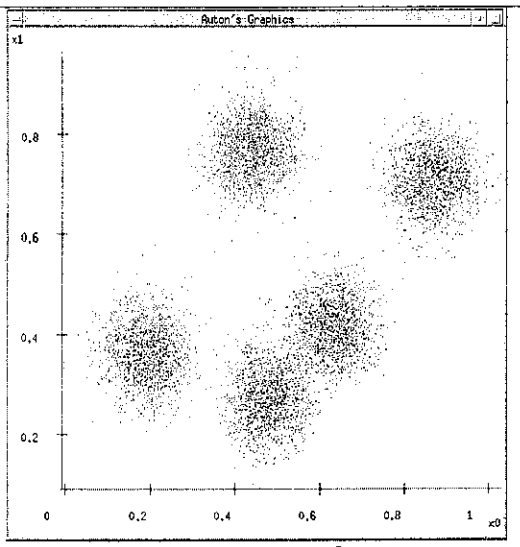


Figure A: The blue dots on the feature space are represented as data points

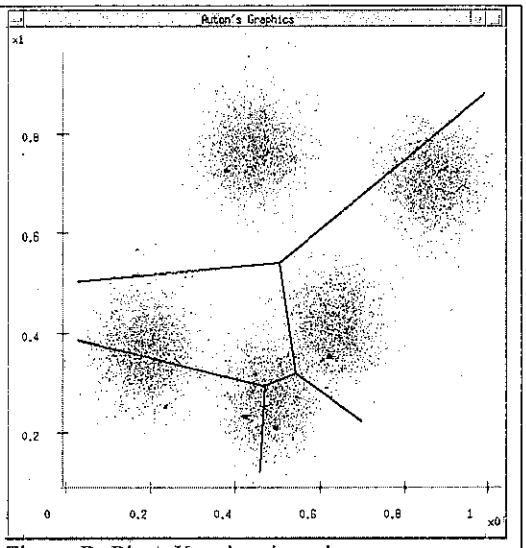


Figure B: Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids

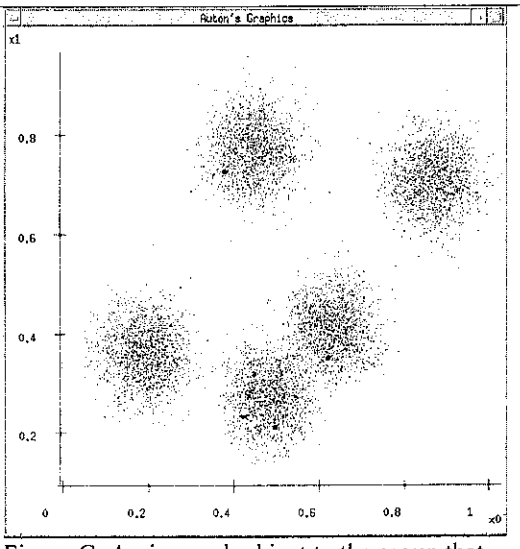


Figure C: Assign each object to the group that has the closest centroid to itself

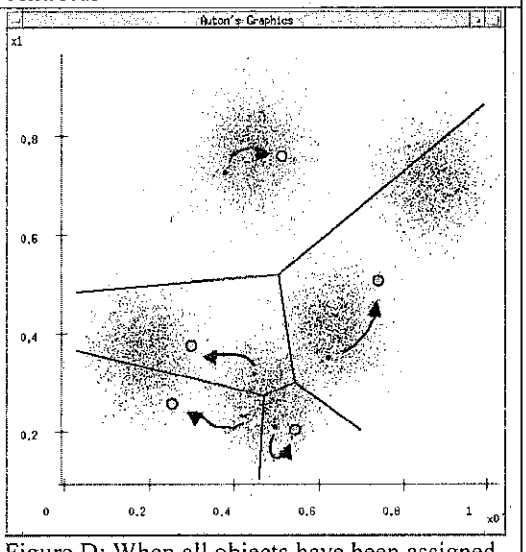


Figure D: When all objects have been assigned, recalculate the positions of the K centroids

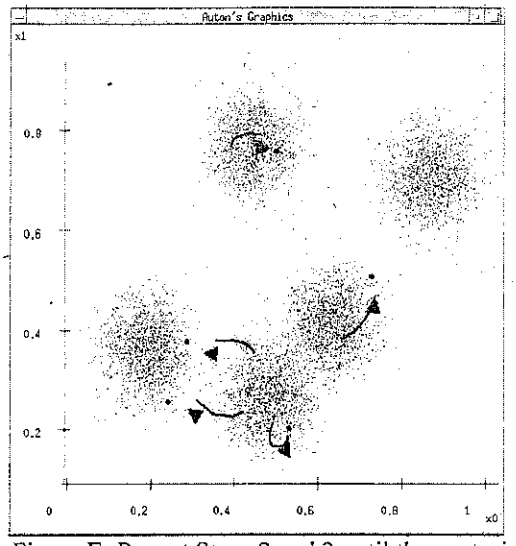


Figure E: Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated

Figure 2.4 Process of K-Means Algorithm [35]

