M-INFOSIFT: A GRAPH-BASED APPROACH FOR MULTICLASS DOCUMENT CLASSIFICATION

by

ARAVIND VENKATACHALAM

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2007

To my parents and friends who have always encouraged me to strive for the best.

ACKNOWLEDGEMENTS

I express my sincere gratitude and thankfulness to Dr. Sharma Chakravarthy for his constant motivation, support and guidance through out this research work. Without his persistent help and advice, this work would not have been complete. I would also like to thank Dr. Mohan Kumar and Dr. Nan Zhang for taking their time and serving on my committee.

I am extremely grateful to Dr. Raman Adaikkalavan and Hari hara Subramanium Chelladurai for continually and convincingly conveying the spirit of adventure in regard to research and introducing ITLAB. I would also like to thank Aditya Telang and Roochi Mishra for their support and help. I must also appreciate all my current and ex ITLABians for their encouragement.

I am grateful to my parents Mr.Venkatachalam and Mrs. Dhanalakshmi Venkatachalam and sister Aruna Venkatachalam for offering their constant love and support. Without their endurance and encouragement, this work would not be even possible. Last but not the least, I thank all my friends (Raaji, Aravind Vummidi, Jaiki, Suresh Kumar, Priya, Bindu and Akarsha) for their love and support. I thank the Almighty for the infinite grace and benevolence.

July 20, 2007

ABSTRACT

M-INFOSIFT: A GRAPH-BASED APPROACH FOR MULTICLASS DOCUMENT CLASSIFICATION

Publication No. _____

ARAVIND VENKATACHALAM, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Sharma Chakravarthy

With the increase in the amount of data being introduced into the Internet on a daily basis, the problem of managing these large amount of data is an unavoidable problem. The area of document classification has been examined, explored and experimented as a technique for organizing and managing vast repositories of electronic documents such as emails, text and web pages. Over the past decade, several approaches such as machine learning, data mining, information retrieval and others have been proposed for addressing this problem of classifying electronic documents. While a majority of these techniques rely on extracting high-frequency keywords, they ignore the aspect of extracting groups of related keywords. Additionally, they fail to capture the salient relationships between a number of keywords and their inherent structure, which can prove to be a decisive element in classifying specific types of documents (e.g., web-pages). To this effect, the design of *InfoSift* was proposed which incorporates graph mining techniques for document classification by using a supervised learning model. Perhaps for the first time it was shown how the structure within a document can be used for classification. It was also shown that the techniques can be applied to different types of documents, such as text, email, and web. This framework focused on identifying representative substructures

using graph mining approach and to classify an incoming unknown document to a folder using a ranking mechanism.

However, in the real world, documents are categorized into multiple folders based on varied characteristics (such as multiple folders for different emails or multiple classes for documents). Existing approaches have not used structural relationships with in a document for classification and are based on the occurrence of words. Adopting these approaches within the *InfoSift* framework do not lead to a feasible solution due to the consideration of group of keywords and their relationships with other words. In order to bridge this gap between the strength of *InfoSift* and issues of Multi-folder classification, a different technique needs to be investigated.

Hence, in this thesis, we introduce a new approach to extend the abilities of *InfoSift* to support Multiple categories (folders). A ranking technique to order the representative - common and recurring - structures generated from pre-classified documents to categorize new incoming documents has been presented. This approach is based on a global ranking model that incorporates several factors regarding document classification and overcomes numerous problems while using existing approaches for multiple folder classification in the *InfoSift* system. A number of parameters which influence the generation of representative substructures in single folder classification are analyzed, re-examined, and adapted to multiple folders. Additional graph representations have been analyzed and their use has been validated experimentally. Exhaustive experiments substantiating the selection of parameters for classification of unknown documents into multiple folders have been conducted for text, emails and web pages.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Data collection and information management has always been one of the major concerns in many application domains. As data in different forms continues to grow at an alarming rate, the problem of extracting relevant aspects of this data and storing for future retrieval become more prominent. The issue is further compounded on the World Wide Web where determining relevant information from diverse and vast data sources is more complicated owing to the heterogeneity of the data. Even though many approaches have been proposed for this purpose, active research is still going on in order to go beyond 'indices'.

Information management is a critical task owing to the inter-dependence of several applications that require relevant information in different structure and forms. Instant access to large amounts of information available through the Internet entails a need for mechanisms that determine the relevance of information being accessed. One of the prominent ways for finding the relevancy is adopted by modern day search-engines (such as Google) to do a simple lookup of a 'keyword' (specified by the analyst or user) in these data sources. But these conventional techniques (or similar ones proposed by the Information Retrieval community) do not always bring out all the necessary details when processing of data is needed with respect to a particular context. Additionally, the non-traditional nature of data means that the traditional approaches can not be applied even if the size of data is relatively small. For instance, if the intent is to bring out information regarding the programming language Java, simply providing the keyword 'java' might result in irrelevant information (such as 'java' as in a type of coffee or name of an island, etc.). Providing additional information for processing and extracting relevant features can circumvent this problem. In other cases, management of information

might be as complicated as generating summary of the extracted information. Another mechanism for information management would be to employ data mining techniques such as classification, clustering, etc.

1.1 Data Mining

Data Mining, also know as Knowledge-Discovery in Databases, is the process of automatically searching large volumes of data for useful patterns that might otherwise be unknown. Some other definitions of data mining are:

'The nontrivial extraction of implicit, previously unknown, and potentially useful information from data'.

'The science of extracting useful information from large data sets or databases'.

Data mining has been used to extract interesting patterns or features of information from large amounts of data. It includes analyzing the data (pre-processing of data), finding relevant frequent patterns and summarizing data (post-processing of results). Data mining tasks are mainly divided into two categories [1]

- *Predictive tasks:* The main objective is to predict the value of an attribute based on already known values of other attributes.
- *Descriptive tasks:* The main objective is to derive patterns that lends information in order to derive the underlying relationships in data.

Based on the main techniques used in data mining AS shown in Figure 1.1, ***Predictive Modeling*** refers to the process of building a model for the target variable as a function of other variables. The main two types of predictive modeling are *Classification* and *Regression*. Classification is used for target variables that are binary valued. For example, predicting whether a customer will purchase the item or not in a retail store. Regression is used for continuous valued target variables. For example, forecasting the future price of an item. The goal of both the tasks is to minimize the error between the predicted and true values of the target variable, ***Association Analysis*** is to dis-

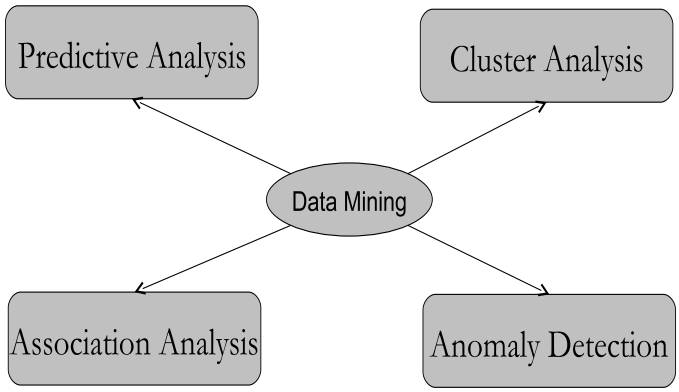

Figure 1.1 Main Techniques in data mining

cover patterns that describe the features associated with the data which are typically represented in the form of rules. The main goal of association is to find the most interesting patterns in an efficient manner. For example, finding web pages that are accessed together, ***Cluster Analysis*** is to find groups of closely related characteristics so that characteristics that belong to the same cluster are more similar to each other than characteristics that belong to other clusters. For example, grouping news articles based on their contents, and ***Anomaly Detection*** refers to the task of identifying observations whose characteristics are significantly different from the rest of the data. These observations are known as outliers. One example of this task is to detect fraudulent transactions in credit card records. Additionally, data mining can be categorized into several branches such as – structured data mining, unstructured data mining, unsupervised learning and supervised learning. In this thesis, we consider 'classification' using supervised learning and act as a mechanism for information management.

1.2 Overview of Classification

The problem of classification involves the process of learning relevant features or attributes of a class and using the same to determine if a new sample belongs to that class. Pre-classified examples in classes are used as a training set to build a descriptor for each class. To determine the destination class for an unknown sample, it is compared

with the descriptors of all classes and categorized where the similarity is maximal. The classification technique analyzes records that are already known to belong to a certain class, and creates a profile for a member of that class from the common characteristics of the records. This can then be used to apply to new records, that is, records that have not yet been classified. This enables us to predict if the new records belong to that particular class or not by checking on similarity between the new records and the descriptors. It allows us to retrieve similar objects easily, as they are grouped together and also it does enable us to search effectively for a particular sample.

A practical scenario would be a consumer company seeking to maximize sales of a new item. User behavior corresponding to a class of customers, who in the past have availed such offers can be learnt to derive relevant attributes. Subsequently, spending patterns of new customers can be compared with what has been learnt to determine if they are potential customers for target marketing. In the view of information management, classification allows retrieval of similar objects seamlessly, as they are grouped together. It also enables analysts to search effectively for a particular sample.

This thesis aims at applying a novel approach based on graph mining to solve the problem of classification, in particular, classifying unknown samples across different classes. Text, email and web page repositories have been considered for our work. Details about graph mining are given in Chapter 3. In the following sections, we will discuss the various issues of text, emails and web page classification and inherent challenges of the domain.

1.3 Problem Domain

The belief that the process of classification or supervised learning (that entails grouping of related or similar entities) can benefit from the application of data mining techniques, has prompted this research direction. We have chosen data mining approach as an answer to the problem of information management since it includes finding out

interesting, non-trivial, implicit and important patterns. Classification, in particular, has been explored in order to address the issue of management.

Existing techniques on classifying documents rely heavily on extracting keywords or highly occurring frequent words in documents. They ignore the importance of extracting a group of related terms that co-occur and more importantly, the inherent structure of a document to classify them. There is no reason to believe that documents within a class/folder adhere to a set of patterns and that these patterns closely correspond to, and are derived from the documents of the particular class or folder. A classification system that determines the patterns of various term associations, with their structure, that emerge from documents of a class and uses these patterns for classifying similar unknown samples is needed. The ability to classify based on similar and not exact occurrences of patterns is singularly important in most classification tasks, as no two samples are exactly alike. This work is an extension to the already existing work, *InfoSift* [2]. *InfoSift* introduced the concept of graph mining to the problem of classification of documents where content of a document can are represented in the form of a graph to preserve the structure and documents can be classified based on the occurrence of similar subgraphs in unknown documents. While *InfoSift* dealt with analyzing whether the incoming unknown samples can be classified to a single folder (binary classification), this thesis extends the approach to classify an unknown sample across multiple folders. To the best of our knowledge, there does not exist any work in the area of text, email or web page classification that infers patterns, along with structure, from text/emails and relies on these learnt patterns for classification.

Since all documents have inherent patterns in them, finding out the interesting or non-trivial patterns would actually help to describe the document which could be used for decision making process. The process of pattern discovery can be automated by data mining techniques and the discovered patterns can be used for classification purposes. The unknown samples can be checked with patterns from different classes in order to find the best match. The motivation behind this thesis is to apply graph mining techniques

for finding interesting patterns from multiple classes and use these patterns for classifying unknown documents. We believe that text and web documents have a structure, as well, in the form of the title, keywords, section headings, the HTML tag elements in case of web pages and the document body. Emails can be represented using structural relationships as it has a structure in the form of the information contained in the headers, the subject and the body. The next section explains the document in each domain in more detail.

1.4 Text, Email, and Web Document Classification

Classification of text is a problem of assigning already known class labels to incoming and unclassified documents/text/emails[1]. The class labels are assigned to the incoming unknown documents based on the sample of pre-classified documents used as the training corpus. In the past, text classification has been used in the context of information retrieval (as is elaborated by the literature on this topic) In this thesis, we deal with general text, email, and web pages for classification using graph mining techniques. The domain and the structure of emails and web pages provide certain knowledge that can be incorporated into the classification task.

1.4.1 Text

Text classification primarily deals with documents where the major part comprises of texts. Some examples are news feeds, research documents, etc. While text classification in the beginning was based mainly on heuristic methods, such as applying a set of rules based on expert knowledge. Lately, the focus has turned to automatic learning and even classification methods. The need for categorization of news stories has prompted a range of solutions that draw upon different techniques from various fields. It includes machine learning techniques such as Support vector machines, Decision tree classifierS, k-Nearest neighbor algorithms and neural networks. Statistical techniques such as Linear least SquareFit, Probabilisitic Bayesian classification and Rule Induction are few other

[1]These terms have been used interchangeably throughout

approaches that have been widely used. Term Frequency and Inverse Document Frequency (TF/IDF) classifiers have also been applied to the problem of text classification.

In all the existing text classification techniques, relevant features are extracted from the training corpus. These features are either used to build vectors which consists of the extracted terms quantified by their occurrence frequencies or to train the classifier to learn those features corresponding to the class to enable classification. These techniques are discussed in detail in Chapter 4.

1.4.2 Email

Electronic mail is a fast, efficient, inexpensive and one of the most preferred way of communication and method of reaching out to a large group of people. It has evolved as one of the most convenient means of communication between individuals as well as groups. Emails can be viewed as a special type of document with some unique identifying information such as From header, To header, CC header, Attachments, etc. Email solves problems like physical traveling and synchronization but as simple as it is, it has got its fair share of disadvantages too. Some problems with emails are loss of context, spam, and inconsistency in information. Another problem with emails is the management of emails. Many users are overwhelmed with a large amount of emails received or sent and SPEND a large amount of time in sifting and classifying them to corresponding folders. A misclassification of an email is as good as losing the email considering the sheer volume of emails received each day and the number of folders to be maintained.

The problem of email management can benefit from a tool for easy storing and retrieval of emails automatically. One aspect of email management is to classify the emails into appropriate folders. An automated technique seems to be important considering the amount of time spent in processing the emails by individuals. This time can be greatly reduced if either traditional classification techniques can be adapted or new techniques developed to address the problem of email classification. In general, any email management system would require a classification component for effective management

of emails. The problem of managing different folders and sub folders only magnifies the issue. Hence, email classification is one of the critical tools needed for the effective management of information in the Internet age.

1.4.2.1 Challenges to Email Classification

Email classification can be viewed as a special case of text classification but the characteristics of documents and emails differ significantly. This poses an additional challenge which is not encountered in text or document classification. Email classification is also more trickier than text classification as it is based on a user's personal preferences (different mail filing habits ranging from who rarely classify emails to one who follow a strict hierarchy), varying criteria for filing emails into folders, etc. Emails also differ from documents in richness of content. Additionally, emails may vary drastically from folder to folder. Hence email classification needs more than just application of conventional approaches. Consequently, email classification uses disparate criteria which are difficult to quantify. In addition, as opposed to a static set of corpus typically used for training in text classification, the email environment is constantly changing with a need for adaptive and incremental re-training. Some of the differences and challenges between email and document classification are:

1. **User preferences**: Manual classification of emails is based on personal preferences and hence the criteria used may not be as simple as those used for text classification. For example, different users may classify the same message into vastly different folders based on their preference. This varies significantly from document classification where the class label associated with a document is independent of the user. This distinction needs to be taken into account by any technique proposed/used for email classification.
2. **Variations in Information**: The information content of emails vary significantly, and other factors, such as the sender, the group the email is addressed to, etc. play an important role in classification. This is in contrast to documents which are

richer in content resulting in easier identification of topics or context. In case of emails, the above factors assume importance as the body of the message may not yield enough information.

3. **Folder's characteristics**: The characteristics of folders may vary from dense (more number of emails) to relatively sparse. A classification system needs to perform reasonably well even in the absence of a large training set. A graceful degradation in accuracy may be acceptable with decreasing training data. Emails within a folder may not be cohesive i.e., the contents may be disparate and not have many common words or a theme. We characterize these folders on a spectrum of homogeneous to heterogeneous. A folder may lose its homogeneity as it becomes dense making it difficult to associate appropriate central theme/structures with the folder.
4. **Sub-folder classification**: Emails are typically classified into sub-folders within a folder. The differences in the emails classified to sub-folders may be purely semantic (e.g., individual course offerings within the courses folder, travel within the projects folder etc.,) or theme oriented. The ability to classify emails to appropriate sub-folders will require a clear separation of representative folder characteristics or traits. Email folders may also be split when the number of emails in the folder becomes unmanageable or contents of many folders may be merged at times. Any approach used for email classification should be able to deal with these nuances which are typically absent in text classification.

Text classification techniques can be used to solve the problem of email classification but they have to take into account the differences listed above. Additionally, they can draw information available in the email domain for classification. A number of text classification techniques have been applied to the problem of classification. Further elaboration on some of these are given in Chapter 2.

1.4.3 Web page

Web pages possess an inherent structure in the form of title, meta tags, anchors to other relevant pages, body, etc. which can be used to classify other unknown web pages. Web-page classification is much more difficult than pure-text classification due to a large variety of noisy information embedded in Web pages. Due to the different dimensionality and different representations of web pages, simply classifying them with techniques based on ordinary text documents would not be suitable. Hence, we use the structure of web page in order to derive patterns with relationships in order to classify them.

1.5 Focus of the Thesis

In this thesis we propose an approach that adapts graph mining techniques for classification of documents (text, emails and web pages) across multiple classes. It is based on the premise that representative (common and recurring) structures/patterns can be extracted from pre-classified classes and can be effectively used for unknown sample documents. Supervised learning along with domain characteristics are exploited to identify the composition of previously labeled documents or emails and these are used for the classification of unknown text samples or incoming emails. This work mainly concentrates on how patterns from multiple classes can be used for classifying unknown samples. We have proposed a scheme for globally ranking the patterns discovered from multiple classes. Ranking function defined over representative subgraphs generated from folders depends on – importance of a structure in a single class, and its uniqueness or commonality across multiple classes. To this effect we have developed a system by the name *m-InfoSift* that deals with multiclass classification of incoming documents.

Documents in a given class correspond to one another and the similarity between them provides the discriminating capability required to distinguish one class from another. Also, users organize email folders based on their content, patterns that occur in the email messages, and personal preferences (for creating folders and sub-folders). Our approach is based on the basis that a class or an email folder consists of representative

documents or emails and the structure and content of the emails can be extracted to work with domain knowledge. We also hypothesize that the notion of *inexact graph match* is critical to our work in order to extract patterns that are **similar** to each other. It helps in grouping similar patterns rather than looking for exact/identical patterns, which may be difficult in textual domains.

Significant work carried out in the area of graph based data mining includes the frequent subgraph discovery algorithm (FSG) [3], the Subdue substructure discovery system [4] and the frequent graph miner:gSpan [5] among others. Our work requires a means of substructure discovery directed by specific constraints (explained later). The notion of matching inexactly within bounds dictated by various domain characteristics is necessary. FSG and gSpan do not have this notion of matching inexactly within a threshold value as they use canonical labeling. We have chosen to use the Subdue substructure discovery system as it supports many concepts such as inexact graph match, which we consider important.

Subdue discovers frequently occurring subgraphs using the minimum description principle. Isomorphism or inexact graph match is used to make sure similar (and not merely exact) substructures are identified. Since Subdue identifies a large number of such patterns, they are sifted into a manageable number of patterns using several criteria such as, the frequency of occurrence, size of the pattern, average size of documents or emails in the class or email folder, the size of the document class or email folder and so on. All these patterns are ranked based on their importance to the class in which they occur and their uniqueness across all the other classes. The incoming sample is classified to the class of the best matching structure.

Given the task of discovering frequent patterns and using them to classify unknown samples, we divide our task into the following phases:

1. Discover interesting structures from all the classes under consideration
2. Rank all the structures across all the classes

3. Classify the incoming unknown samples to the class of highest matching ranked structure

Phases 1 and 3 have been extensively researched by Aery [2]. While the thesis established a framework for adapting graph mining techniques for classification of unknown sample to single folder, this thesis extends its functionality and utility to multi-folder classification by ranking all the structures across all folders. Our approach also includes a formula for calculating ranks for the representative structures which is based on the intuition behind the use of TF-IDF principle.

The rest of the thesis is organized as follows: Chapter 2 presents the related work in the area of text, email and web page classification. Chapter 3 gives a basic overview of graph mining and discusses the Subdue system. Chapter 4 presents an overview of graph mining system. Chapter 5 discusses the working details of the system explaining the different parameters concerned. Experimental results, comparisons and implementation details are presented in Chapter 6 while Chapter 7 outlines the conclusion and future work.

CHAPTER 2

RELATED WORK

Document classification is a problem of assigning an unknown electronic document to one or more categories, based upon its contents. In other words, assigning pre-defined class labels to incoming, unclassified documents. These class labels are defined based on a sample of pre-classified documents used as a training corpus. This problem of document classification has been well researched upon and a lot of techniques have been proposed which include information retrieval, machine learning, and probability-based techniques among others.

This chapter briefly presents a concise overview of some of the widely used approaches for document classification. A lot of these techniques involve email and web page classification as well. The various techniques proposed for classification include Support Vector Machines (SVM) [6], decision trees [7, 8, 6], k-Nearest-Neighbor (k-NN) classifiers [9, 10, 11], Linear Least Square fit technique [12], rule induction [13, 14, 15, 16], neural networks [17, 18] and Bayesian probabilistic classification [19, 20, 21, 8, 6, 22]. Also, the class of Term Frequency - Inverse Document Frequency (TF-IDF) classifiers [23] from information retrieval have been applied to the problem of text classification.

The following sections describes some of the text classification techniques along with techniques that deal in email classification and web page classification. A discussion of some of these systems is presented when we consider the related work in area of email classification. Techniques and approaches that automate the task of filing emails and classifying them is also discussed in this chapter. For classifying web pages, techniques that combine conventional text classification approaches THAT incorporate the domain knowledge have been proposed.We will now discuss some of the text classification approaches that have been outlined earlier.

2.1 Text Classification Techniques

This section provides a brief overview of some of the popularly used text categorization techniques. The various text classification methods have been drawn from fields as diverse as machine learning, probability theory and statistical learning theory amongst others. To provide an idea of the diversity by way of which each of these methods solve a text classification problem at hand, we will now discuss some of the aforementioned techniques.

2.1.1 Term Frequency - Inverse Document Frequency

The TF-IDF weight is often used in information retrieval and text mining techniques. It is a statistical measure that is used to evaluate how relevant or important a word is to a document or in a collection of documents. The importance of the word is directly proportional to its occurrence frequency in the same document but is offset by the frequency of occurrence in the corpus. A lot of work in information retrieval use the TF-IDF weight scheme as part of their work. Variations of TF-IDF techniques have been used in search engines as a tool in scoring and finding relevant documents to the words in a given user query.

Term frequency is the number of times the given term occurs in a document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term frequency regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document. The *term frequency* is given as

$$t_i = \frac{n_i}{\sum_k n_k} \tag{2.1}$$

where n_i is the occurrence frequency of term t_i and the denominator if the number of occurrence of all the terms.

The *inverse document frequency* is a measure of the general importance of that term across all the documents in the corpus. This weight is calculated by dividing the

number of words contained in the document by the total number of documents in the corpus. The *inverse document frequency* is given as

$$idf_i = \log \frac{|D|}{|\{d{:}d \ni t_i\}|} \tag{2.2}$$

where $|D|$ is the total number of documents in the corpus and $|\{d{:}d \ni t_i\}|$ is the total number of documents in which term t_i occurs in.

Then the tf-idf of that term is given by

$$tfidf = tf.idf \tag{2.3}$$

A high weight can be obtained by a high term frequency (in the given document) and a low document frequency of the term in the whole corpus of documents. This tends to filter out the common terms across the documents giving in a high weight to the unique terms in a document. We have adopted a similar technique in our approach of filtering out common representative substructures from the unique substructures that are mined from the input graph. This will be explained further in the upcoming chapters.

2.1.2 Naive Bayesian Classifier

The naive Bayesian classifier is a probability based classifier that assigns class membership or a posterior probability value to a sample based on a combination of the prior probability of occurrence of a class and probabilities of the terms (also called the likelihood), given they belong to that class. For the text classification task, this translates to the combination of the probabilities of terms and the existing categories to predict the category of a given document. Using the Bayes rule we can predict the posterior probability of a category C_j among a set of possible categories $C = C_1, C_2, C_3.....C_n$ and given a set of terms $T = t_1, t_2, t_3.....t_n$ as

$$p\left(C_j|t_1, t_2, t_3.....t_n\right) \propto p\left(t_1, t_2, t_3.....t_n|C_j\right)p\left(C_j\right) \tag{2.4}$$

Naive Bayesian classifiers make a simplifying assumption of term independence (albeit incorrectly), that the conditional probability of a term occurring in a document is independent of the conditional probabilities of other terms that appear in that document i.e.,

$$p\left(T|C_j\right) \propto \sum_{k=1}^{n}(t_k|C_j) \tag{2.5}$$

Using this naive assumption, the posterior probability can be re-written as

$$p\left(C_j|T\right) \ \propto p\left(C_j\right)\sum_{k=1}^{n}(t_k|C_j) \tag{2.6}$$

Although the assumption made is strong and not often accurate, it does simplify the computation of term probabilities (as they can be calculated independent of each other). The performance of the classifier is good and compares well with other sophisticated techniques such as decision trees and neural networks. In our evaluation process, we have compared the performance of our approach with that of Naive Bayes and the results are shown in the forth coming chapter.

2.1.3 k-Nearest Neighbor Classifiers

In k-NN classifiers, as the name suggests, the classification of an unknown test sample is based on its 'k' nearest neighbors. The assumption is that the classification of an instance is based on others that are similar to it. Each document in the training set is represented by a feature vector, which is the set of relevant attributes of the sample. The technique for feature extraction can be as simple as the occurrence frequency of the term in the document. To classify an unknown sample, its corresponding feature vector is constructed and compared with the feature vectors of all samples in the training set. The similarity metric used is generally a distance measure such as the cosine distance function given in equation 2.7.

$$SIM(V_j, X) = \frac{\sum_{k\in(V_j \bigcap X)}\left(w_{k,j} \times f_k\right)}{\sqrt{\sum_{k=1}^{n}\left(w_{k,j}\right)^2 \cdot \sum_{k=1}^{n}\left(f_k\right)^2}} \tag{2.7}$$

where, V_j is the vector corresponding to the j^{th} document in the training set
X is the unknown sample to be classified
$w_{k,j}$ is the weight of the k^{th} term in document j
f_k is the weight of the same term in the test sample and
the denominator is the norm of the two document vectors

Only those terms that occur both in the test and training documents are considered. This similarity measure produces a high value when the two vectors being compared are similar. A value of 1 indicates the two vectors are identical, while a similarity value of 0 indicates that the two are unrelated.

The training examples are ranked according to their distance from the test sample and the k nearest examples are selected. To assign a class label to the test sample, weighting schemes have been devised, but a simple rule that assigns a class label that corresponds to the majority of its neighbors can be used. The performance of k-NN again, is among the best for techniques proposed for text classification [24]. The classifier performs well as it uses the majority decision of k training samples. Due to the same, the effect of noisy data is also reduced. The drawback of the approach is the presence of a large feature space, which can become problematic as the size of the training set increases.

2.1.4 Support Vector Machines

Support Vector Machines (SVM) were introduced by Vapnik [25] in 1979 [26], but have become popular in the last decade or so. Support Vector Machines belong to the set of discriminant classifiers (which include neural networks and decision trees among others). They are based on the Structural Risk Minimization principle [26] and aim at minimizing structural risk instead of empirical risk. Let us consider the simplest case that corresponds to a linearly separable vector space. The problem here is to find a decision surface that best separates the positive and negative examples of a class. A decision surface that does so is called a 'hyperplane' and includes the notion of a margin,

which corresponds to how much the decision surface can be moved without affecting classification. A linear SVM can therefore be stated as a hyperplane that maximizes this margin between a set of positive and negative examples of a class. The margin is the distance from the hyperplane to the nearest of the negative and positive samples. The solid line in figure 2.1 shows the hyperplane that separates the positive and negative training samples of a class and the thin lines on either side define the margin by which the hyperplane can be moved without causing misclassification. The hyperplane in the figure has maximal margin, any other decision surface will have a smaller margin than the one shown.

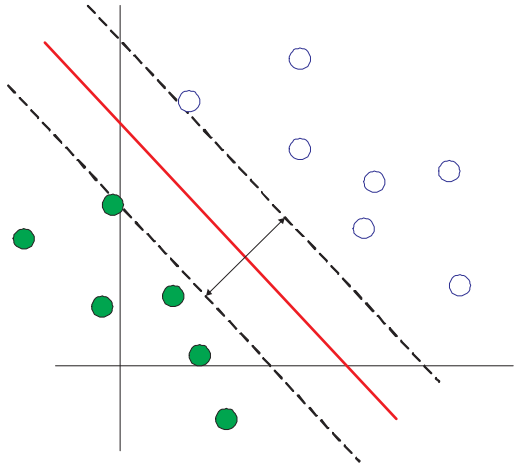

Figure 2.1 The decision line(solid) with maximal margin
The Support Vectors are points on the dashed lines

The optimal separating hyperplane is given by the equation

$$w * x - b = 0 \tag{2.8}$$

The linearly separable cases can be generalized to linearly non-separable cases. The performance of Support Vector Machines for text classification tasks has been studied in detail and they exhibit a remarkably better performance than most other classification techniques [24]. SVM classifiers perform well even in the presence of sparse data, as in

effect the classification mainly depends upon the support vectors of a class. They are capable of handling large data sizes and the classifier performance is consistently good.

From the above discussions on classification techniques, it is clear that the problem has been studied in depth and different methodologies have been applied to solve the task at hand. With this discussion on text classification techniques, we now present the relevant work in the area of email classification.

2.2 Email Classification Techniques

As we have stated before, the problem of email classification presents certain challenges that are not found in text classification. Certain characteristics of the domain (e.g., information contained in the headers and so on) have to be taken into account to ensure good classification. Many text classification techniques have been applied to the problem of email classification. Based on the mechanism used, email classification schemes can be broadly categorized into: i) Rule based classification, ii) Information Retrieval based classification and iii) Machine Learning based classification techniques. In the sections that follow, we present an outline of some systems that have been developed to automate the task of email classification.

2.2.1 Information Retrieval Based Classification

Segal and Kephart [27] use the TF-IDF classifier as the means for classification in *SwiftFile*, which is implemented as an add-on to Lotus Notes. The system predicts three likely destination folders for every incoming email message. The TF-IDF classifier is based on the TF-IDF technique used in information retrieval. For each email folder, a vector of terms that are frequent across the emails in the folder (term frequency) and infrequent across other folders (inverse document frequency) is created. The set of terms thus selected is capable of discriminating the features of a given folder with those of other folders. To classify an incoming email, the term frequency vector of the email is constructed. It is compared with the TF-IDF vectors of all folders using a cosine

similarity metric that is similar to the one stated in equation 2.7. The new email is classified to the folder where the value of the cosine distance function is maximum.

The TF-IDF classifier performs well even in the absence of large training data and the classifier accuracy remains reasonable as the amount of training data increases, adding to the heterogeneity of a folder. The classifier learns incrementally with every new message that is added or deleted from a folder, eliminating the need for re-training from scratch.

2.2.2 Machine Learning Based Classification

Various machine learning based classification systems have been developed. The *iFile* system by Rennie [28] uses the naive Bayes approach for effective training, providing good classification accuracy, and for performing iterative learning. The naive Bayesian probabilistic classifier has also been used to filter junk email effectively as shown by Sahami et.al [29]. The *Re:Agent* email classifier by Boone [30] first uses the TF-IDF measure to extract useful features from the mails and then predicts the actions to be performed using the trained data and a set of keywords. It uses the nearest neighbor classifier and a neural network for prediction purposes and compares the results obtained with the standard IR, TF-IDF algorithm. Mail Agent Interface (*Magi*) by Payne and Edwards [31] uses the symbolic rule induction system *CN2* [32] to induce a user-profile from observations of user interactions. The system suggests actions such as 'delete', 'forward' and so on for each new email message based on the training, hence results for multi-class categorization are difficult to assess.

2.2.3 Rule Based Classification

Rule based classification systems use rules to classify emails into folders. William Cohen [33] uses the *RIPPER* learning algorithm to induce "keyword spotting rules" for email classification. RIPPER is a propositional learner capable of handling large data sets [34]. Cohen argues that keyword spotting is more useful as it induces an

understandable description of the email filter. The *RIPPER* system is compared with a traditional IR method based on the *TF-IDF* weighting scheme and both show similar accuracy. The *i-ems* (Intelligent Mail Sorter) [35] rule based classification system learns rules based only on sender information and keywords. *Ishmail* [36] is another rule-based classifier integrated with the Emacs mail program Rmail.

Although rules are easy for people to understand [37], managing a rule set may not be so. As the number and characteristics of incoming emails change, the rules in the rule set may have to be modified to reflect the same. This puts a cognitive burden on the user to review and update the rule-set from time to time, often involving a complete re-writing of rules.

Most of the email managers (e.g., outlook, eudora), allow users' to set rules for classifying email to folders. These rules have to be specified manually and can use words from various categories. The main problem here is in the manual specification and management of these rules which can become cumbersome and need to be changed often to make them work properly.

2.2.4 Temporal Feature based classification

Kiritchenko et.al.,[38] employs temporal features in order to classify email messages into classes. Temporal features such the day of the week, time of the day, etc. have been incorporated into the traditional classification approaches. Relevant temporal features are extracted from emails and combined along with conventional content-based classification approaches in order to build a much richer information space to improve accuracy.

A set of emails is viewed as an *event sequence* $(c_1, t_1) \rightarrow (c_2, t_2) \rightarrow .... \rightarrow (c_n, t_n)$, where each event corresponds to an email and is represented as a pair (c_i, t_i) with $c_i \in C$ being the category of email (event type) and t_i being the timestamp of the email. The events are based on the tiemstamps. These temporal relations are then transformed into patterns called *temporal sequential patterns* which is an ordered sequence of event types

$c_1 \rightarrow c_2 \rightarrow ... \rightarrow c_k$ along with an ordered sequence of time intervals $d_1 \rightarrow d_2 \rightarrow ... \rightarrow d_k$. A Apiori-like algorithm was developed to mine frequent sequential patterns in the input dataset. These sequential patterns are used to classify the incoming unknown emails.

2.2.5 InfoSift

The work done in *InfoSift* [2] by Manu Aery forms the foundation for this thesis. It showed the feasibility of graph mining approach for document classification and established a framework that included the identification and evaluation of parameters that are relevant for this approach to classification. Although graph mining techniques for classification was first employed in *InfoSift*, it only addressed binary classification to establish the feasibility and framework by performing extensive experiments to tune parameters. The use of domain knowledge for the purpose of classification has also been highlighted in this work. Experimental results are shown in comparison with naive Bayes for classification of text, email, and web pages. This thesis expands the framework developed in *InfoSift* to multiple classes by ranking the patterns obtained from various classes in order to maintain a global rank list based on the interestingness, uniqueness and commonality of that pattern in all classes. This thesis also re-examines some of the parameters for their sensitivity to multi-folder classification. The simple ranking formula used in *InfoSift* has been generalized to address multiclass classification.

From the discussion on email classification techniques, it is clear that a classifier should be able to learn from the email environment of the user. The learnt information should be used to automatically file emails to the corresponding folders or to provide intelligent suggestions to the user. The information contained within the email headers is important as many systems that learn rules based on the same or derive features from the information in the headers consider it useful for classification. The use of domain knowledge for classification when available, adds to the set of features to make a domain informed decision during classification. We will now move onto the problem of web page

classification, which again, can make use of the features that are unique to the domain of the web.

2.3 Web Page Classification Techniques

We will now briefly discuss the relevant work in the area of web page classification. Although it may seem that text classification techniques can be applied to solve the problem of web page classification, it may not be as straightforward since HTML pages have an underlying structure represented by the various tag elements. It is important to take into account this structural information for classification. Attardi et. al., [39] argue that conventional text classification techniques are content driven and to not exploit the hypertext nature of the web that is characterized by linked pages that have structure. They believe that web page classification needs to be context driven and must derive useful information from the structure and link information. The idea that the content in a link and the context around it must provide enough information to classify the document pointed by the link is the main motivation. The authors claim that a blurb of a page provides significant information about the content of page in a concise manner than the page itself (thereby reducing the noise). The information contained within the links that point to a given page and the context around them are used to assign a category to the page.

Schenker, Last et.al., [40] have used graph models for classifying web documents. The graph is constructed from the text of the web page and the words contained in the title and hyperlinks. An extension of the k-NN algorithm is used to handle graph based data. The graph theoretical-distance measure for computing the distance translates to the maximal common subgraph distance proposed in [41]. The graph model for classifying web pages is compared with the k-NN algorithm that uses the conventional feature vector approach. The performance of the graph model is better than the conventional bag-of-words approach and is also more efficient.

2.4 Multiclass Classification

The approaches explained in the previous sections give an insight into different techniques used for the task of classification. The approaches have been used for building binary classifiers, whether the target variable (unknown document, in our case), can be classified to a single class or not. This section presents the approaches that have extended the binary classifiers to handle multiclass problems.

2.4.1 One-Against-Rest(1-r) approach

The one-against-rest [42] approach decomposes the problem into multiple binary classification problems. If $Y = y_1, y_2, ..y_k$ is the set of classes of the input data then using this approach, the problem is divided into K binary classification problems. For each class $y_i \in Y$, a binary problem is created where all instances that belong to y_i are considered as positive instances, while the remaining are considered as negative instances. A binary classifier is then constructed to separate instances of class y_i from the rest of the class. A voting scheme is typically employed to combine the predictions, where the class receiving the highest number of votes is assigned to the unknown document. In this approach, if an instance is classified as negative, then all the other classes except for the positive class receive a vote.

2.4.2 One-Against-One(1-1) approach

In the one-against-one approach [43, 44, 42], $K(K-1)/2$ binary classifiers are constructed where each classifier is used to distinguish between a pair of classes, (y_i, y_j). Instances that do not belong to either of the two classes, y_i or y_j, are ignored when constructing the binary classifier for (y_i, y_j). As in the previous approach, the test document is classified to a class by combining the predictions made by the binary classifiers. A voting system is also employed in this approach. The output of the binary classifiers can be transformed into probability estimates rather than just votes as voting might lead

to ties between classes. The unknown document can e classified to the class with the highest probability.

2.4.3 Error-Correcting Output Coding

The problem with the previous two multiclass classification approaches are that they are sensitive to the binary classification errors, i.e., even if one binary classifier make a mistake in its prediction, then there might be a tie between different classes ending up in wrongly classifying the documents. The Error-Correcting Output Correction(ECOC) [45, 46, 47] method provides a more robust way for handling multiple folder classification. It is based on information-theoretic approach foe sending messages across noisy channels. The idea behind this approach is to add redundancy into transmitted message by means of a codeword, so that the receiver may detect errors in the received message and perhaps recover the original message if the number of errors is small.