APPENDIX B

EXPERIMENTAL RESULTS OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) USING REUTERS-21578 ARTICLE 0009757

Reuters-21578: 0009757						
Compression rate (%)	**Text Extractor**			**AutoSummarizer**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.63	0.43	0.48	0.74	0.39	0.47
10	0.70	0.33	0.41	0.69	0.33	0.42
15	0.71	0.47	0.54	0.66	0.40	0.47
20	0.68	0.35	0.43	0.62	0.36	0.42
25	0.69	0.40	0.47	0.69	0.42	0.49
30	0.69	0.33	0.42	0.67	0.33	0.41
35	0.68	0.47	0.51	0.71	0.42	0.49
40	0.63	0.37	0.44	0.71	0.35	0.44
45	0.68	0.41	0.47	0.69	0.46	0.51
50	0.68	0.37	0.45	0.69	0.34	0.43
55	0.66	0.41	0.48	0.70	0.42	0.50
60	0.68	0.33	0.43	0.66	0.29	0.38
65	0.65	0.42	0.48	0.69	0.43	0.51
70	0.73	0.37	0.45	0.69	0.31	0.39

Table 4.2: The precision, recall, and F-score for Text Extraction and AutoSummarizer using article 0009757

APPENDIX C

EXPERIMENTAL RESULTS OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) USING REUTERS-21578 ARTICLE 0012249

Reuters-21578: 0012249						
Compression rate (%)	**Text Extractor**			**AutoSummarizer**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.61	0.9013	0.7931	0.726	0.8267	0.8408
10	0.6	0.6975	0.7518	0.697	0.7768	0.7711
15	0.618	0.6789	0.7094	0.631	0.7094	0.7294
20	0.609	0.6529	0.703	0.629	0.7112	0.7141
25	0.61	0.6793	0.5368	0.617	0.7016	0.5876
30	0.566	0.65	0.52	0.605	0.6994	0.5798
35	0.548	0.6933	0.5013	0.596	0.6945	0.5575
40	0.545	0.6692	0.469	0.549	0.6893	0.5462
45	0.478	0.6811	0.4652	0.524	0.6907	0.4993
50	0.47	0.6714	0.4618	0.517	0.6874	0.4737
55	0.481	0.675	0.4327	0.51	0.6901	0.4482
60	0.441	0.668	0.411	0.494	0.6767	0.4331
65	0.415	0.648	0.4109	0.483	0.6666	0.4098
70	0.391	0.6121	0.3918	0.449	0.6459	0.401

Table 4.3: The precision, recall, and F-score for Text Extraction and AutoSummarizer using article 0012249

APPENDIX D

EXPERIMENTAL RESULTS OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) USING REUTERS-21578 ARTICLE 0011164

Reuters-21578: 0011164						
Compression rate (%)	Text Extractor			AutoSummarizer		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.793	0.931	0.791	0.804	0.919	0.793
10	0.771	0.852	0.654	0.797	0.857	0.631
15	0.754	0.814	0.638	0.763	0.841	0.63
20	0.691	0.983	0.625	0.728	0.806	0.607
25	0.667	0.755	0.587	0.691	0.76	0.594
30	0.609	0.732	0.562	0.615	0.736	0.561
35	0.598	0.691	0.516	0.61	0.722	0.502
40	0.565	0.972	0.475	0.572	0.683	0.48
45	0.551	0.669	0.456	0.566	0.671	0.465
50	0.549	0.623	0.45	0.551	0.636	0.451
55	0.512	0.586	0.431	0.538	0.609	0.428
60	0.486	0.545	0.429	0.509	0.562	0.424
65	0.433	0.541	0.391	0.491	0.555	0.356
70	0.346	0.498	0.342	0.452	0.515	0.318

Table 4.4: The precision, recall, and F-score for Text Extraction and AutoSummarizer using article 0011164

APPENDIX E

EXPERIMENTAL RESULTS OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) USING REUTERS-21578 ARTICLE 0012866

Reuters-21578: 0012866						
Compression rate (%)	**Text Extraction**			**AutoSummarizer**		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.63	0.9214	0.8642	0.74	0.8427	0.8608
10	0.7	0.5338	0.7518	0.701	0.6544	0.8261
15	0.65	0.6299	0.7094	0.66	0.6742	0.8043
20	0.649	0.6248	0.6932	0.62	0.8041	0.7853
25	0.61	0.6733	0.5312	0.69	0.7418	0.6074
30	0.589	0.6537	0.5189	0.66	0.706	0.6014
35	0.59	0.714	0.4412	0.649	0.75	0.5841
40	0.549	0.7047	0.4604	0.634	0.7219	0.5504
45	0.499	0.6892	0.4499	0.611	0.7064	0.5097
50	0.49	0.69	0.4456	0.597	0.6919	0.4816
55	0.487	0.6891	0.4171	0.587	0.6901	0.4587
60	0.46	0.6799	0.41	0.574	0.7	0.4262
65	0.459	0.65	0.3847	0.52	0.649	0.4095
70	0.44	0.6249	0.3095	0.5	0.6316	0.4055

Table 4.5: The precision, recall, and F-score for Text Extraction and AutoSummarizer using article 0012866

APPENDIX F

ROC CURVE ON PRECISION AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0009757

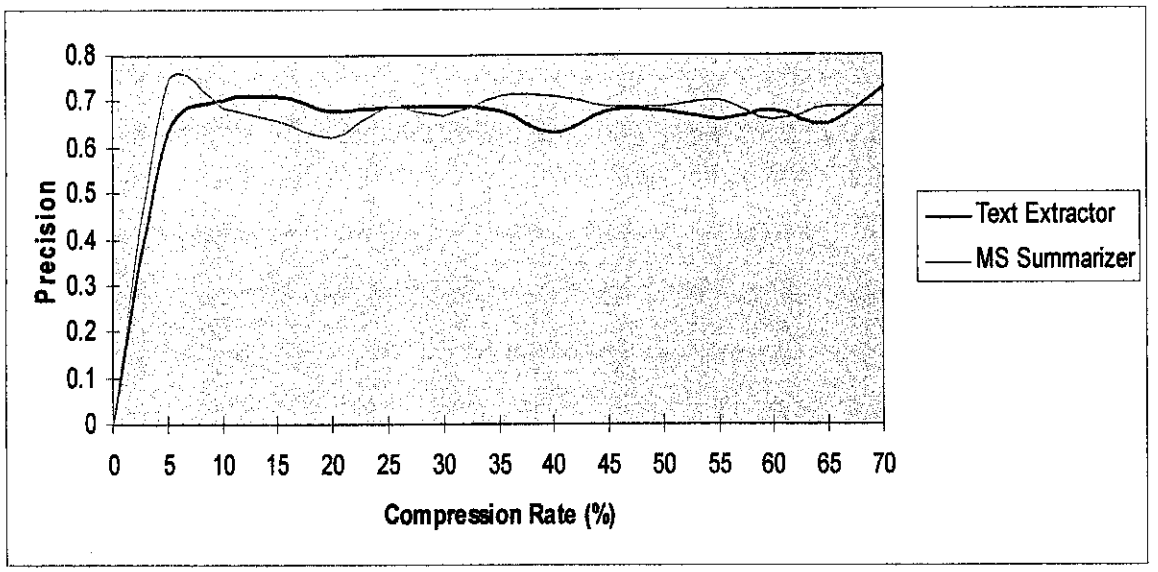


Figure 4.5: The precision graph for Text Extractor and AutoSummarizer for Article 0009757

APPENDIX G

ROC CURVE ON RECALL AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0009757