For multiclass learning, each class y_i is represented as a unique bit string of length n kown as its codeword. Then n binary classifiers are trained to predict each bit of the codeword string. The predicted class of test instance is given by the codeword whose Hamming distance(distance between a pair of bit strings is given by the number of bits that they differ) is closest to the codeword produced by the binary classifiers. A property of the ECOC method is that if the minimum distance between any pair of codewords is d, then any $\lfloor (d-1)/2) \rfloor$ errors in the output code can be corrected using its nearest codeword. An important issue is how to design the appropriate set of codewords for different classes. The codewords between different classes should be made as different or the distance between them should be made as large as possible.

The approach proposed in this thesis is different from the earlier approaches applied to the problem of document classification. It is also different from the approaches that have been attempted for document classification. Though a graph based model has been used for classifying web pages, a conventional classification technique adapted to work with graphs is used for the actual classification. To the best of our knowledge, we are

not aware of any work on the use of graph mining techniques for text, email or web page classification. With this overview of the related literature in the area of document classification, the discussion on graph mining and graph mining techniques is presented in the next chapter.

CHAPTER 3

OVERVIEW OF GRAPH MINING

Data in many applications have an inherent structure and reducing them to non-structural (or transactional) format will result in loss of information. Graph representation provides a natural format for preserving the inherent structural characteristics. If processing can be done on this representation it will provide better results as the semantics of the applications (in the form of relationships) is preserved during processing. Complex structural relationships can be modeled as graphs if no constraints are assumed (such as no cycles, no multiple edges, only directional edges, and constraints on vertex and edge labels). Graphs model the data in the form of a **vertex** (to characterize the data), and **edges** (that typify extra information). Unlike transaction mining, Graph mining is used to mine structural data such as DNA sequences, electrical circuits, chemical compounds, social networks, schemes (such as money laundering and fraud) that have associations and relationships of transactions, etc. A graph representation comes across as a natural choice for representing complex relationships as the data visualization process is relatively simple as compared to a transactional representation. Data representation in the form of a graph preserves the structural information of the data which may otherwise be lost if it is translated into other representation schemes.

An email message is inherently made up of a structure that can be used be used for its representation and can be exploited for its classification. The structural relationships between the *headers*, *subject* and *body* of an email can be represented as a graph. This approach of considering the structure of a document is unique from other forms of document classification that assume the document as a set (or bag) of words having no particular structure. This is also true of other documents such as web pages and text documents that have a structure in the form of title, section headings, HTML tag

elements, meta tags, anchors to other pages and document body. By making use of the structural information they can be made amenable to graph mining.

3.1 Overview of Subdue

Subdue [4, 48], earliest work on graph mining, uses information-theoretic model for determining the best substructure given a forest of unconstrained graphs. It is a substructure discovery system that was developed by Cook and Holder. The Subdue *discovery algorithm* discovers repetitive patterns and interesting substructures in graph representations of input data. A substructure is a connected subgraph within the graph representation. Within the representation, entities and objects are mapped to vertices and the relationship between these objects is represented as an edge between the corresponding pair of vertices. An instance of a substructure in an input graph is a set of vertices and edges from the input graph that match the graphical representation of the substructure. The input to Subdue is a forest of graphs and the output is a set of substructures that are ranked based on their ability to compress the input graph using the *Minimum Description Length* [4] (MDL) principle. The compression technique is elaborated in detail in the following sections.

The input is in the form of a table consisting of a list of unique vertices in the graph and its corresponding edges between them. The output is list of representative substructures discovered in the input graph where each is qualified by its size and occur-

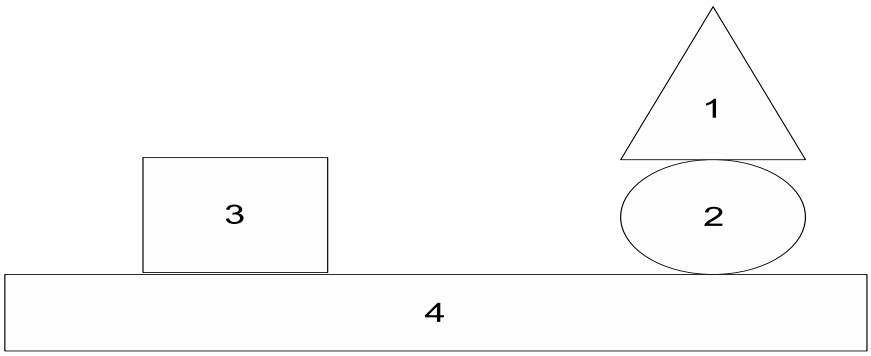

Figure 3.1 High-level view of shapes

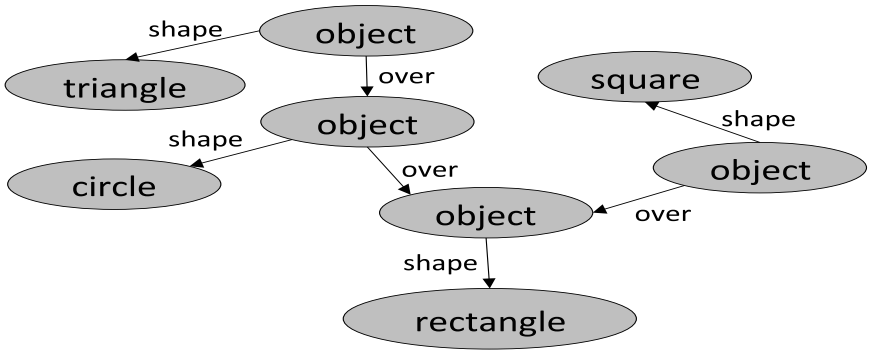

Figure 3.2 Graph representation of shapes example

```
v 1 object
v 2 object
v 3 object
v 4 object
v 5 triangle
v 6 circle
v 7 square
v 8 rectangle

u 1 2 over
u 2 3 over
u 2 4 over
u 3 4 over
u 1 5 shape
u 2 6 shape
u 3 7 shape
u 4 8 shape
```

Figure 3.3 Subdue Input for shapes example

rence frequency in the input graph. Consider the example in figure 3.1. It is a high-level view of shapes resting on a table. The graphical representation of these shapes is shown in figure 3.2.

The input for Subdue (for this particular example) is as shown in figure 3.3. This input is in a form of a file consisting of the list of vertices and the edges between the vertices. Subdue generates the best substructures that compress the input graph the most and lists out the top n substructures. The output given by subdue for the example in Figure 3.3 is displayed in Figure 3.4. The following section briefly explains the Subdue's *substructure discovery process.*

```
Best 3 substructures:

(1) Substructure: value = 0.96959, pos instances = 1, neg instances = 0
 Graph(2v,1e):
  v 1 object
  v 2 object
  u 1 2 over

(2) Substructure: value = 0.953003, pos instances = 1, neg instances = 0
 Graph(2v,1e):
  v 1 object
  v 2 square
  u 1 2 shape

(3) Substructure: value = 0.953003, pos instances = 1, neg instances = 0
 Graph(2v,1e):
  v 1 object
  v 2 rectangle
  u 1 2 shape
```

Figure 3.4 Subdue Output for shapes example

3.1.1 Substructure Discovery in Subdue

The substructure discovery in Subdue is done by using a *beam* search and progresses in an iterative manner stating with substructures of size 1 and expanding to successively larger substructures. A list consisting of a set of substructures to be expanded is maintained. The input graph is compressed by replacing the instances of these substructures by a single node. The resulting input graph is then used for the next iteration to find other interesting substructures. This process continues until the number of iterations specified by the user is reached or it meets one of the several halting conditions, such as the total number of substructures needed , provided by the user.

The occurrences of substructures that have an exact match are unlikely to occur in most domains. Substructure instances that are not exactly same but are similar can also be discovered by Subdue. Subdue is capable of discovering both exact and inexact (isomorphic) substructures in the input graph. Subdue employs a *branch and bound*

algorithm that runs in polynomial time for inexact graph match and discovers graphs that differ by a *threshold* given by the user. This discovery process is to find repetitive and hence, interesting substructures or patterns, and to compress the graph by replacing the instances of these patterns by a single node in order to provide a hierarchical view of the original input graph. This heuristic is explained in the next section.

3.1.2 Compression and Evaluation of Substructures

There are two schemes that Subdue uses for evaluating the candidate substructures in order to determine the best substructures. They are:

1. *Compression based on MDL principle:* The MDL principle states that the best theory to describe a set of data is one that minimizes the description length of the entire data set. The description length corresponds to the number of bIts required to encode the input. This theory was described by Rinssanen [49] and has been used in various applications such as decision tree induction, image processing and others. Subdue employs this principle for substructure discovery where the best substructure is the one that minimizes the description length of the original input graph. According to the principle, the description length of the input graph is given as

$$DL(S) + DL(G|S) \tag{3.1}$$

where,

S is the discovered substructure

G is the input graph

$DL(S)$ is the number of bits required to encode the substructure

$DL(G|S)$ is the number of bits required to encode the input graph G after it has been compressed by the substructure S

The final value of the MDL is defined as

$$MDL = \frac{DL(G)}{DL(S) + DL(G|S} \tag{3.2}$$

A higher MDL value signifies a substructure, that reduces the description length of the original data or in other words, compresses it better. The compression is defined as

$$Compression = \frac{1}{MDL} \tag{3.3}$$

2. *Compression based on size of the graph:* The second compression scheme, based only on size, uses a simple and more efficient but less accurate measure as compared to the MDL metric. The value of a substructure S in graph G is

$$\frac{Size(G)}{(Size(S) + Size(G|S))} \tag{3.4}$$

Here,

Size (G) = Number of vertices (G) + Number of edges (G)

Size (S) = Number of vertices (S) + Number of edges (S)

*Size(G|S) = (Number of vertices (G) - i * Number of vertices (S) + i) +*

*(Number of edges(G) - i * Number of edges(S))*

where,

G is the input graph,

S is the discovered substructure,

$G|S$ is the input graph after it has been compressed by the substructure and

i is the number of substructure instances.

3.1.3 Inexact Graph Discovery

Inexact graph discovery in Subdue aids in grouping similar substructures as a single substructure for both identification and representation. The algorithm developed by Bunke and Allerman [50] is used for inexact graph discovery where a cost is assigned for each dissimilarity. The distortion between two substructures might be a variation in the

edge or in the vertex descriptions like an addition, deletion or substitution of vertices or edges. Two substructures are considered to be isomorphic as long as the cost difference in generating both the substructures to be identical falls within the range the user considers acceptable. Even finding similar substructures is an NP complete problem and requires an exponential algorithm. Subdue uses a branch and bound algorithm that is executed in polynomial time by considering reduced number of mappings. The threshold parameter is used in order to control the number of differences between two substructures. Grouping similar substructures as the same substructure forms an integral part in classification of documents which we will elaborate in further sections.

The concept of inexact graph match is one of the most important aspects of our approach. It allows for substructures that vary slightly in their vertex or edge label descriptions to be chosen as instances of a single substructure. The amount of variation permissible is determined by the *threshold* parameter provided by the user. It specifies the bound on the difference that is allowed between instances of a substructure. Subdue assigns all transformations (insertion, deletion of an edge or vertex and so on) between instances an uniform cost of 1. For a given substructure instance *inst*, to be classified as an instance of another substructure *sub*, the following condition needs to be satisfied: $matchcost(sub, inst) \leq size(inst) * threshold$. In other words, the total transformation cost needs to be less than the number determined by the particular value of threshold and substructure size.

If the size of substructures is large, then even with a small value of *threshold*, there can be a large variation in the edge and vertex labels of the two instances being considered. The default value for *threshold* is 0.0, which means that the graphs have to match exactly. A very large value of *threshold* may not be meaningful as it will match two dissimilar graphs. The exact value of *threshold* has to be determined from the size of the input graph; knowledge of folder characteristics is essential for determining the same.

3.2 Parameters for Subdue substructure discovery

There are a number of parameters that Subdue provides the user in order to control the flow of the substructure discovery process. The input to Subdue is the file containing the list of vertices and corresponding edges as shown in Figure 3.3. Some of the parameters are briefly described below:

1. **BEAM**: This parameter specifies the number of top substructures that are retained for expansion in each iteration of the discovery algorithm. The default value of the *beam* is 4.
2. **ITERATIONS**: Iterations is used to specify the number of iterations to be made over the input graph. The best substructure from the previous iterations is taken to compress the graph for the next iteration. The default is no compression.
3. **LIMIT**: Limit specifies the number of different substructures to be considered in each iteration. The default value is *(number of vertices + number of edges)/2*.
4. **NSUBS**: This parameter is used to specify the number of substructures to be returned as the result from the total number of substructures that Subdue discovers.
5. **OUTPUT**: This parameter controls the screen output of Subdue. The various values are
   1 Print the best substructure found in each iteration.
   2 Prints the best 'n' substructures, where n is the number specified in the nsubs parameter.
   3 Print the best 'n' substructures, as well as the substructure instances.
   4 Print the best 'n' substructures along with their instances and intermediate substructures as they are discovered.
   5 Same as above, prints also each substructure considered.
6. **OVERLAP**: Specifying this parameter to Subdue allows the algorithm to consider overlap in the instances of the substructures. Instances of substructures are said to overlap if they have a common substructure in them. During graph compression an OVERLAP_ <iteration> edge is added between each pair of overlapping instances,

and external edges to shared vertices are duplicated to all instances sharing the vertex.

7. **PRUNE**: If this parameter is specified, then the child substructures whose value lesser than their parent substructures are ignored. Since the evaluation heuristics are not monotonic, pruning may cause SUBDUE to miss some good substructures, however, it will improve the running time. The default is no pruning.
8. **SIZE**: This parameter is used to limit the size of the substructures that are considered. Size refers to the number of vertices in the substructure. A minimum and maximum value is specified that determines the range of the size parameter.
9. **THRESHOLD**: This is the parameter that provides a similarity measure for the inexact graph match. Threshold specifies how different one instance of a substructure can be from the other instance. The instances match if $matchcost(sub, inst) <= size(inst) * threshold$. The default value is 0.0, which means that the graphs should match exactly. Currently, Subdue supports threshold values up to 0.3.

With this overview of Graph Mining and an introduction to the Subdue discovery system, we elucidate the process of incorporating documents for folder classification using graph mining in the *InfoSift* System.

CHAPTER 4

OVERVIEW OF GRAPH BASED CLASSIFICATION SYSTEM

The main approach for solving document classification problems is to develop a scheme that can scan through the contents of the documents and assign a class label to it indicating the folder that best matches the interest of the document. In this chapter, we present a brief overview of the *m-InfoSift* system and describe the parameters adopted for document classification. The system has been developed to adopt graph mining techniques for document classification across multiple folders. *InfoSift*, its predecessor, had the ability to distinguish and categorize incoming documents into a single class. Our work in this thesis extends the current approach to multi-folder classification by generalizing the ranking formula proposed earlier to a global ranking formula that can be used for multi-folder classification. Changes to the existing system and new additions are discussed in detail in Chapter 5. In this chapter, we delve into the specifications of the parameters involved in graph mining (in both, *m-InfoSift* and *InfoSift*) and the reasons for choosing the same.

We adopt a supervised classification approach wherein the training set comprises of pre-classified documents (text, emails, web-pages) with a class label assigned to each of them. The substructures generated by applying graph mining to these training samples are ranked based on their representativeness and uniqueness and inserted into a global sorted list. Once the substructures have been ranked, the incoming test documents are processed based on their contents and representativeness. The overall flow of control is showed in Figure 4.1.

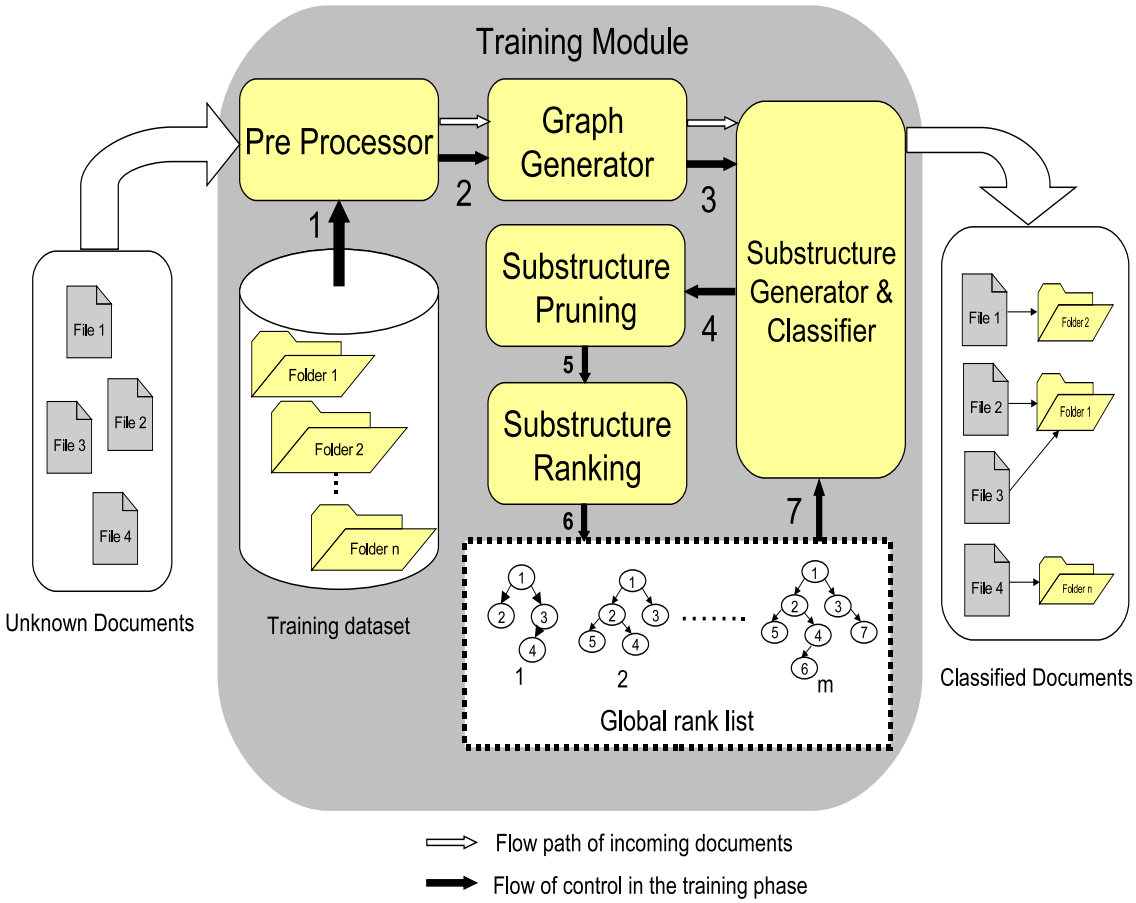

Figure 4.1 System Overview

4.1 System Overview

The document classification process in mainly divided into two phases: Training phase and Classification phase. The classifier is first trained on a set of data where substructures are generated, pruned and ranked. Then the classifier uses these substructures to classify the incoming unknown documents. This section gives a brief description of each step followed concerning the training and classification phase as shown in Figure 4.1.

1. **Pre-Processing**: The documents in the training set contain stop words and different forms of the same word which need to be eliminated in order to generate interesting substructures. Stop word (such as – is, to, from, etc.) elimination, Stemming and Feature selection are done before various characteristics of the class

can be calculated in order to derive the parameters for substructure discovery. Some examples of the class characteristics are *average size of documents in a class*, *number of unique words*, etc. Details of stop word elimination and Stemming are discussed in Section 4.2.1.