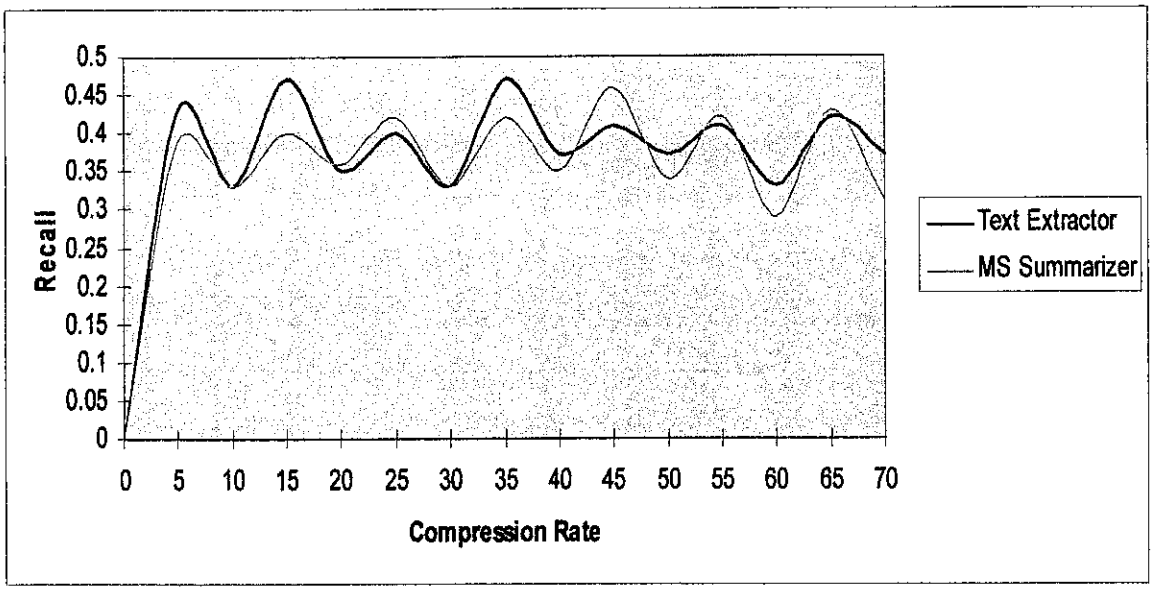


Figure 4.6: The recall graph for Text Extractor and AutoSummarizer for Article 0009757

APPENDIX H

ROC CURVE ON F-SCORE AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0009757

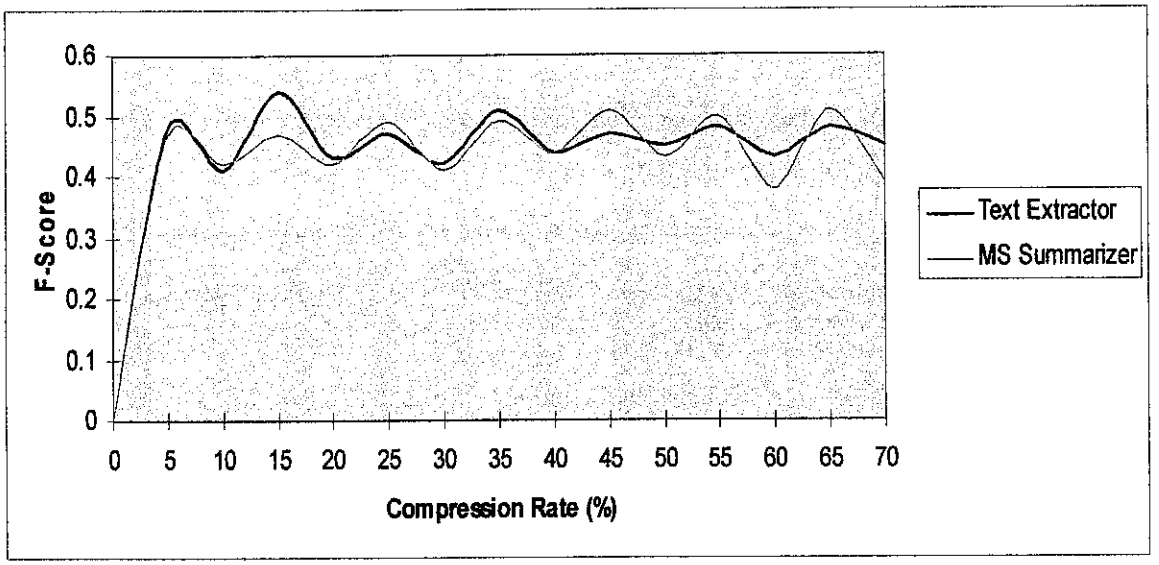


Figure 4.7: The F-Score graph for Text Extractor and AutoSummarizer for Article 0009757

APPENDIX I

ROC CURVE ON PRECISION AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012249