2. **Graph Representation**: Once the test documents have been pruned and the respective folder characteristics have been computed, the documents are represented in the form of graphs using the **Graph Generator** module. Text, emails and web pages have a definite inherent structure attached to them which is used to generate graphs. We have proposed different canonical graph representation schemes in order to generate substructures with better structural representation. The details about the different schemes are elaborated in Section 4.2.2.
3. **Substructure Extraction**: Subdue (described in Chapter 3) is used to extract interesting representative substructures from the training data set. The parameters for Subdue depend upon the folder characteristics which are computed during the process of representing the input in the form of graphs. One of the salient parameters is the *threshold* which allows grouping of similar substructures to be considered as the same substructure.
4. **Substructure Pruning**: The output of the discovery process generates a large number of substructures; however, only a percentage of them contribute towards the classification process since the variation observed amongst majority of these substructures is only minimal. Furthermore, the cost of retaining and processing all the generated substructures is high and certainly not desirable in an already expensive processing technique (graph mining). Due to these reasons, the substructures have to be significantly pruned before they can be ranked and employed for classification. The aim of pruning is to identify only those substructures that would help in discriminating the unknown documents during classification. Hence, the output substructures are pruned based upon a range of conditions (elaborated

in Chapter 5) and only those which cover a significant portion of the class are retained.

5. **Substructures Merging and Ranking**: Once all the substructures from different classes have been generated and pruned, they are merged into a single list in order to be ranked. Some representative substructures occur more frequently or measure well in terms of size; hence, these can be considered to be more important than the others. It is therefore important to discriminate the substructures from the view point of classification. Certainly, there is a difference in a match with a highly ranked substructure versus a lower one. The ranking of substructures globally across all the training classes mainly dismisses the problem of selecting the order of folders for processing. More details about ranking are discussed in further sections.
6. **Processing incoming unknown Documents**: Pre-processing (similar to the one applied to the test samples such as stop word removal, stemming, etc.) is applied to the unknown sample to be classified to bring it into a canonical representation. The canonical representation of this document is then converted into a graph for classification.
7. **Classification**: The test document, augmented with the graph representing the substructure in the ranked order, is fed to the classifier to check for any occurrences of the substructure in the test document. The test document is grouped to the same class as that of the highest ranking substructure that occurs in it.

4.2 System Description in Detail

In the following discussions, an elaborate description of the pre-processing, graph generation and various parameters in substructure generation step is provided.

4.2.1 Pre-processing

A document usually contains a number of unnecessary words that can adversely affect the characterization process and do not help in characterization of the document.

Moreover, using the entire document, with all the words, would be overwhelming. For example, words constituting articles, conjunctions and more common words that occur frequently across all the documents does not aid in classification of the document and can be pruned without affecting the outcome. Even inflected and derived words, such as 'eat','eating','ate',etc., could be reduced to their stems in order to preserve the semantics and yet reduce the number of unique words in the document. It is important that the original document is pre-processed appropriately as it will otherwise add noise in the form of irrelevant words and reduce the effectiveness of any mining approach.

Several techniques have been used for pre-processing the documents in order to prune the size of input to retain only interesting words. The main goal for pre-processing in InfoSift is to retain the frequent substructures across the document. In order to achieve this, all the words that comprise the substructures have to be retained in the document as well. The terms have to occur frequently across all documents instead of a single one. This notion of retaining the frequent words across the documents takes care of the disparity of some documents being longer than others. Therefore, prior to representing the documents as graphs, the documents are pre-processed by these consequent techniques.

4.2.1.1 Stop Word Elimination

Stop words, such as conjunctions, articles and even common words that occur frequently across all documents, are eliminated. Some of the more frequently used stop words for English include "a", "of", "the", "I", "it", "you", and "and". These are generally regarded as 'functional words' which do not carry meaning (are not as important for communication). The assumption is that the meaning can be conveyed more clearly, or interpreted more easily, by ignoring these functional words. Stop word elimination is performed by many search engines in order to assist users with queries to provide better results by avoiding searching for functional words.

Consider the document shown in Figure 4.2. The conjunctions or articles in the document do not assist in generating interesting substructures or in classification. Con-

CCC AUTHORIZES ADDITIONAL AID
The Commodity Credit Corporation CCC, has authorized an additional 8.0 mln dlrs in credit guarantees for sales of vegetable protein meals to Hungary for fiscal year 1987, the U.S. Agriculture Department said. The additional guarantees increase the vegetable protein meal credit line to 16.0 mln dlrs and increases the cumulative fiscal year 1987 program for agricultural products to 23.0 mln dllrs

Figure 4.2 Document Sample

sequently, the words considered for representing a document are those which occur frequently, preferably across all the documents in a given class and not merely in a single document. Assuming the set of words in that document sample is as shown in Figure 4.3, the frequent set considered for further processing after stop word elimination is displayed in Figure 4.4.

Figure 4.3 Words in Document Sample

4.2.1.2 Stemming

Stemming is the process of reducing the inflected words to their roots/base/stem. This process reduces the number of unique words through out the documents and also aids in classification. For example, the words 'seeing', 'see', 'seen' are all reduced to the same word 'see'. Words ending with 'ed', 'ing', 'ly', which are used to represent

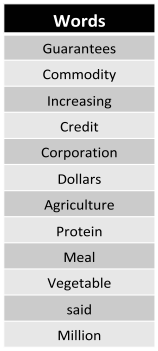

Figure 4.4 Words in Document Sample after Stop Word Elimination

the tenses of the verb or adjectives in English grammar, are stripped to their root. A problem with Stemming might be *homograph disambiguation*, which means a single word can have more than one meaning. For example, the word 'saw', which would be reduced to its root 'see', like in the previous example, can also mean the tool used in carpentry to cut of wood. Since the advantages of stemming surmounts its limitation, it is followed as a part of our preprocessing.

4.2.1.3 Feature Selection

Feature Selection [51] or feature reduction is a technique commonly used in machine learning for selecting a subset of relevant features in order to build the learning model. This process removes the most irrelevant and redundant features from data and also helps improve the performance of learning models by enhancing the generalization capability and ameliorates the learning process. In our system, words, after stop word elimination and Stemming, are ranked based on their occurrence frequencies across the documents in a class and only those words whose frequencies account for more than $f\%$ of the sum of all frequencies are retained. Occurrence of the unique words across different folders are counted while multiple occurrences of the word in the same document are not considered. Words that are a part of this frequent set are considered for generation of graphs. The postulation behind this being that lower frequency words may not contribute towards classification. The parameter f is tuned to observe its effect and identify any possible

Words	Occurrence Frequency
Guarantees	22
Commodity	17
Credit	14
Corporation	12
Dollars	11
Agriculture	7
Protein	7
Meal	5
Vegetable	2
Increase	1
Fiscal	1
Million	1
Year	1

Figure 4.5 Frequent Set of words

Words	Occurrence Frequency
Guarantees	22
Commodity	17
Credit	14
Corporation	12
Dollars	11
Agriculture	7
Protein	7

Figure 4.6 Frequent Set after Feature Selection

dependency on the effect of classification. This ensures the words chosen are frequent not only in a single document, but across a substantial number of documents in class.

Consider the sample document, belonging to a class, in Figure 4.2. To construct the graph corresponding to this document, the set of frequent terms across all the documents is considered. Assuming the set of frequent terms is as shown in Figure 4.5 and the feature selection parameter, f, is 90, the top words that correspond to 90% of the sum of frequencies of all words in the documents is taken. In our example, the summation of frequencies comes to 101 and 90% of 101 is approximately 90. Only the top n words whose sum of frequencies add up to 90 is considered for graph generation. The words in the frequent set after feature selection are illustrated in Figure 4.6.

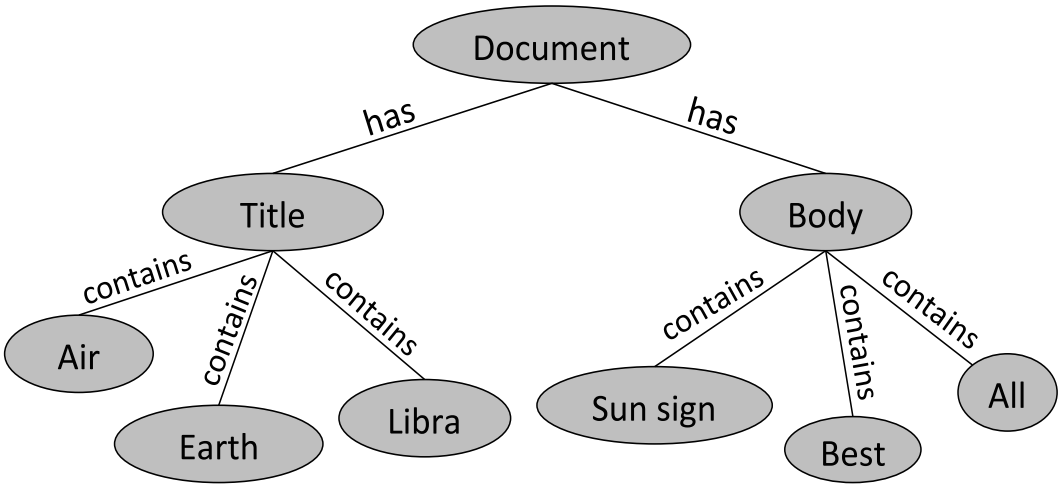

Figure 4.7 Tree Representation of a Text Document

4.2.2 Graph Representation

The graph representations are chosen based on the domain knowledge in order to provide emphasis on the domains. For example, information about the structure of an email message sent over the network or the structural layout of a web page would help in representing the documents as graphs. We have proposed two graph representations that can be used across different domains such as text, emails and web pages. The canonical representation shown in Figure 4.7 is a tree representation. The graph starts with the type of document as the root and then branches out based on the domain. In this example, the document is a text and hence the root is attached to two other vertices, title and body. All the words in the title and body of the document are attached to the title and body vertex respectively with the edge label as *contains*. The representation of an email message under this representation is shown in Figure 4.8. This scheme considers all the information in an email message with each word in the email connected to the central root vertex.

Figure 4.9 illustrates an alternative graph representation, *star representation*, developed to be used across different domains. It consists of a central anchor or root vertex. The chosen words from the document form the the remaining vertices, along with the edges that connect them to the central root vertex with the edge *contains*. The example

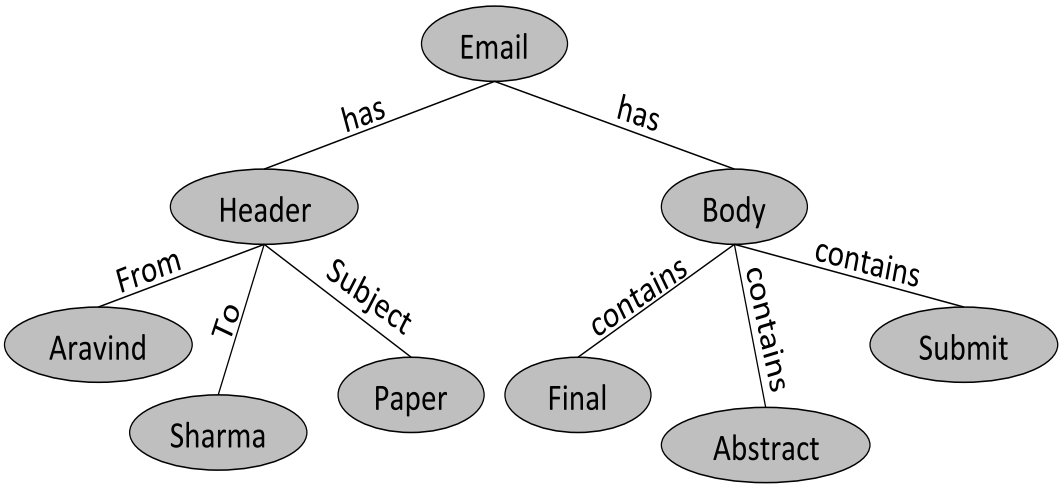

Figure 4.8 Tree Representation of an Email

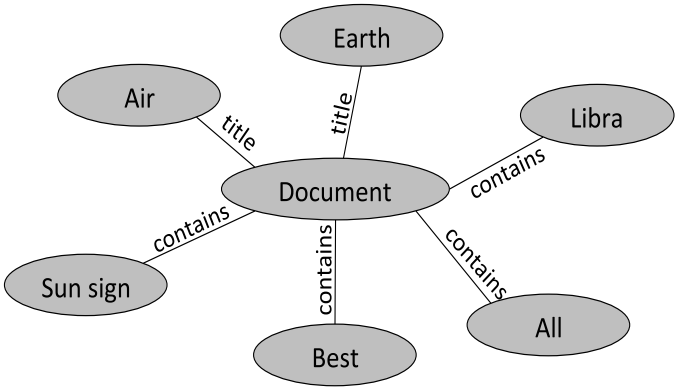

Figure 4.9 Star Representation of a Text Document

shown in Figure 4.9 is for a document. A star representation of an email would contain 'Email' as the central vertex and corresponding labels directing to the other vertices constructed from the message. The ability to label edges makes this simple representation quite effective if the labels corresponds to the various components of a document, email or web page.

Figure 4.10 shows the representation of a web-page in the form of a graph that takes into account the information represented by the title of the page, hyper links that point to other pages and the information represented in the page. Hyper links have also been represented in the graphs since they point to information sources relevant to the current page. The star representation of a Web document is shown in Figure 4.11.

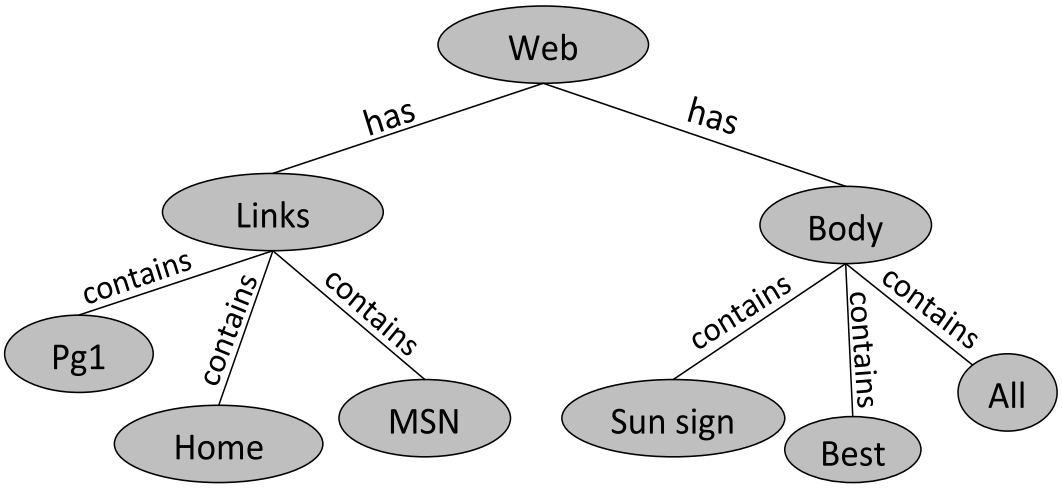

Figure 4.10 Tree Representation of a Web Document

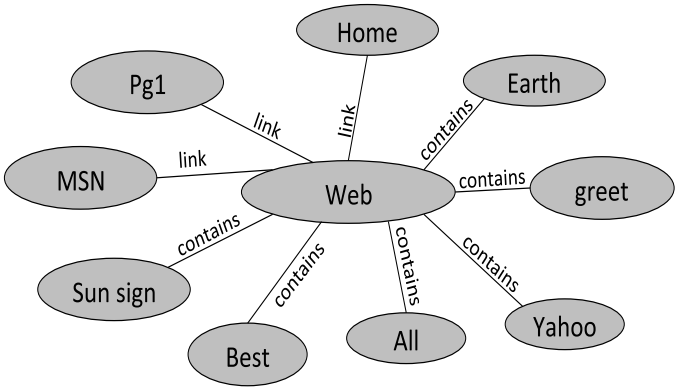

Figure 4.11 Star Representation of a Web Document

Once the documents have been represented as graphs, they can be mined for finding the representative substructures. The input file to the graph miner consists of vertex and edge entries corresponding to the graphs. Each vertex entry associates a unique vertex id with every vertex label. Each entry corresponding to an edge is represented as an undirected edge between a pair of vertices and the corresponding edge label. The input file to the Subdue system corresponding to the representation in Figure 4.9 is shown in Figure 4.12.

The discovery process is derived by certain parameters that are determined as part of pre-processing and graph generation phase. The discussion of parameters warrants

```
V 1 Document
V 2 Air
V 3 Earth
V 4 Sun sign
V 5 Best
V 6 All
V 7 Libra

U 1 2 Title
U 1 3 Title
U 1 4 contains
U 1 5 contains
U 1 6 contains
U 1 7 contains
```

Figure 4.12 Input to Subdue of Sample Document

a detailed study and are explained as the next step towards classification followed by substructure generation.

4.2.3 Computation of Folder Characteristics

The main goal in our approach is to identify representative substructures for a given class of documents and use them for classification. In order to achieve this, we have to choose a number of input parameters for the Subdue algorithm that determine the number and type of substructures identified during substructure discovery. The training set of classes itself needs to be used as a source in order to derive these parameters. Certain characteristics of the class need to be taken into account in order to determine the representative substructures that best characterize a particular class. These parameters must be tunable and effective for diverse document and class characteristics. Not all classes exhibit similar properties; certain classes may be more dense as compared to others and certain others may have larger document content providing extensive amounts of information for training the classifier. For instance, in the email domain, due to constant addition, deletion and movement of emails, the folder contents keep changing rapidly. Hence, class characteristics need to be quantified and specified as input parameters to the Subdue discovery algorithm to ensure that the substructure discovery process is based on traits of the class. If the discovery process is guided by these parameters, the substructures generated are likely to better reflect the contents of the class. Some of

these characteristics, which we believe are substantial to compute the parameters, are considered in the following discussions.

4.2.3.1 Average Document Size, Discovery and Classification Threshold

In the textual domain, it is impractical to find instances that match exactly. For the purpose of classification, flexibility in matching instances of substructures is important. This latitude in matching similar instances should be applicable while building the descriptor for the document as well as while comparing an unknown sample with the class descriptor. This matching of similar instances is carried by the inexact graph match in Subdue. As discussed in Section 3.1.3, the threshold parameter determines the amount of inexactness between two instances. This is by determining the number of vertices and edges that vary among the instances of the same substructure. The actual number is determined by 4.1.

$$(num\,of\,vertices + num\,of\,edges) \times threshold \tag{4.1}$$

A small value of *threshold* allows a significant amount of inexactness while comparing substructure instances of documents that contain a large number of words. It is because even with a small value, the value computed by Equation 4.2 would allow reasonable number of variations. However, for documents with relatively smaller content and hence fewer vertices in the input graph representation, a larger value of threshold is required. Employing the size of the documents in a class, we can determine the amount of inexactness to allow for a graph match. If the amount of inexactness to be allowed in terms of the number of edge/vertex label variations is 'i', then value of threshold is computed as in Equation 4.2

$$threshold = \frac{i}{avg_s} \tag{4.2}$$

where, avg_s is the average size of the documents in the class.

The above formula derives the right value of threshold taking into account the size of the documents in the class. For smaller documents, the value of the threshold will be larger compared to that of larger sized documents. In any case, the maximum number of variations is capped at 4 (larger values only lead to different substructures being considered as similar substructures which in turn lead to increase in misclassification). Here, we have interpreted average document size as a parameter that affects pattern discovery and used it to compute the value of threshold that allows for a reasonable amount of variation and at the same time, preserves the similarity between instances. The value of threshold is used during substructure discovery process and further during classification.