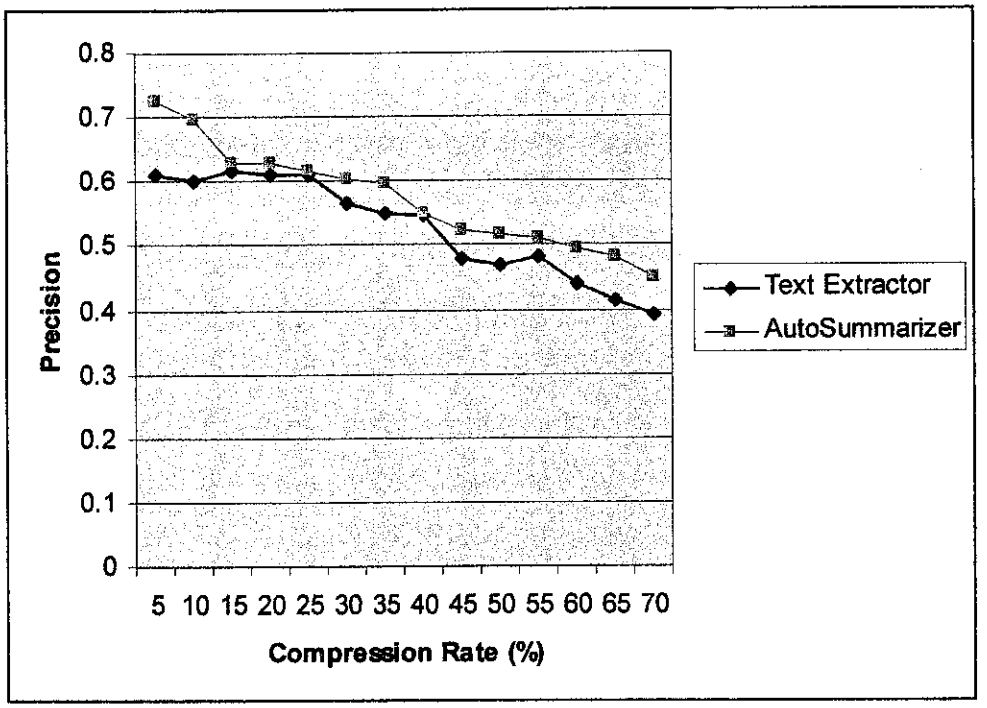


Figure 4.8: The precision graph for Text Extractor and AutoSummarizer for Article 0012249

APPENDIX J

ROC CURVE ON RECALL AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012249

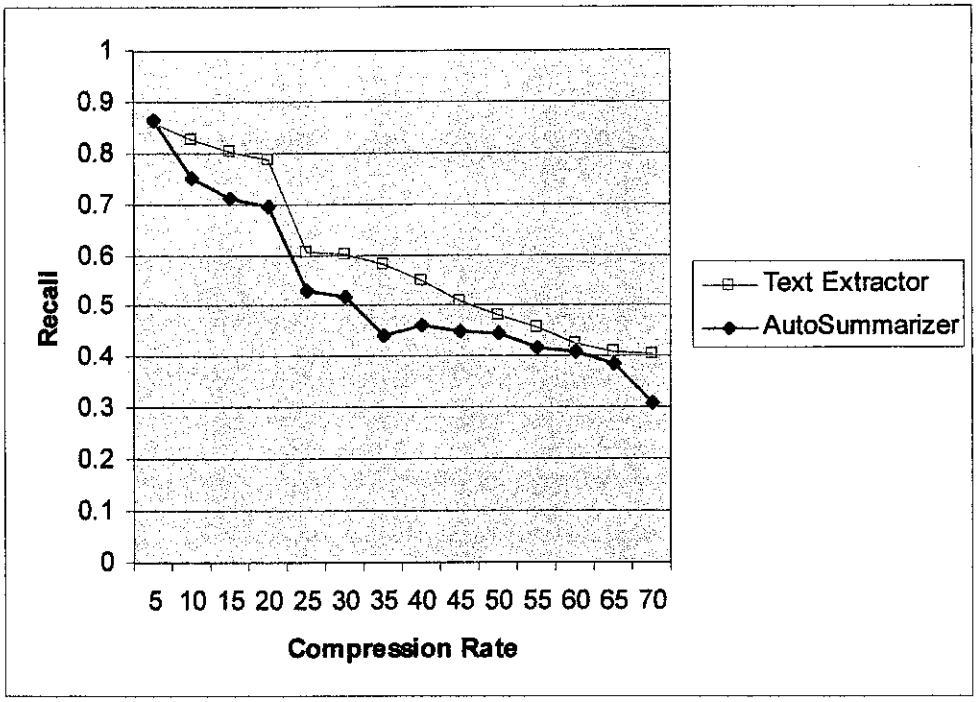


Figure 4.9: The recall graph for Text Extractor and AutoSummarizer for Article 0012249

APPENDIX K

ROC CURVE ON F-SCORE AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012249

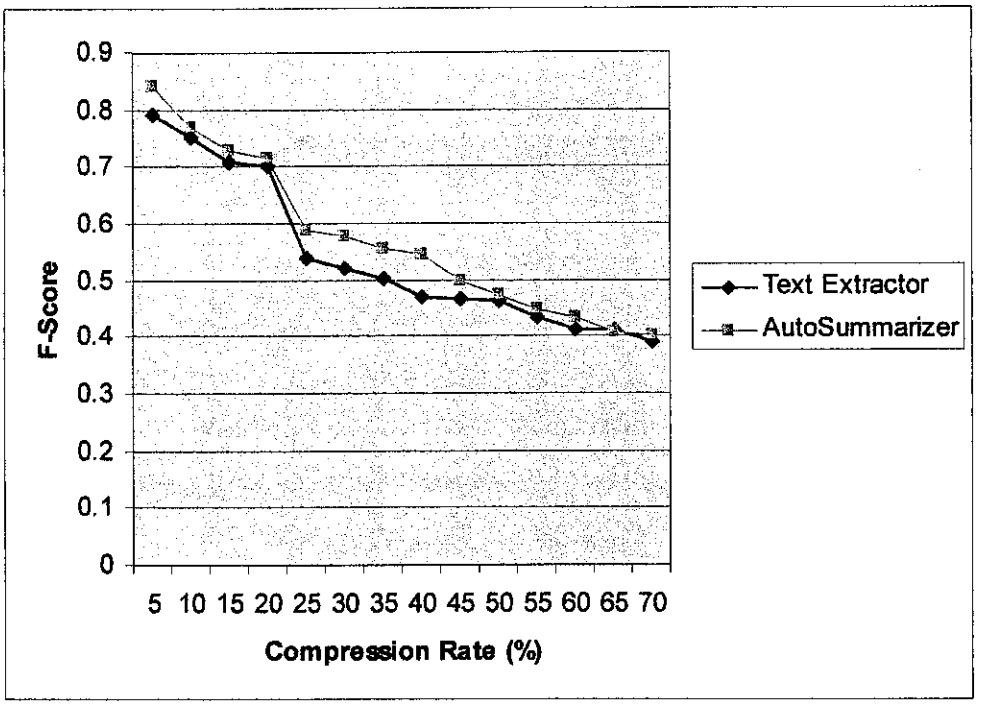


Figure 4.10: The f-score graph for Text Extractor and AutoSummarizer for Article 0012249

APPENDIX L

ROC CURVE ON PRECISION AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0011164

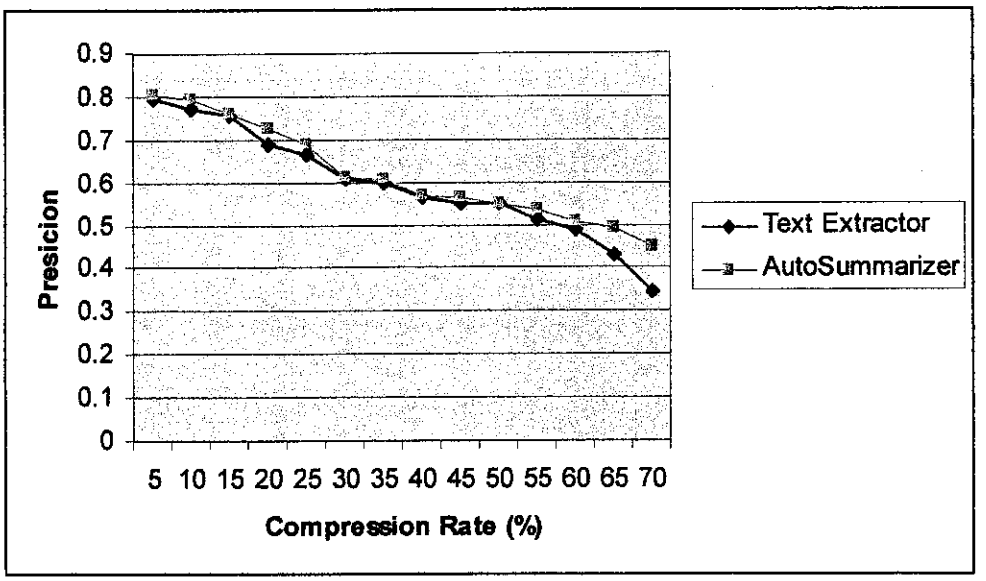


Figure 4.11: The precision graph for Text Extractor and AutoSummarizer for Article 0011164

APPENDIX M

ROC CURVE ON RECALL AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0011164

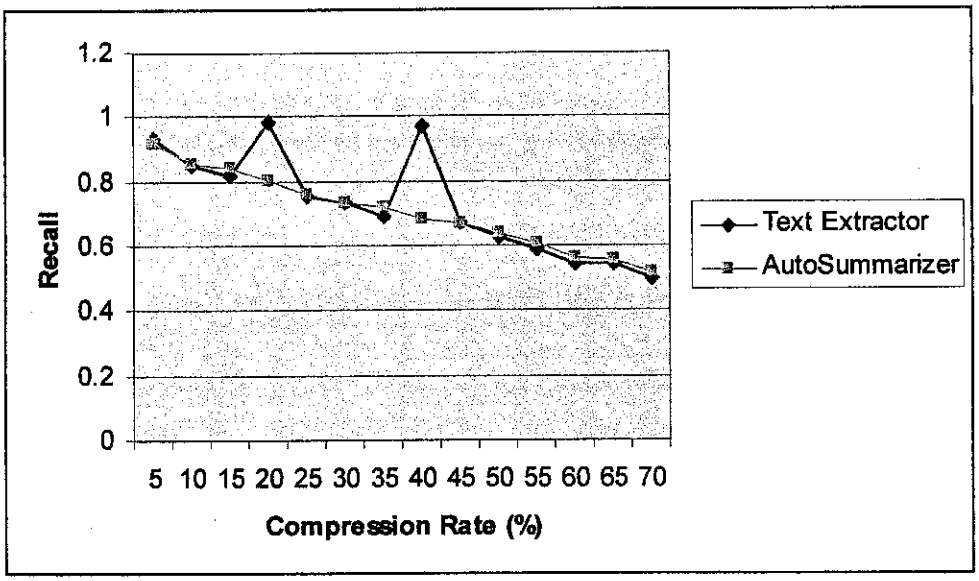


Figure 4.12: The recall graph for Text Extractor and AutoSummarizer for Article 0011164