4.2.3.2 Number of Substructures

The number of substructures returned by Subdue is limited by the parameter *nsubs*. To ensure that the representative set consists of substructures that characterize the class, the number of substructures to be returned has to be derived from the class characteristics. If there are a large number of documents in a class, there probably will be a large number of substructure instances as well. But all of these substructures do not aid in classification. We have derived the number of substructures by using both the class size and the average document size along with weights to emphasize each factor. The formula is given in the Equation 4.3.

$$nsubs = w_1 \times C_s + w_2 \times avg_s,\ w_1 > w_2 \tag{4.3}$$

where, C_s is the size of the class and
w_1 is the weighting factor applied to the same
avg_s is the average size of the documents in the class and
w_2 is the weight applied to the average document size

The formula for *nsubs* is built on two class characteristics: 1) Size of the class and 2) Average document size in the class. As evident, the size of the class has got a greater

impact in deriving the value for *nsubs*. Classes have been discriminated into small(less than 60 documents) , medium(61 to 200) and large(Greater than 200 documents) based on the number of documents contained within them. The value of w_1 is based on the class size. The formulas for calculating *nsubs* based upon the class size in shown in Figure 4.13.

Class Size	*nsubs* Formula
Small	$1.25 \times C_s + 0.50 \times avg_s$
Medium	$0.90 \times C_s + 0.50 \times avg_s$
Large	$150 + 0.50 \times avg_s$

Figure 4.13 Formulas for calculating nsubs

Subdue generates and picks substructures based on their ability to compress the original graph. Hence, for a smaller class, large substructures, despite their low frequencies, are picked up as best substructures because abstracting even their few instances, results in greater compression. To make sure that smaller substructures with higher frequencies are also considered, a larger value of *nsubs* is required. Therefore, taking into account the need for a large *nsubs* with a small class size and scaling it to increase in average document size, w_1 has been assigned a value of 1.25 and 0.50 for w_2 for small classes. These values have been determined based on experimental observations for the *InfoSift* framework [2]. The weight w_2 is fixed at 0.50 for all average document sizes.

However, for medium classes, it is likely that repetitive substructures, rather than isolated instances of long substructures, will be reported as the best substructures. Thus, the value of *nsubs* can be taken as a fraction of the class size and scaled with an increase in average document size leading to a value of 0.90 for w_1. Classes with more than 200 documents are considered to be large and an increase in class size thereafter will serve to increase substructures instances rather than the number of substructures themselves.

Consequently, the term corresponding to the class size has been capped at 150 to include the top most frequently occurring substructures.

4.2.3.3 Beam

As explained in Section 3.2, *beam* determines the number of best substructures retained at the end of each iteration of the discovery algorithm. *Beam* ensures that interesting substructures discovered during each set of iterations are available for further consideration. The *beam* value is chosen in proportion to the class size. Large sized classes typically contain many patterns owing to the presence of a large number of documents. A low value of *beam* results in loss of some interesting substructures while a larger value of *beam* only increases the computation and processing time. Hence, the *beam* values have to be chosen based on the class size to ensure no interesting substructures are missed. Experiments employing different beam values on different class sizes were performed. Beam value of 4 returned good results and have been used for experiments. Larger value of beams only leads to increased computation time and resources during substructure discovery and pruning of unwanted substructures while a smaller value of beam did not include many interesting substructures.

4.2.3.4 Minimum Size

The representative substructures that are chosen should provide enough information for differentiate against folders. Substructures that are common across all the folders/emails provide no differentiating capability. For instance, a substructure that consists of information regarding only the headers of emails, like sender and addressee, will not help in classification as emails with same information will be hard to differentiate from each other. This is not an acute problem for single folder classification (where the email is classified to one folder or not). However, for multiple folder classification, this problem aggravates by a great extent. It becomes vital to have further information for enabling successful multi-folder classification. The representative substructures chosen

should provide enough information for discrimination amongst folders. The size should be constrained above a minimum to pick up substructures that contain information more than just a common 'core'.

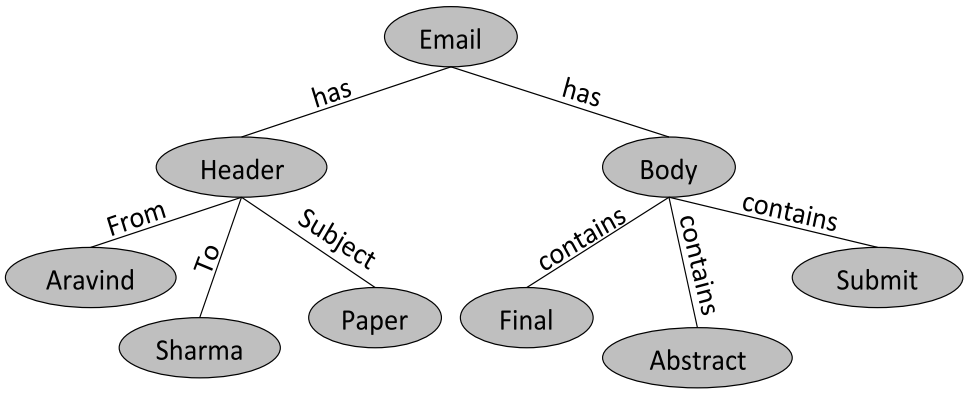

Figure 4.14 Tree Representation of Sample Email

From the graph representation of Figure 4.14, it can be inferred that the smallest sized substructure contain at least four vertices(Email, Header and any two among 'To', 'From','Cc'). Substructures smaller than this are common to all emails within a folder and also across all folders. Therefore, the minimum size of the substructures to be reported is constrained at 4. Using a lower minimum size result in a lot of misclassification of the test documents. This constraint needs to be determined from the graph representation scheme employed.

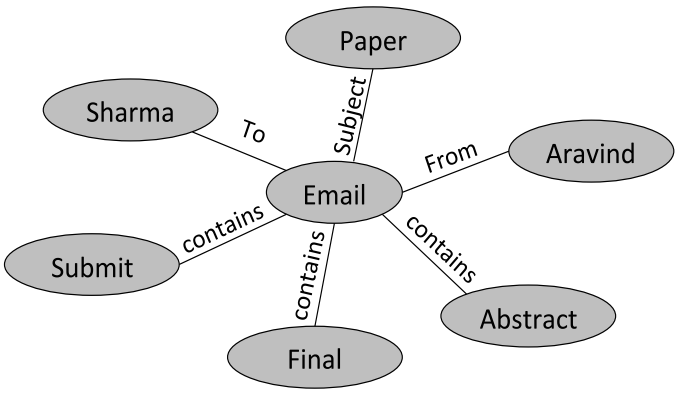

Figure 4.15 Star Representation of Sample Email

For the star representation shown in Figure 4.15, the minimum size of the substructure should be 3. This ensures that the substructures that are picked have sizes greater than the size of substructures that are most likely to be common across many document folders, and hence capable of discriminating between the same.

Having provided a detailed description of the parameters that affect substructure discovery, vis-s-vis, classification of incoming test documents, as well as a means to derive these parameters with appropriate validation, we now elaborate the post discovery processing steps before classification in Chapter 5 and implementation along with our findings in Chapter 6.

CHAPTER 5

FRAMEWORK FOR MULTIPLE FOLDER CLASSIFICATION

This chapter discusses the processing steps after the representative substructures have been discovered. The generation of representative substructures is explained in Chapter 3 using the Subdue's substructure discovery process. The goal of any classification system is to classify the unknown test document into a folder exhibiting the most similar characteristics. In order to achieve this, representative substructures are generated based upon the input folders' (training set) characteristics, and are used against the incoming test documents to determine the best match. Once the substructures have been generated, they are pruned in order to retain only the unique patterns of words and then they are ranked based on how well they represent the class they were generated from. These two process are discussed in this chapter along with the classification of test documents.

5.1 Substructure Pruning

The substructure discovery process generates the top *nsubs* substructures from the training set. Retaining and processing all of these substructures is a problem since a majority of these substructures may be redundant and are not likely to be useful for the purpose of classification. As inexact graph match has been employed, substructures that are variants of each other in terms of just one edge or one vertex are returned as best substructures. Retaining several substructures that have the same frequency and size but vary only sightly in terms of content will not aid in distinguishing the incoming document and will only contribute towards increasing the processing time. Therefore, pruning is necessary in order to retain only those substructures that truly represent the class and

cover a wide area in the input graph during compression in each set of iterations. Pruning of substructures is required in the following two cases:

- *Substructures with same frequency, size and MDL value*: Substructures differing in either frequency, size or MDL value are retained to ensure uniqueness. For example, two substructures, each having ten vertices and different occurrence frequency, are not similar since the same substructure is not reported twice with different occurrence frequency. But two substructures, each with ten vertices and same occurrence frequency with minimal difference, such as different vertex or edge label, are redundant and are pruned. Therefore, each substructure in the representative set after pruning refers to an unique pattern that follows from the documents of the class under construction.
- *Substructures with low frequency in large classes*: On the account of using compression as a heuristic, the discovery algorithm also identifies certain large substructures that do not occur frequently. It is due to the fact that replacing these huge substructures greatly compresses the original input graph. Hence, these substructures are returned by Subdue as part of the best substructure list even though they do not occur frequently. These substructures do not significantly add to the substructure set as they do not cover substantial portion of the class contents. Therefore, substructures with very low frequency as compared to the class size are discarded from consideration. The representative substructures generated from different categories (folders) are generated and pruned to retain unique substructures for ranking.

5.2 Substructure Ranking

The representative set of substructures are generated and pruned separately for each folder. Thus, each of the folders in the training set have a list of pruned representative substructures correspondingly. In order to classify the incoming test document in the best category exhibiting similar characteristics, the test document has to be matched against the representative substructures in each category and the best match is found

by trying to find which of the representative substructures occur in the test document. Ranking of the representative substructures is done in order to position the representative substructures in an ordinal scale in relation to each other so that the test document can be classified to the same class as that of the first matching representative substructure. The *InfoSift* framework currently ranks the substructures in each category in relation to the other substructures in the same category only.

Extending this scheme for multiple categories poses a problem of tyring to classify the test documents based on the local rank of a representative substructure. It additionally depends on the size of the folders in the training data set. A match with a higher ranked substructure in one category holds more weight than a match with a lower ranked substructure in another folder the degree of match for a test document with the substructure is not represented by the rank of the substructure in the folder. For example, a test document can match with a representative substructure RS_1 ranked 10 belonging to Folder f_1 and also match with another representative substructure RS_2 ranked 1 belonging to folder f_2. Though it would be right to label the test document as belonging to folder f_2, the test document could match RS_1 with a better similarity therefore belonging to f_1. This calls for a scheme that could rank the substructures from each other based on their representativeness in a folder. This thesis proposes a formula in this direction for ordering the representative substructures based on their representativeness across all the folders in the training set.

5.2.1 Rank Formulation

The pruned representative substructures list of each folder is collected and appended into a single list to rank them globally. All the representative substructures are compared against each other and ranked based on how unique they are. A rank, called the $GlobalRepresentativenessRank(GRR)$, for each representative substructure is calculated and are ordered based on the same. The GRR of a representative substructure is given in Equation 5.1:

$$GRR(RS) = \left[\frac{FRS(f_{RS}, RS)}{FRS(A, RS)} \times \frac{1}{IFF(RS)^2}\right] \times \frac{S_{RS}}{Max_{RS}} \tag{5.1}$$

where,

RS is the Representative Substructure

f_{RS} is the folder in which RS was extracted from

$FRS(f, RS)$ is Frequency Term of RS across folder f

A is training data set containing all the folders

$FRS(A, RS)$ is the Frequency Term of RS across all the folders in Training set represented as A

$IFF(RS)$ is the Inverse Folder Frequency of RS

S_{RS} is the Size of RS

Max_{RS} is the Size of the largest RS in the global list

The equation in 5.1 computes the rank of the representative substructure RS globally across all the folders in the training set given as the input to the learning model. The rank comprises of two characteristics of the substructure in concern: i) its *occurrence frequency* in both – the folder it was extracted from and the training set and ii) its *size*. The details of these characteristics and the method of computation for the same are discussed in the following subsections.

5.2.1.1 Frequency of Representative Substructure

Representative substructures are a group of words with a structure that co-occur throughout the document class. It is relevant to calculate their occurrence frequency in order to rank them in comparison with each other. The frequency term is shown in the enclosed square brackets in the Formula 5.1. It is based on the principle that the weight assigned to the representative substructure is proportional to its frequency in the folder it was extracted from and inversely proportional to its frequency across other folders. The frequency term comprises of three elements, thus, elaborating the importance of the

representative substructure based on its occurrence frequency. The three elements are discussed below:

5.2.1.2 Frequency Term of Representative Substructure in folder f_{RS}

$FRS(f_{RS}, RS)$(Frequency of Representative Substructure in folder f_{RS}) term captures the importance of the group of words with a structure that relate together in the representative substructure RS. The value of $FRS(f_{RS}, RS)$ is in turn computed by equation given in 5.2:

$$FRS(f_{RS}, RS) = \frac{freq(RS, f_{RS})}{\sum_{i=1}^{n} freq(RS_i, f_{RS})} \tag{5.2}$$

where

RS is the Representative Substructure

f_{RS} is the folder in which RS is extracted from

$freq(RS, f_{RS})$ is the occurrence frequency of RS in f_{RS}

$freq(RS_i, f_{RS})$ is the occurrence frequency of ith RS in f_{RS}

n is the total number of substructures extracted in f_{RS}

In the above formula, the denominator is the sum of the frequency of all representative substructures in that folder. The denominator remains same for each RS in a folder. If the frequency of the RS is high, this term results in a higher value. Otherwise it will result in a lower value. The frequency of each RS is normalized against the total frequency in that folder.

5.2.1.3 Frequency Term of Representative Substructure in all folders A

This term represents the commonality of the representative substructure by computing the occurrence of RS through out the different folders in the training set. The importance of a representative substructure RS is inversely proportional to its occurrence

in other document classes since it does not aid in classification of test documents. This term can be computed by the equation in 5.3:

$$FRS(A, RS) = \frac{\sum_{j=1}^{m} freq(RS, f_j)}{\sum_{j=1}^{m}\sum_{i=1}^{n} freq(RS_i, f_j)} \tag{5.3}$$

where

A is training data set containing all the folders

RS is the Representative Substructure

$freq(RS, f_j)$ is the occurrence frequency of RS in f_j

$freq(RS_i, f_j)$ is the occurrence frequency of RS_i in f_j

m is the total number of folders in the training set

n is the total number of substructures in f_j

The numerator in the above formula is the frequency of RS in all the folders of the training set. The denominator is the total frequency of all RS in the training set. If RS occurs in many folders, including the one it was extracted from, then a higher value is evaluated for the numerator as the frequencies of RS in each of the folder, it exists in, is added up to compute its total frequency across the training set. Its total occurrence frequency is normalized against the total occurrence of all RS in training dataset. The more frequently RS occurs across different folders, the more common is the substructure and hence lower is the value of the rank assigned to it.

5.2.1.4 Inverse Folder Frequency

This term determines the number of folders in which representative substructure RS occurs. The intuition is that, if a representative substructure occurs in many document classes, then it is not a good discriminator and it should be ranked lesser than the ones which occur in fewer document classes. The basis of IFF weighting is the ob-

servation that words that occur frequently across the same document class and rarely across other documents are likely to be of particular importance in identifying relevant material. The IFF term provides a high value for common representative substructures and low value for unique representative substructures. So when the inverse of IFF(RS) is considered, it provides a high value for rarely occurring substructure and low value for a representative substructure that exists across many folders. For example, if RS exists in 3 folders (including the folder it was generated from), then the value for $\frac{1}{IFF(RS)^2}$ would be 0.1111(1/9). Whereas, a RS that exists only in the folder it was generated from would have a value of 1.

5.2.2 Size of the Representative Substructure

The final term in the global rank formula in Equation 5.1 considers the *size of the representative substructure*. Relatively large sized frequent substructures signify greater similarity among the documents in a class. The size of a representative substructure is computed by Equation 5.4.

$$S_{RS} = (number\,of\,vertices\,in\,RS + number\,of\,edges\,in\,RS) \tag{5.4}$$

The size of the representative substructure is compared with the largest substructure in the global list. Based on its relative size, a weight is evaluated to the representative substructure. Therefore, a representative substructure that compares well with the size of the largest substructure, is assigned a higher weight when compared to smaller substructures.

5.2.3 Computation of GRR terms

The GRR of a RS is directly proportional to its occurrence frequency in the folder it was extracted from and inversely proportional to the its commonality. The commonality of RS is defined by its occurrence across all the folders in the training set. The

previous section introduced the GRR and its behavior. This section elaborates on how the frequency terms are computed.

5.2.3.1 Frequency Term of Representative Substructure in folder f_{RS}

The frequency of RS in f_{RS} is given by Subdue in its output of best substructures. For example, in Figure 3.4, the *positive instance* for each representative substructure denotes its frequency of occurrence across the folder. The frequency of all the representative substructures across folder f_{RS} can be computed by summing up the positive instances of all the representative substructures that have been extracted in folder f_{RS}. This term is evaluated to a *high value* when the representative substructure occurs more frequently across the document class it was generated from.

5.2.3.2 Frequency Term of Representative Substructure in all folders A

This term is assigns a weight based on the commonality of RS. The numerator of this term denotes the frequency of representative substructure RS in all the folders in the training set. This is computed by calculating the frequency of RS across all the folders in the training set and then discounting its occurrence frequency in the folder it was generated from. Checking whether RS exists in all other folders is not straight forward because RS might be a subgraph of the representative substructures in other folders or the same words constituting RS might not be in the same order in the representative substructures of other folders. Inspecting whether a representative substructure occurs, as a whole or part of another substructure, in other folders is done using the *graph match* module of Subdue. The intuition behind finding if a representative substructure exists as whole or as a subgraph in other folders is that the common words that comprises a representative substructure, whole or a part of it, needs to be lowly ranked so that they do not lead to wrongly classified test documents. The graph match module is used instead of Subdue in order to increase the efficiency in terms of processing time.

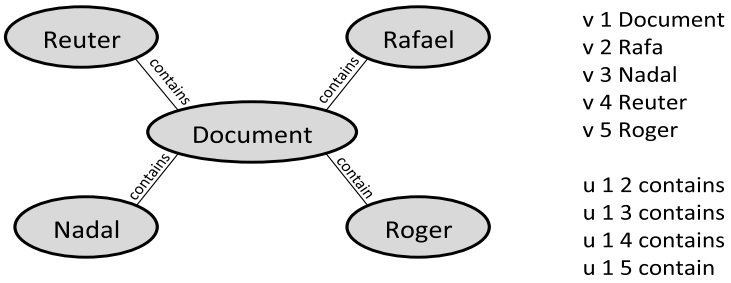

Figure 5.1 Sample Graph g1

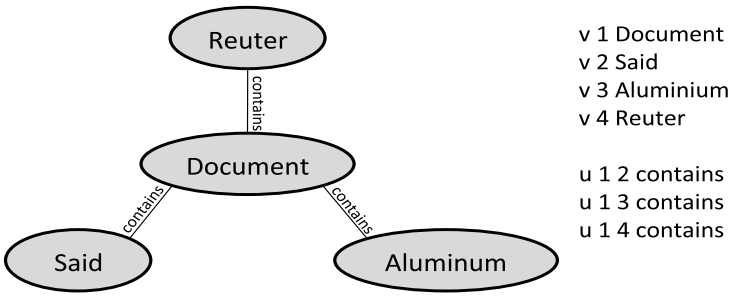

Figure 5.2 Sample Graph g2

The graph match module takes two graphs as input and computes the cost of transforming the largest of the input graphs into the smaller graph. The cost is computed by summing up the number of operations to be done on the larger graph, such as adding or deleting a vertex label or an edge label, to map it to the larger graph. The output of the graph match module comprises of the vertex mapping of largest graph to the smaller graph along with the cost. For example, consider transforming graph g1 to graph g2 as shown in Figures 5.1 and 5.2.

```
Match Cost = 4.000000
Mapping (vertices of larger graph to smaller):
 1 -> 1
 2 -> 3
 3 -> 2
 4 -> 4
 5 -> deleted
```

Figure 5.3 Output for Transforming g1 to g2

As illustrated, graph g1 has got 5 vertices and correspondingly 5 edges while graph g2 has got 4 vertices and edges. Vertices of g1(larger graph) are mapped to vertices of g2 along with the edges. The output of transforming g1 to g2 is shown in Figure 5.3. A cost of 2 is assigned to change the labels of vertices 2 and 3 of g1 (Rafael, Nadal) to vertices 2 and 3 of g2 (Said, Aluminum), cost of 1 to delete the extra vertex 5 (Roger) and a cost of 1 to delete the extra edge 'u 1 5 contain'. Therefore, the graph match module computes a cost of 4 for transforming g1 to g2.

The graph match module can be used to find if two representative substructures are exactly the same or if one representative substructure is a subgraph of the other. The Algorithm 1 shown below has been developed for this purpose.

Algorithm 1 To find if a RS is a subgraph of another RS

1: Obtain the two representative substructures $RS1$ and $RS2$
2: Find the largest substructure of the two and initialize it $RS2$ and the smaller one as $RS1$
3: Calculate $(v2v1) + (e2e1)$ where
   $v2$ and $e2$ are the number of vertices and edges of $RS2$
   $v1$ and $e1$ are the number of vertices and edges of $RS1$
4: Compute the match cost between $RS1$ and $RS2$ using Graph match module. Let $M.C$ be the cost of transforming $RS2$ to $RS1$
5: **if** $(M.C \leq ((v2v1) + (e2 - e1)))$ **then**
6: $RS2$ is $RS1$ or contains $RS1$ as a subgraph
7: **else**
8: $RS2$ is not $RS1$ and does not contain $RS1$ as a subgraph
9: **end if**

The algorithm is initiated by finding the larger of the two subgraphs/substructures given as input. The largest subgraph is tried to map to the smaller one as explained before. The difference between the sizes of the two substructures are found by the formula $(v2 - v1) + (e2 - e1)$. The match cost of transforming one substructure to another is computed using the graph match module. The difference in size is compared with the match cost computed. Two cases arises as explained below

- If the match cost is lesser or equal to the difference in size of the two substructure, then the two substructures are considered to be same or one substructure is part of the other
- If the match cost is higher than the difference in the size of the two substructures, then the two substructures are neither same nor one is a subgraph of the other

Using this algorithm, the frequency of the substructure across different folders is computed by summing up the frequencies computed for RS against each other RS of other folders. The denominator of the term $FRS(A, RS)$, as shown in equation 5.3, is the summation of the frequencies of all the representative substructures in all the folders of the training set. This term determines the common representative substructure Rs occurs across all the folders. The importance of RS is inversely proportional to its commonality in occurrence across folders.

5.2.3.3 Inverse Folder Frequency

The Inverse folder Frequency of RS determines the number of different folders RS occurs in. This measure is computed while calculating $FRS(A, RS)$ by maintaining an index of all the different folders which contains the same representative substructures as RS or part of RS. The calculation of $FRS(A, RS)$ and $IFF(RS)$ is done by making one pass through all the RS of all the folders in the training data set.

In conclusion, the terms of the GRR are computed from Subdue's output and using the graph match module. The graph match module is used to overcome the unnecessary overhead of using Subdue and increase efficiency in terms of processing time. All the terms of the GRR are computed by making one pass through the list of RS in each folders of the training data set. A representative substructure gets a higher GRR due to the following reasons:

1. It occurs frequently across the same folder it was extracted from
2. It occurs less frequently across all the other folders in the training set

3. The size of the representative substructure compares well to the largest substructure in the global list

Once all the representative substructure are ranked, they are ordered based on their GRR. The list of ordered substructures are then used during classification. The classification process is explained in the following section.

5.3 Classification

The ranked substructures are used for classifying the incoming unknown documents. In order to assign an appropriate label to this document, it is compared with the ranked substructures. As with the generation of representative substructures, inexact graph match is used for comparing the unknown document with predefined representative substructures. Each ranked substructure is embedded into the test document to create a forest of two graphs: 1) the test document represented as a graph, and 2) the graph of the representative substructure.

The classifier is used to generate representative substructures from the forest of graphs with a minimum size set to the size of the representative substructure embedded into the test document. The list of substructures generated by the classifier are checked for its occurrence frequency. If the occurrence of the substructure (which is the embedded representative substructure) is greater than 1, then the test document has got an instance of the representative substructure in it. It denotes that the test document contains the words that make up the representative substructure, along with the same relationship. It also means that the test document comprises of the frequent words that represent the document class in the form of the representative substructure. Hence, the test document is filed to the same class with the highest ranked substructure match signifying higher correlation with the class contents.

With this discussion, we move on to the implementation aspects and present our findings for our approached with elaborate experimental results.

CHAPTER 6

EXPERIMENTAL EVALUATION

This chapter presents the experimental analysis and results performed to reinforce our premise that words in document classes exhibit relationships and these patterns can be used to learn and aid in classification of unknown documents. The applicability of this approach across multiple folders for heterogeneous textual domains namely text, emails and web pages have been considered. The performance of the classifier on these domains is consistent and the results have been presented in detail in separate sections. The experimental setup and a brief description of the dataset used is also provided. A brief overview of the system implementation and the details of the configuration parameters is presented below.

6.1 Implementation Details

The framework for multi-folder document classification in InfoSift has been designed in Perl. Perl has been chosen due to its excellent support for text manipulation and processing. As Perl was designed for string processing and extraction, its a natural choice for document pre-processing and feature extraction. The availability of pre-developed modules and functions for many routine tasks and the ability to handle complex data structures in Perl have been utilized in the implementation. As the discovery algorithm is implemented in C, the choice of using an interpreted language for developing the various modules of the document classification system does not slow down the overall performance. The prototype system is an amalgamation of separate yet interrelated set of modules namely, *document pre-processing*, *graph generation*, *substructure extraction*, *substructure pruning*, *representative substructure ranking* and *classification*.

The input to the training model of system is a set of one or more document classes, along with various parameters for graph generation and pattern discovery . The parameters are provided in a configuration file comprising of options for – split for cross validation, choice of graph representation, beam value, etc. The system pre-processes the classes in the training set, generates graphs, computes the various class characteristics and invokes the substructure discovery algorithm. The output generate is pruned and ranked across all the folders in the training set to produce the *global rank list*. This list is then used during classification of the test document. The outcome of the classification along with the output of each module are logged for analysis. In the discussion that ensues, we will briefly describe some of the implementation aspects of the various modules and details about the different configuration parameters.

6.1.1 Configuration Parameters

The modules in the document classification system operates based on the values provided by the user in the form of the configuration file. Options for various parameters such as *choice of graph*, *representation scheme*, *randomized generation* of training and test data sets and so on have been provided. Values that are substantial such as the *substructure discovery threshold* can also be provided in the configuration file. Default values are assigned in case any of the parameters are absent in the file. Besides the parameters specified in the configuration file, parameters such as *nsubs* are computed based upon the document class size and average document size during pre-processing of documents. The various parameters along with the various values are listed below:

1. **Number of Document Classes:** The total number of document classes fed to the training model.
2. **Name of Document Classes:** The names of the document classes or folders that contain the documents to train the classifier.
3. **Graph Representation:** Various graph representations such as star and tree have been proposed for different domains. Each graph representation have been assigned

an unique number id. The choice of the graph representation is input to the system using the option for the scheme.

4. **Training Test Set Split:** The document classes containing the different documents provide information for training the classifier. The percentage of the class sample to be used for training can be specified as a ratio using this parameter in the configuration file. An training/test split of 80:20 and 60:40 have been used i.e., 80% or 60% of the documents in the class are used to train the documents in order to generate representative substructures that represent the documents and the remaining 20% or 40% of the documents are used for classification.
5. **Feature Subset Selection:** The top *f%* of the features representing the document class to be selected during the pre-processing of the folder content.
6. **Random/Sequential Generation:** The option of choosing the *first* n% of the documents in the class or *randomly* chosen n% documents to act as the training set can be determines using this parameter.
7. **Seed:** In case of random selection of documents for the training set and test set for classification, a seed value can be provided for the randomized generation. If left unspecified, the system supplies a default value of 100 for generation process.
8. **Log File:** The file name to log the results of the outputs of each modules during processing and classification. The logged information also contains values of the parameters that were specified in the configuration file. Additionally, information regarding the substructure generation and the representative substructure that matched with the test document are logged for further analysis. In case the log file is not specified, a default name derived from the class names and other attributes is used.
9. **Graph File:** The documents in the training set are represented as a forest of graphs for the substructure discovery process. The file name to store the forest of graphs is specified using this parameter. If a file name is not specified, a default value is assigned to the file based on the class names and its attributes.

10. **Substructure Output File:** The file name to store the output of the substructure generation process. It contains the values of the input parameters to the substructure discovery algorithm in the Subdue system along with the top best substructures that were extracted from the input training data set. This file is used during the pruning process in order to filter out the repetitive substructures that were extracted. A default filename is taken if no name is specified.
11. **Substructure Discovery Threshold:** The amount of inexactness that is permissible during substructure discovery process is specified using this parameter. This value can be specified by the user else it is calculated as explained in Equation 4.2. A value of 0.1 has been used for our experiments to compare the effect of inexact graph match on classification with exact graph match.
12. **Classification Threshold:** This value represents the threshold during the classification of the test document. As the classifier searches for the occurrences of the representative substructure inside the graph representation of the test document, an amount of inexactness is also allowed to group similar instances of patterns just as in substructure discovery. This value can be specified by the user else the same value as substructure discovery threshold is used as default. A value of 0.05 have been used for our experiments.
13. **Minsize:** The minimum size of the substructures generated by the substructure discovery process can be constrained above a certain value by this parameter.
14. **Beam:** The value of the *beam* for the substructure discovery algorithm. Values of the 2,4,8 and 12 have been used for the experiments. If the values are unspecified, then a value of 4 is used for small classes and a value of 8 has been used for medium and large classes by default.
15. **Prune:** This parameter can be turned on or off depending upon whether the output of the discovery process containing the list of best substructures needs to be pruned or used as is for classification. The default is to prune the substructures.

With this overview of the configuration settings, we move onto the details of other implementation issues.

6.1.2 Graph Representation and Generation

The documents that are collected as training set from the document class are used to derive substructures to represent the respective class. For this reason, the documents are converted into graphs that act as inputs to the substructure discovery process. The graph generator developed is capable of generating graphs for various domains such as text, email and web pages. For processing emails, the Perl packages *Mail::Internet* and *Mail::Address* are used to extract the header and body information respectively. *HTML::TokenParser* package is used for processing web pages and deriving the necessary information from the HTML tags in the pages.

Associative arrays or Hashes in Perl have been used to store the term-frequency pairs of the features in the training set. The documents that form the training set are used to construct a *global hash* of term occurrences across all the documents in the class. This set of terms are pruned based on the feature subset selection percentage that is to be retained. During the construction of the graphs for sample documents, only those terms that occur in the global hash after pruning are considered. Class statistics such as document class size and average document size in the class are also computed and logged for substructure discovery during graph generation.

6.1.3 Substructure Discovery

The pattern discovery pattern is handled by the Subdue substructure discovery algorithm. The input to this system such as *threshold*, *minsize*, *graph input file*, *output file name*, etc. are specified in the configuration file. The output of the substructure generation process is written to the file which is processed to prune substructures and generate the representatives of the class under construction.

6.1.4 Representative Substructure Pruning and Ranking

The representative substructures are compared to eliminate those that are similar in terms of substructure size, MDL and frequency and differ only in their description of an edge or vertex label. The list of best substructures from the substructure generation output file is analyzed to discount the similar substructures as per our definition of similarity explained in Section 5.1. In addition, certain large-sized classes substructures that are highly infrequent are pruned as well.

The pruned substructures from each folder in the training set are merged together into a single list and ranked against each other to position them in an ordinal scale based on their representativeness. Associative arrays are used to store the representative substructures. The *Data::Dumper* module is used to save the information in the form of hash data structure. The ranked substructures are sorted using the sort function provided by Perl. For each unique rank(key of the hash), information about the corresponding substructure such as substructure name, folder it belongs to, rank value, size, $FRS(f, RS)$, $FRS(A, RS)$, $IFF(RS)$ are stored. During classification, the classifier tries to match the test document with the substructures in the sorted order. Once a match is found, the classifier stops further comparison with representative substructures and assigns the same label as that of the representative substructure that it matched with.

6.2 Experimental Results

The results of classification experiments on different domains such as text, emails and web pages repositories are discussed here. The experiments have been carried out on Intel Xeon CPU 2.80Ghz dual processor machines with 2GB memory. Exhaustive experiments on a large number of classes with diverse characteristics (different document class size, dense,sparse classes, etc.) have been carried on to study the effect of parameters on classification of unknown test documents in a multiple folder environment. Since each domain presents issues that are unique to it, they have been considered separately for

discussion. The following subsections describes each domain with an introduction to the data set used for experimenting along with presenting and discussing the results.

6.2.1 Classification on Text Repositories

The dataset for text classification is derived from *Reuters-21578* [1] corpus, which has been used as a benchmark for text categorization tasks. The data was originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system. The documents in these classes comprise of news articles from various categories, with multiple category assignments for many documents. The category distribution is skewed with majority of the categories containing varying number documents (from a few to a few thousand) in it. The unlabeled documents in the corpus have not been considered for our experimental analysis. The resulting set of 60 topic categories such as *Cotton*, *Cocoa*, etc. have been used for training and testing purposes.

Numerous experiments have been performed to determine the viability of the proposed approach and to study the effect of various class and document characteristics on classification. The performance metrics used for evaluation is **Accuracy** (given by Equation 6.1) and **Error rate** (given by Equation 6.2). The performance of our approach is compared with the *probabilistic Naive Bayesian classifier*, implemented in the Bow library developed by Andrew McCallum [2]. The experiment results are now discussed in detail:

$$Accuracy = \frac{CD}{\sum_{i=1}^{n} TD_i} \tag{6.1}$$

[1] available at http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] available at www.cs.cmu.edu/ mccallum/bow/

$$ErrorRate = \frac{WD}{\sum_{i=1}^{n} TD_i} \tag{6.2}$$

where,

CD is the number of correctly classified test documents

WD is the number of wrongly classified test documents

TD is the test document

n is the total number of test documents to be classified, i.e., test dataset size

6.2.2 Graph Mining Vs Naive Bayes

Figure 6.1 shows the comparison between the performance of our approach with the Naive Bayesian one. Both the approaches have been tested on different training set size from multiple folders (2 to 16 folders) containing small, medium and large classes.

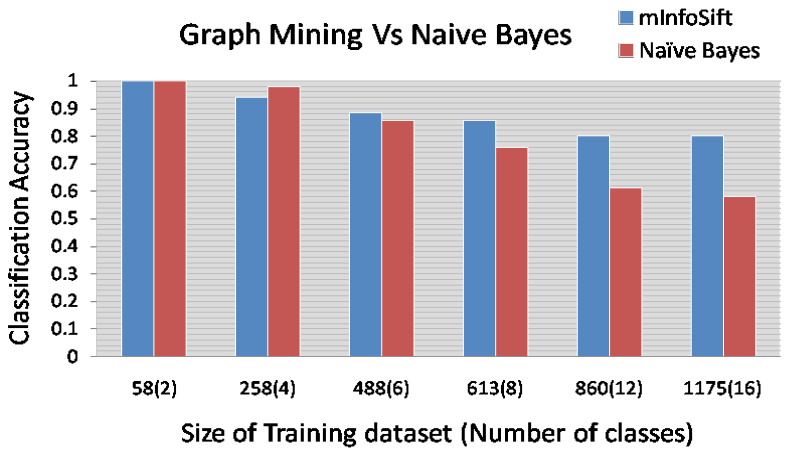

Figure 6.1 *m*-InfoSift Vs Naive Bayes

The classification of the Graph Mining approach is consistently better than the Bayesian approach. Both the classifiers perform well with small number of folders such

as 2, 4 and 6. With the increase in the number of folders in the training set, classification accuracy of both the approaches decreases due to the increasing number of misclassified test documents. Some of the reasons as to why a test document might get wrongly classified is listed below.

- *Lack of adequate data in test document:* Some of the test documents from the class did not contain enough information for it to be classified to any specific class. Test documents with minimum information tend to match with substructures of minimum size, which occur in most of the documents in various classes and hence gets a low rank. This leads to the test document being classified to the wrong folder.
- *Folders with lot of heterogeneous documents:* The input datasets used for experimental analysis contain classes with documents already labeled and classified. Sometimes the classes contains documents dealing with diversified information , i.e., the documents under the same class are very heterogeneous. This leads to large substructures with very low frequency returned as the best substructures by the discovery algorithm. These substructures are ranked lower than the substructures of smaller size with high frequency from other folders. Therefore, small substructures with very high frequency in folders tend to get a higher rank than the large substructures with very low frequency of the heterogeneous folder. So when the test document, which actually belongs to this heterogeneous folder, is tried to classified, it is likely to get classified to the wrong folder due to presence of highly ranked small substructures of other folders.

In the case of Naive Bayes approach, the classifier depends largely on the size of the classes in the training set. When a large class is paired up with small classes in the training dataset, the probability that the most commonly occurring word in the test document to occur in the large class is higher and hence test documents are classified to the wrong folder. The global ranking of representative substructures overcomes this problem by ranking group of words with structure based on how uniquely represent the

class. As evident from the result shown in Figure 6.1, the difference in performance is clearly distinguishable for larger number of classes in the training set. The term probability based Naive Bayes approach clearly assigns the wrong label to a lot more test documents than our approach for large folders. Though Naive Bayes has been proven successful for classification of binary values(whether a test document can be classified to a particular class or not), its independence assumptions are mostly inaccurate when used for multiclass classification. The error rate computation corroborating this discussion is illustrated in Figure 6.2.

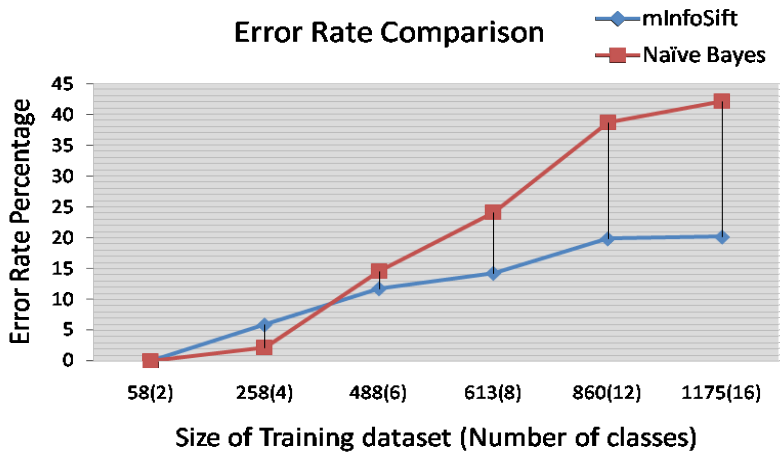

Figure 6.2 Error Rate comparison between m-InfoSift and Naive Bayes

It is clear from the results shown that our approach of ranking the substructures across multiple folders in the training dataset out performs conventional techniques like Naive Bayes in terms of classification accuracy.