APPENDIX N

ROC CURVE ON F-SCORE AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0011164

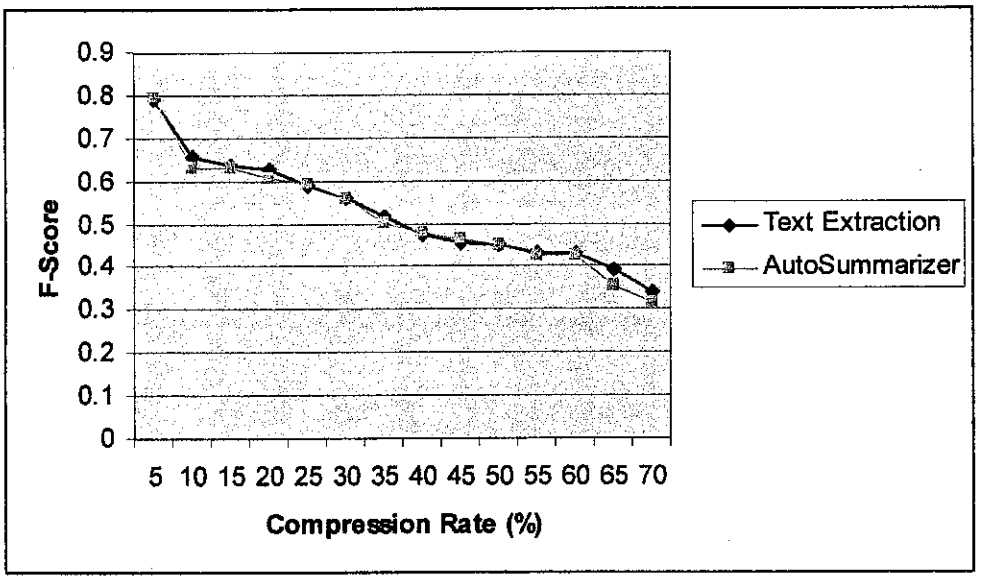


Figure 4.13: The f-score graph for Text Extractor and AutoSummarizer for Article 0011164

APPENDIX O

ROC CURVE ON PRECISION AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012866

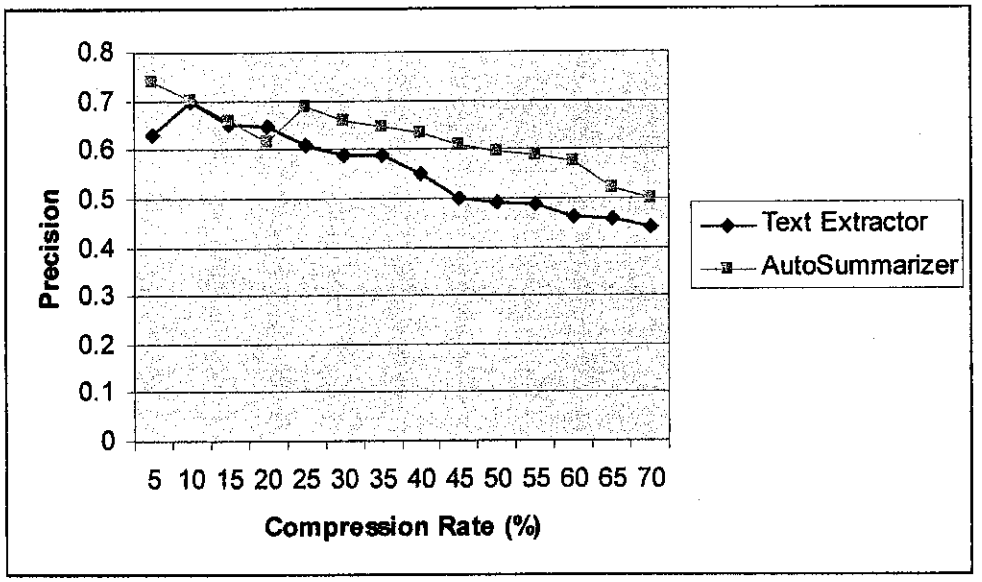


Figure 4.14: The precision graph for Text Extractor and AutoSummarizer using Article 0012866