6.2.3 Feature Subset Selection

Experiments were conducted to study the effect of the size of the vocabulary or feature set selection on classification. In our experiments, we have used four values for feature set selection by extracting the top 60%, 80%, 90% and 100%. Using a feature selection value of 100% would actually retain all the not so frequent words. It is to

study if these in frequent words does have an effect on classification of the incoming test document though retaining all of them would increase processing and substructure discovery time. graphs representing the training set documents are constructed only from the words occurring in the the feature set. Details about feature subset selection have been elaborated in Section 4.2.1.3. The results of the comparison is shown in Figure 6.3. It is expected that the presence of large number of features,which are words, will result

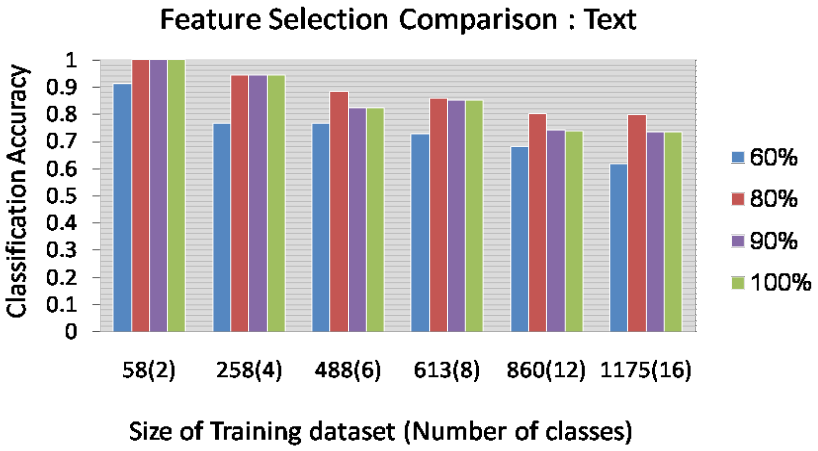

Figure 6.3 Effect of Feature Subset Selection

in better classification, along with a decrease in accuracy with a reduction in feature set size. However, the results show that retaining all the words in the feature set, by using a feature selection size of 100%, do not give the best classifcation accuracy. A feature selection of 80% and 90% performed better over 60%. One reason is that a feature subset selection of 60% contained words that were common across lot of documents which in turn affect the substructure discovery by extracting substructures that were common across different document classes. Feature selection of 80% and 90% contained much more features than 60% which resulted in generating substructures that better represented the document class from each other. It did not include all words, like in 100%, which led to many unwanted substructures being generated. A value of 90% also contained less frequent words in its frequent global set which led substructures to contain lot more less

frequently features. This in turn led to a performance that is similar to 100% feature selection. This makes a strong case for using a value of 80% as feature selection for the rest of our experiments.

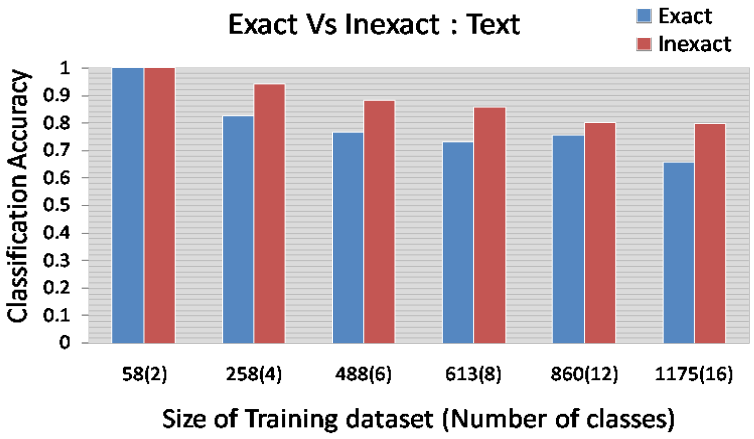

Figure 6.4 Exact Vs Inexact Substructure Discovery

6.2.4 Inexact Vs Exact Graph Match

The ability to match similar substructure instances while making allowances for small variations is important for classification tasks exploiting graph mining techniques where exact matches are hard to find. To this end we have performed experiments to study the classification accuracy on textual domains using exact and inexact graph match. The results are shown in Figure 6.4.

As evident from Figure 6.4, inexact graph performs better than exact graph match. The performance of exact and inexact graph match is similar for smaller number of folders but with the increase in the number of folders, the difference in performance becomes more clearer. The training set of classes comprises of both small classes and large classes. A reason as to the better performance of inexact graph match is because it is able to group similar instances that vary slightly even in the absence of large training data in the case of small folders.

6.2.5 Comparison between Tree and Star Representation

We have proposed two graph representations that have been used to represent the training and test documents to the graph mining system, Subdue. Figure 6.5 show the results of the comparison of the two graph representations on the classification accuracy.

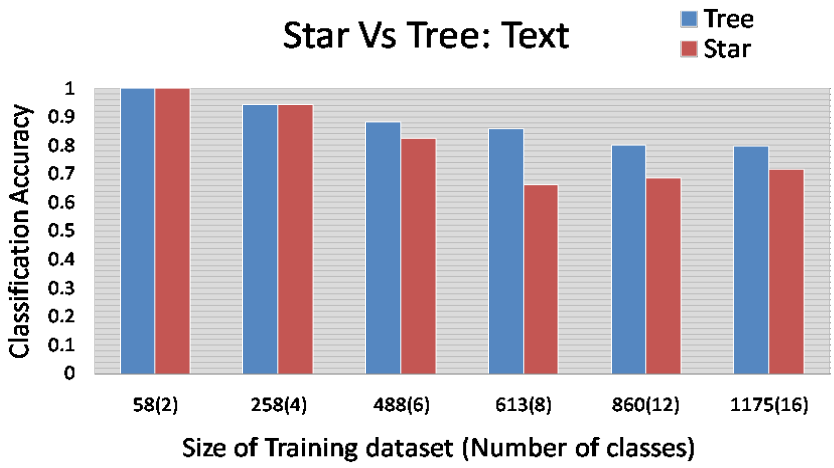

Figure 6.5 Star Vs Tree representation

As seen in Figure 6.5, the tree representation has a better accuracy over the star representation. The tree representation exhibit better structural relationships between the words than the star representation. The better performance of the tree structure is contributed due to the presence of extra layer in its structure(the first layer contains only the root node followed by another layer containing vertices of different components that make up the document like title, body, etc. and a third layer of vertices with all the features attached to the corresponding vertices in the second layer). The star has got only two layers: the root forms the first and the rest of the vertices form the second.

6.2.6 Prune Vs No Pruning

The substructure discovery process generates lot many substructures from the training dataset given as the input. But all of them are not likely to contribute towards

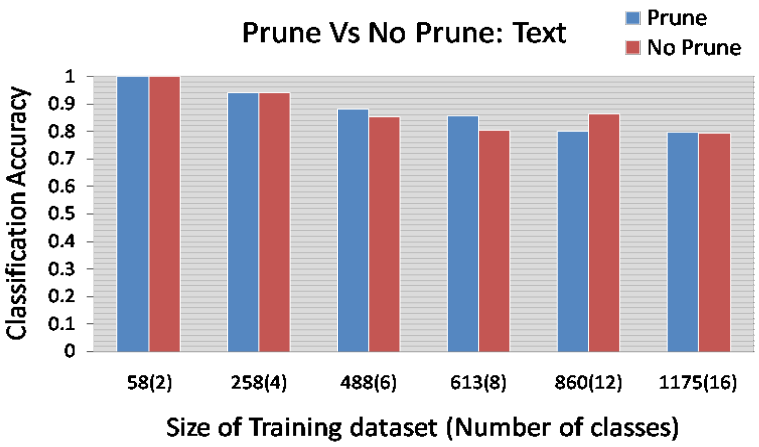

Figure 6.6 Prune Vs No Prune comparison

classification. Furthermore, the cost of processing all of the substructures only increases. Due to these reasons, a pruning option was developed where instances of slightly varying substructures were analyzed and pruned except for a single instance. Figure 6.6 illustrates the results on classification accuracy with the pruning option set on and with no pruning.

The approach combined with pruning turned out to show better result. This was due to the lesser number of substructures that were retained that still exhibited their corresponding class characteristics. The more the number of substructures retained, the more the chance for a test document to get misclassified. Therefore, our approach with the pruning option turned on gave a better accuracy.

With these experiments carried on the text corpus we are able to make claims that the graph mining with a global rank scheme compares and even outperforms a conventional text classifier in many cases with regard to multiple folder classification. As expected and suggested earlier, inexact graph match yields better classification results when compared to exact graph match. A tree representation with its superior structure showed better classification ability than the star representation consistently. Pruning of the substructures aided towards a better classification result. With these results, we

move on to show our result of our ranking approach with the global rank scheme on email collections.

6.3 Classification on Email Corpora

Although text classification techniques have also been applied to classify email messages, certain features of this domain that present challenges needs to be addressed. Many of these challenges have been outlined in Chapter 1. We have used various folders that were selected from public Listserv's and personal emails from different persons in order to provide a diversity. These several distinct folders' size varies from 10 to 470 odd emails. This email repository was collected to perform experimental analysis for the *InfoSift* system.

Additionally, to show that our approach is consistent and complete, experiments have also been carried on Enron Email Dataset [3]. The Enron email dataset was collected and prepared by the CALO [4] project. It comprises of data in the form of emails from about 150 users organized into folders. The email folders in this dataset have been cleaned and organized before it can be used for training the classifier. Some pre-processing steps include removing the non-topical folders(folders containing email messages regardless of their contents such as Inbox, Sent, Trash, Drafts, etc.), removing folders that are too small(does not contain any messages) or too large(contain more than 600 messages), etc. Experiments were conducted on both these data sets in order to show that our approach is compatible to different types of messages from the same domain. The results of our experiments are presented in the following sections.

6.3.1 Comparing Graph Mining with Naive Bayes

The performance of our approach is compared with Probabilistic Bayesian approach as done in text classification. Our approach does perform well irrespective of the type of

[3]Publicly available at http://www.cs.cmu.edu/ enron/

[4]More information at http://www.ai.sri.com/project/CALO

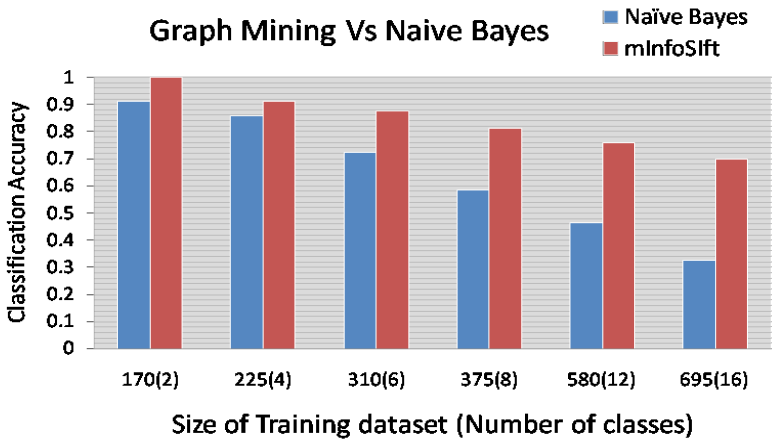

Figure 6.7 m-InfoSift Vs Naive Bayes for Listserv dataset

folders that were grouped together in the training data set because with the increase in the number of folders that were used to train the classifier, the diversity in folders also increased as all the folders of different sizes were collected as training set. Figure 6.7 shows the results comparing the accuracy of our approach against Naive Bayes for the Listserv dataset.

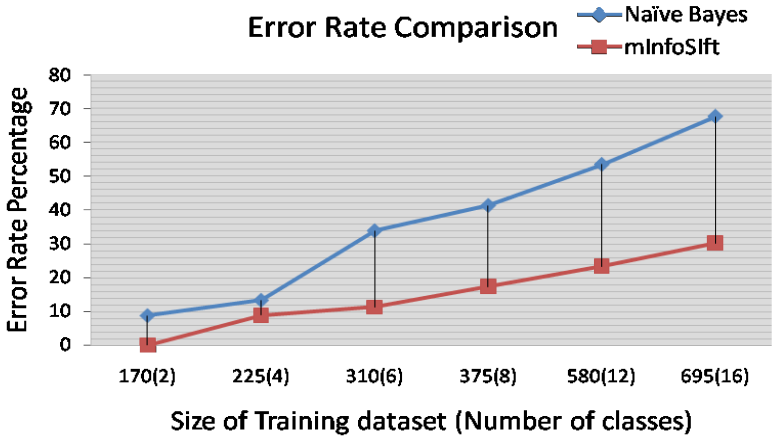

Figure 6.8 Error Rate comparison:m-InfoSift Vs Naive Bayes

The Bayesian classifier compared poorly due to large number of false positives. One of the main feature of the global rank scheme is the presence of fewer false positives when

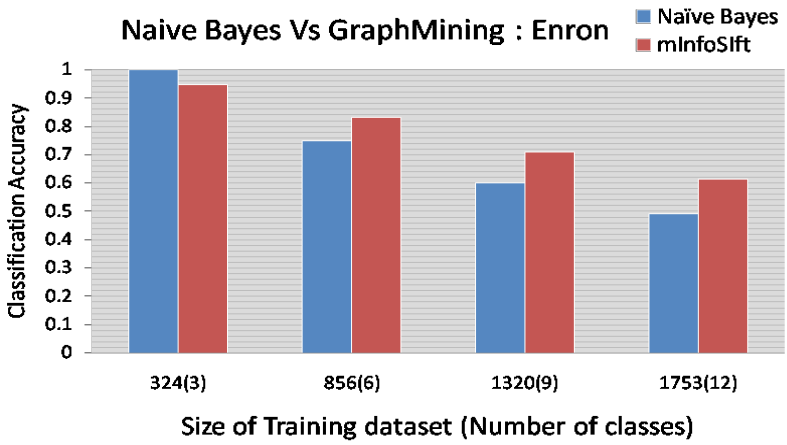

Figure 6.9 m-InfoSift Vs Naive Bayes for Enron dataset

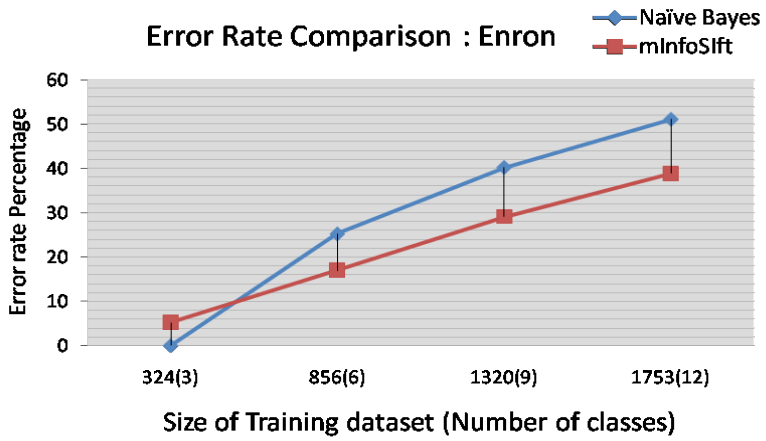

Figure 6.10 Error Rate comparison:m-InfoSift Vs Naive Bayes for Enron dataset

compared to a conventional classification technique such as the Naive Bayes approach. This is evident from the Figure 6.8.

The performance of graph mining classification is consistent across Enron dataset too. Though the difference in performance of both the classifiers for the Enron dataset is not as distinguishable when compared to that over Listserv dataset, our approach out performs Naive Bayes approach. The results for Enron dataset is presented in Figures 6.9 and 6.10.

As explained in Text classification, Naive Bayes assigns probabilities to word occurrences. Terms that are common to multiple folders and having a higher weight assignment

in one folder will outweigh others during classification which in turn leads to lot many wrongly classified emails. Our approach, on the other hand, identifies patterns of word occurrences and rank them across all the folders thereby avoiding this problem.

6.3.2 Effect of Feature Set Size

The features that comprise the email graphs are chosen from the top '*f%*'of the sum of frequencies in folders making up the training dataset. Experiments have been carried out with three different values as in text classification. A value of 60% or 80% as feature set retained only the top 60% or 80% of the terms that made in the frequent word list whereas a value of 100% would retain all in the frequent set of words. The results of the experiments are shown in Figures 6.11 and 6.12. As in the case of text classification, a similar observation was noted in the case of email classification.

Figure 6.11 Effect of Feature Subset Selection for Listserv dataset

The performance of the classifier was similar to the performance observed in test domains. When a value of 80% was used as the feature selection ratio, the performance of the classifier was consistently better than retaining all the words by using 100% or using a lower number of frequent words in using 60%. Though feature set size of 100% fared well when comapred to 80%, the amount of processing time during graph generation and

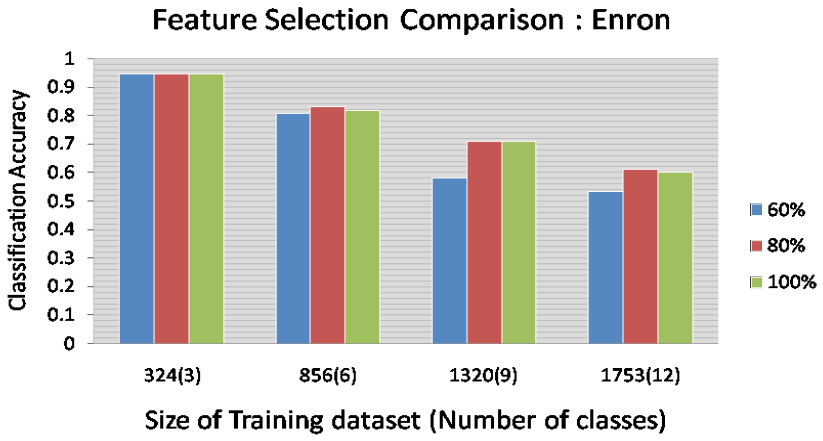

Figure 6.12 Effect of Feature Subset Selection for Enron dataset

substructure discovery is larger than in 80%. The classification accuracy reduces with the increase in the number of folders. This behavior is expected as the chances of email getting correctly classified to the right folder decreases with the increase in the number of substructures in the global rank list.

6.3.3 Exact Vs Inexact Graph Match

As in the case of text classification, inexact graph match exhibits better performance when compared to exact graph match. Emails do not correspond to a set of vocabulary and the information content of emails is relatively low as compared to text documents. Therefore, it is difficult to find exact patterns throughout the document class and the ability to match instances with slight variations becomes significant. Results for the comparison between exact and inexact graph match is shown in Figures 6.13 and 6.14, for Enron dataset.

The training data set contained folders of varying size and evidently inexact graph match performs better despite the heterogeneous email content for training the classifier. This differs from the exact match which groups instances that are identical, something that is hard to come by in a training data set with diverse content.