APPENDIX P

ROC CURVE ON RECALL AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012866

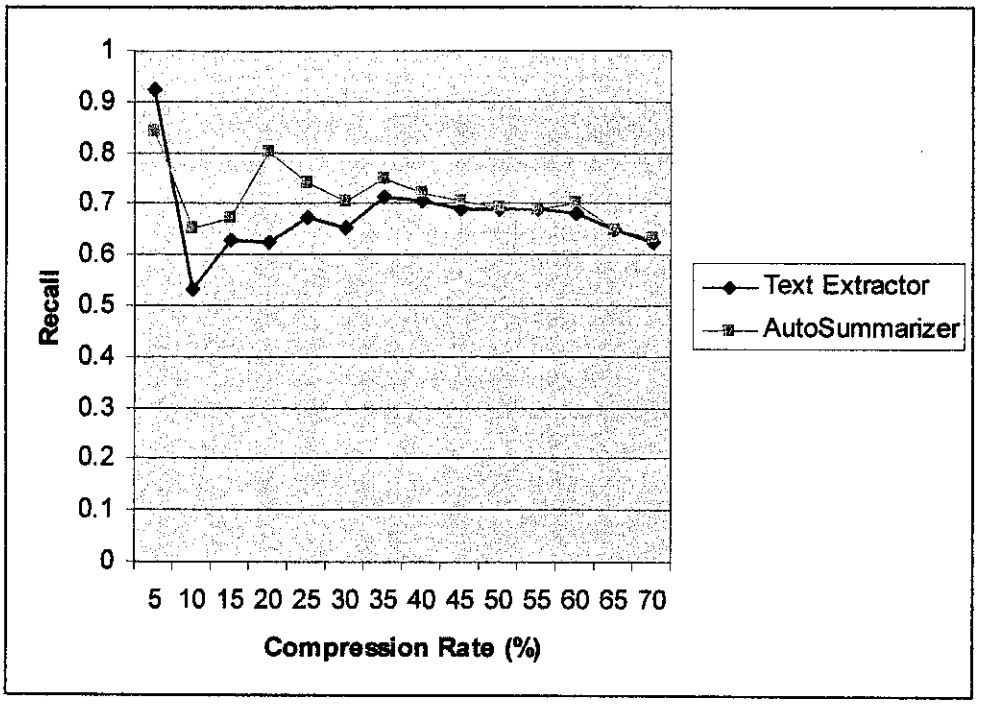


Figure 4.15: The recall graph for Text Extractor and AutoSummarizer using Article 0012866

APPENDIX Q

ROC CURVE ON F-SCORE AND COMPRESSION RATE OF AUTO-GENERATED SUMMARIZERS (TEXT EXTRACTOR & AUTOSUMMARIZER) FOR REUTERS-21578 ARTICLE 0012866

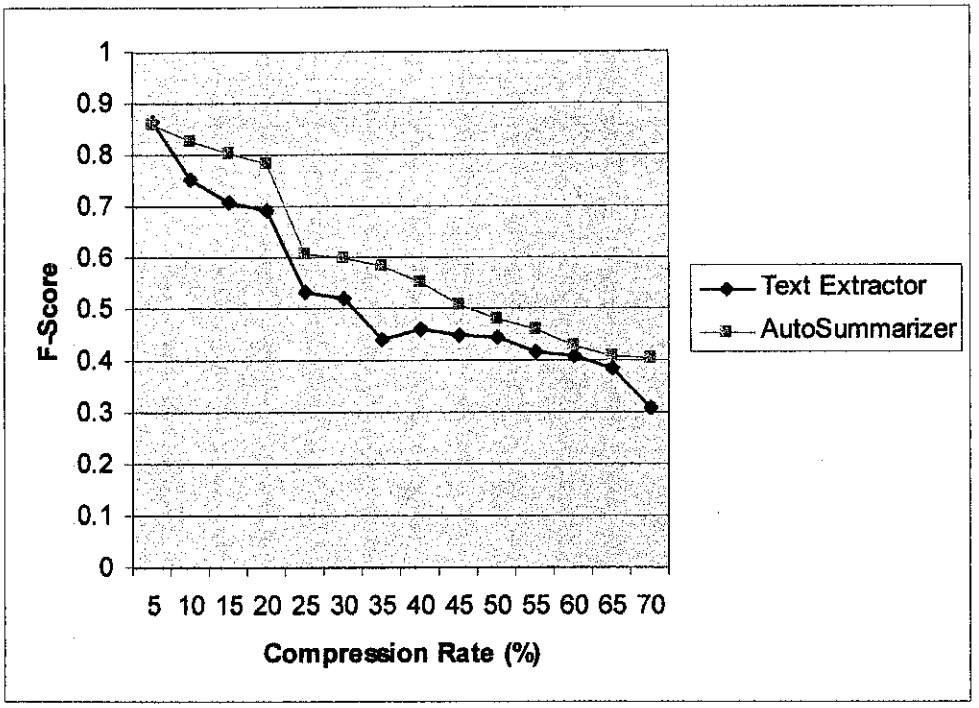


Figure 4.16: The F-Score graph for Text Extractor and AutoSummarizer using Article 0012866