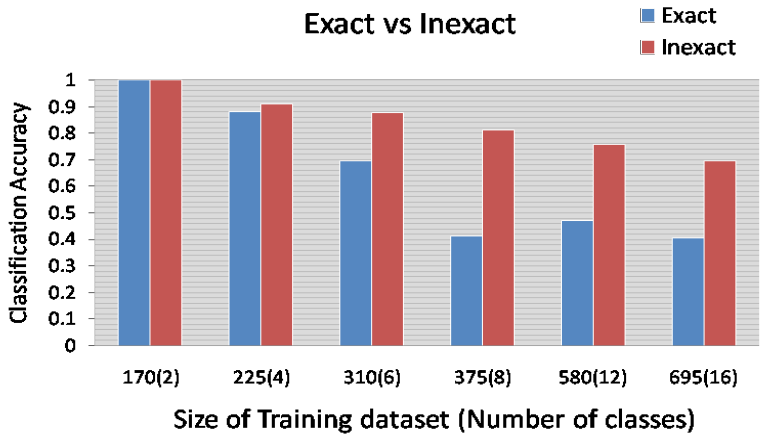

Figure 6.13 Exact Vs Inexact Substructure Discovery for Listserv dataset

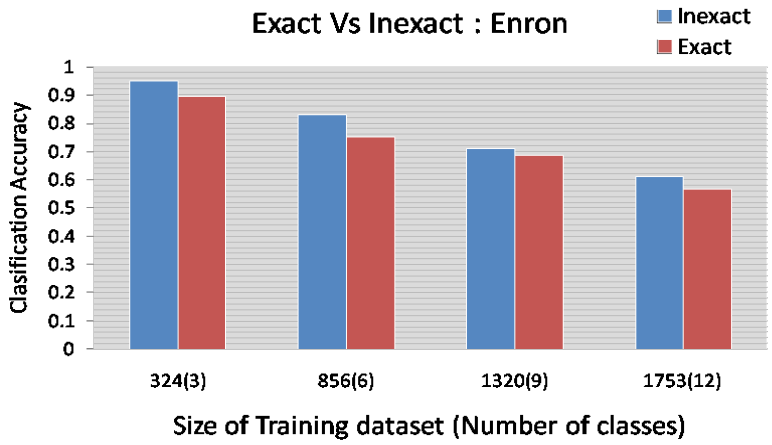

Figure 6.14 Exact Vs Inexact Substructure Discovery for Enron dataset

6.3.4 Tree Vs Star Representations

Experiments were conducted to study the effect of different graph representations on email classification. The tree representation showed better performance in classification as shown in Figures 6.15. This figure corresponds to the Listserv dataset collected for the *InfoSift* framework.

The difference in performance between star and tree representation is greater in email domain rather than in text classification. This is attributed to the enhanced structural information exhibited by email messages when compared to text documents. The tree representation represents emails, along with their structural information, in

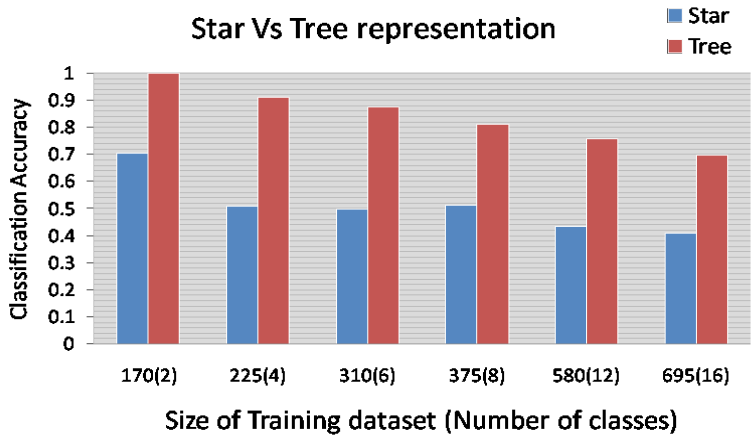

Figure 6.15 Star Vs Tree representation for Listserv dataset

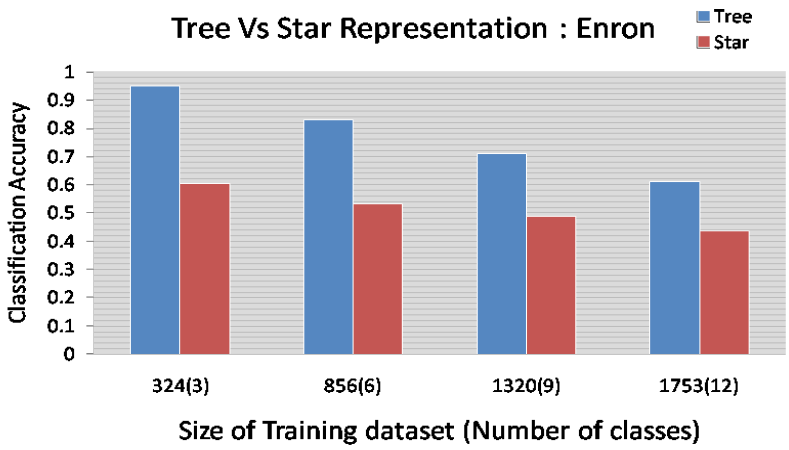

Figure 6.16 Star Vs Tree representation for Enron dataset

a better way than the star representation. This is reinforced with the results of the experiments conducted on Enron data set. Figure 6.16 illustrates the results of Tree Vs Star representations on Enron email data set. Evidently, tree representation has a better performance in classification than the star representation.

With the above experimental results, we can draw conclusions on email classification. The performance of the system is consistent though the classification accuracy reduces with the increase with the training and test data size, which is expected. These experimental results strengthens our argument that each domain has got structural information which can be exploited for classification purposes. These structural information

when translated to graphs using tree representation showed good performance for classification. Our document classification system was up to the challenge of finding similar substructures in email folders though emails in folders may not exhibit significant similar characteristics as they deal with diverse issues. This ability of find instances of similar substructures and grouping them is done using inexact graph match. In summary, the performance of the document classification system for multiple folders was consistent over various folder and email traits with which we validate our premise for the adaptation of graph mining techniques for classification.

6.4 Web Page Classification

For evaluation of web pages, we have conducted experiments on web collections called the *K-Series* [5]. The K-series consists of around 2,300 documents that belong to 20 different categories such as Art, Entertainment, Music, etc. A random selection of 850 documents have been used for experimental evaluation, and similar to the text classification methodology, Accuracy and Error rate are considered as performance metric.

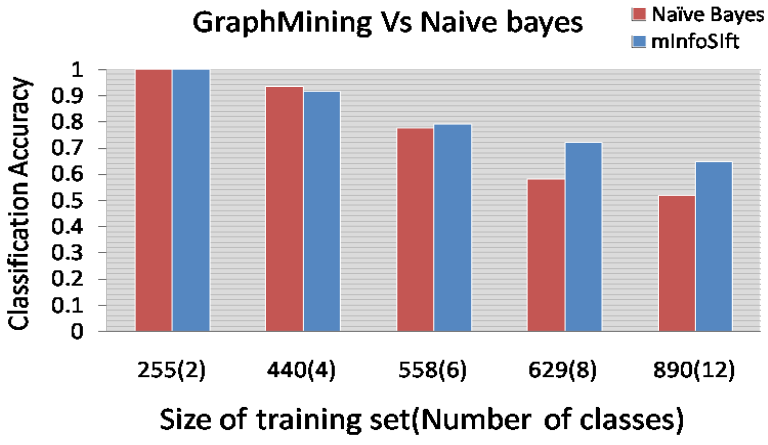

Figure 6.17 *m*-InfoSift Vs Naive Bayes for Web page classification

[5]publicly available at ftp://cs.umn.edu/users/boley/PDDPdata/

Experiments carried on the K-corpus for comparison with Naive Bayes is shown in Figure 6.17. The Naive bayes approach's performance was consistently below the performance of our document classification approach except for smaller training size. The error rate of test documents being wrongly classified is shown in Figure 6.18. The rate of test documents being wrongly classified is similar for a smaller training set. However, for a larger training set our graph approach consistently performed better.

Figure 6.18 Error rate comparison:m-InfoSift Vs Naive Bayes for Web page classification

The strength of our approach lies in ranking the substructures globally across all the document classes in the training set which in turn produces lower number of false positives when compared to Naive Bayes. With the increase in the number of distinct folders in the training set, the error rate also increases for both the approaches but when compared with each other, our approach exhibits a superior performance. Experiments were also conducted to study the classification due to exact and inexact graph match. The results are shown in Figure 6.19. As expected, the performance of inexact graph match was better than exact graph match.

As a summation, we have conducted exhaustive experiments across various domains and presented our results of our findings. The ranking scheme proposed and developed

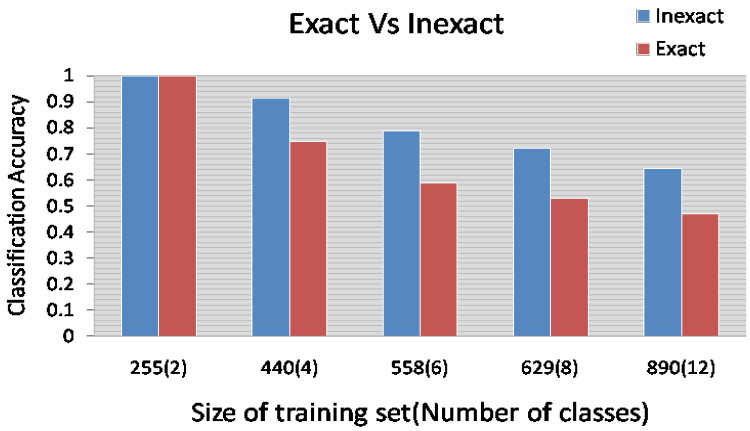

Figure 6.19 Exact Vs Inexact Graph Match for Web page classification

has shown consistent performance in terms of aiding multiple folder classification of documents.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis, we have proposed a new ranking technique that is appropriate for graphs and substructures that works well for multi-folder classification. The proposed technique ranks the representative substructures generated from each document class in the training set. This scheme of ranking overcomes the problem of having to depend on the size of the folders in the training set which is common in conventional probability based classification techniques. The ranking formula developed and tested in this thesis is a generalization of the formula used for single folder classification. In other words, this formula can also be applied for single folder classification to determine the confidence with which an incoming document can be assigned to a folder.

The classifier based on the ranking works well with several textual domains such as text repositories, email folders, and web pages collections as has been shown experimentally. The validity of the global ranking technique has been established by the consistent performance of the classifier over these domains. Various parameters that affect the ranking and therefore classification of the test documents have been identified and analyzed in detail. The results of our approach validate the effectiveness of the ranking technique to adapt multiple category classification for the existing *InfoSift* framework.

Additional document preprocessing approaches like Stemming were incorporated with the existing techniques like stop word elimination. The concept of feature subset selection enables us to classify data even when the amount of data available for training purpose is insufficient. Graph representations like tree and star have been studied in order to represent the documents in training and test set to incorporate useful domain information for classifying unknown samples. Inexact graph match forms the main basis for classifying test document even when the training set exhibit diverse characteristics.

The ability to match instances of similar substructures as the same substructure boosts the chances of a test document to get classified. Experiments have been conducted with different graph representations on domains such as text, email and web pages and it can be ascertained from that results that the tree structure gives a better accuracy. Different values of thresholds have been employed during substructure discovery and classification and the value of 0.1 for discovery process and 0.05 during classification showed better performance. The classification threshold is lower than the discovery threshold in order to reduce the misclassification of test documents. Finally,our approach has also been shown to work for documents from different domains experimentally.

Although the performance of the classifier system is as expected of a classifier that uses a mining subsystem, further work is needed to reduce the error rate to lower misclassification of the test documents. Some of the enhancements that can be done are outlined in the following discussion. Though the Subdue system is used for substructure discovery, it is not directly suitable nor built for classification tasks. Other classifying techniques which take structure of the content into account can be delved upon to improve classification accuracy. Currently, the pruning of the substructure is done after the discovery of the substructures outside Subdue. In order to reduce the processing time, pruning can be incorporated along with the substructure discovery process in Subdue. Substructures with same MDL, frequency and size can be pruned once they are discovered rather than pruning them separately to save time. A more detailed analysis of the data sets can be done in order to derive more characteristics for better classification. For example, the amount of diversity among the documents of the same class, the number of heterogeneous or homogeneous documents in the folder, etc. needs to be looked upon before it can be part of the training data set. Current graph representations have no means to differentially weigh different parts of the graph. For instance, a greater significance can be attached to the words in the title of the document rather when compared to the words in the body. A scheme incorporating this concept can be used for better classification.

In the case of email classification, the adaptation of the classifier to the changes in the email folders is critical to achieve good classification accuracy. The current system does not deal with changes as the whole classification scheme is based on static training data set. Therefore, incremental learning mechanisms, such as selective learning and batch learning, can be added in order for the classifier to adapt to the dynamic changes in the email domain. The basic assumption in our approach of document classification has been that an incoming test document belongs to a single folder only and the classifier matches the test document with a single folder only. In the future, the test document can be tried to classified to multiple folders using similarity techniques.

Currently, the accuracy of the document classification system has been compared with Naive Bayes approach alone. Comparison besides naive Bayes or other ranking techniques for the purpose of classification will reveal useful insights to enhance performance. The future work also includes the development of a graphical user interface for the email classifier to be coupled with an email agent. While this thesis focuses on the need for a global ranking scheme and establishes the ranking formula developed for classification of documents in textual domains, other application domains needs to be investigated. For example, our ranking scheme can be adapted to check if a document already exists in multiple categories in a patent database.

In conclusion, we believe that the global ranking technique developed will bridge the gap between adapting graph mining techniques and document classification in regard to multiple folder document classification.

REFERENCES

[1] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining.* Addison Wesley, 2006.

[2] M. Aery, "Infosift:adapting graph mining techniques for document classification," Master's thesis, The University of Texas at Arlington, 2004. [Online]. Available: http://www.cse.uta.edu/research/publications/Downloads/CSE-2003-39.pdf

[3] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," *IEEE International Conference on Data Mining*, pp. 313–320, 2001.

[4] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," *Journal of Artificial Intelligence Research*, vol. 1, pp. 231–255, 1994. [Online]. Available: citeseer.ist.psu.edu/article/cook94substructure.html

[5] X. Yan and J. Han, "gspan:graph-based substructure pattern mining," *Proceedings of the IEEE International Conference on Data Mining*, 2002.

[6] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *ECML*, pp. 137–142, 1998.

[7] C. Apte, F. Damerau, and S. M. Weiss, "Text mining with decision trees and decision rules," *Conference on Automated Learning and Discovery*, 1998.

[8] I. Molinier, "Is learning bias an issus on the text categorization problem?" *Technical Report LAFORIA-LIP6, Universite Paris VI*, 1997.

[9] W. Lam and C. Ho, "Using a generalized instance set for automatic text categorization," *In the Proceedings of the 21st Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98)*, pp. 81–89, 1998.

[10] B. Masand, G. Linoff, and D. Waltz, "Classifying news stories using memory based reasoning," *In the 15th Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'92)*, pp. 59–64, 1992.

[11] Y. Yang, "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," *In the 17th Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'94)*, pp. 13–22, 1994.

[12] Y. Yang and C. G. Chute, "An example-based mapping method for text categorization and retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 12(3), pp. 252–277, 1994.

[13] C. Apte, F.Damerau, and S. Weiss, "Towards language independent automated learning of text categorization models," *In the Proceedings of the 17th Annual ACM/SIGIR conference*, 1994.

[14] W. W. Cohen, "Text categorization and relational learning," *In the Twelfth International Conference on Machine Learning (ICML'95)*, 1995.

[15] W. W. Cohen and Y. Singer, "Context-sensitive methods for text categorization," *In SIGIR'96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development of Information Retrieval*, pp. 307–315, 1996.

[16] I. Moulinier, G. Raskinis, and J. Ganascia, "Text categorization: a symbolic approach," *In the Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval*, 1996.

[17] E. D. Wiener, J. O. Pedersen, and A. S. Weigend, "A neural network approach to topic spotting," in *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, 1995, pp. 317–332. [Online]. Available: citeseer.ist.psu.edu/wiener95neural.html

[18] H. T. Ng, W. B. Goh, and K. L. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, A. D. Narasimhalu, and P. Willett, Eds. Philadelphia, US: ACM Press, New York, US, 1997, pp. 67–73. [Online]. Available: citeseer.ist.psu.edu/ng97feature.html

[19] A. K. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," *In the 15th Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'92)*, pp. 59–64, 1992.

[20] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text categorization," *In Proceedings of the 21st Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98)*, pp. 96–103, 1998.

[21] D. Koller and M. Sahami, "Heirarchically classifying text using very few words," *In the 14th International Conference on Machine Learning (ICML'97)*, pp. 170–178, 1997.

[22] K. Tzeras and S. Hartman, "Automatic indexing based on bayesian inference networks," *In the 16th Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'93)*, pp. 22–34, 1993.

[23] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, pp. 947–980, 1991.

[24] Y. Yang and X. Liu, "A re-examination of text categorization methods," *ACM SIGIR*, pp. 42–49, 1999.

[25] V. N. Vapnik, *The Nature of Statistical Learning Theory.* Springler, New York, 1995.

[26] ——, *Estimate of Dependences based on Emperical Data [In Russian].* Nauka, Moscow, 1979. (English Translation Springer Verlag, New York, 1982).

[27] R. B. Segal and J. O. Kephart, "Swiftfile: An intelligent assistant for organizing e-mail," *Proceedings of AAAI 2000 Spring Symposium on Adaptive User Interfaces*, pp. 107–112, 2000.

[28] J. D. M.Rennie, "ifile:an application of machine learning to e-mail filtering," *Proceedings of KDD-2000 Text Mining Workshop,Boston Aug*, 2000.

[29] M. Sahami, D. Heckerman, and E. Horovitz, "A bayesian approach to filtering junk e-mail," *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[30] G. Boone, "Concept Features in Re:Agent, an Intelligent Email Agent," in *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, K. P. Sycara and M. Wooldridge, Eds. New York: ACM Press, 9–13, 1998, pp. 141–148. [Online]. Available: citeseer.ist.psu.edu/boone98concept.html

[31] T. Payne and P. Edwards, "Interface agents that learn: An investigation of learning issues in a mail agent interface," *Applied Artificial Intelligence*, pp. 1–32, 1997.

[32] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine Learning*, pp. 261–283, 1989.

[33] W. Cohen, "Learning to classify English text with ILP methods," in *Advances in Inductive Logic Programming*, L. De Raedt, Ed. IOS Press, 1996, pp. 124–143. [Online]. Available: citeseer.ist.psu.edu/cohen96learning.html

[34] W. W. Cohen, "Fast effective rule induction," *In Machine Learning: Proceedings of the Twelfth International Conference, Lake Taho, California, Morgan Kaufmann*, 1995.

[35] E. Crawford, J. Kay, and E. McCreath, "Automatic induction of rules for e-mail classification," *Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia*, 2001.

[36] J. Heflman and C. Isbell, "Ishmail: Immediate identification of important information, at&t labs," 1995.

[37] J. Catlett, "Megainduction: A test flight," *International Conference on Machine Learning*, 1991.

[38] *Email Classification with Temporal Features*, 2004.

[39] G. Attardi, A. Gulli, and F. Sebastiani, "Automatic we page categorization by link and context analysis," *In Chris Hutchison and Gaetano Lanzarone (eds.), Proc. of THAI'99*, pp. 105–119, 1999.

[40] A. Schenker, M. Last, H. Bunke, and A. Kandel, "Classification of web documents using a graph model," *icdar*, vol. 01, p. 240, 2003.

[41] H. Bunke and K. Shearer, "A graph distance metric based on maximal common subgraph," *Pattern Recognition Letters*, pp. 753–758, 2001.

[42] D. Tax and R. Duin, "Using two-class classifiers for multiclass classification," 2002, pp. II: 124–127.

[43] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 9–16. [Online]. Available: citeseer.ist.psu.edu/allwein00reducing.html

[44] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10. The MIT Press, 1998. [Online]. Available: citeseer.ist.psu.edu/hastie98classification.html

[45] A. Berger, "Error-correcting output coding for text classification," 1999. [Online]. Available: citeseer.ist.psu.edu/article/berger99errorcorrecting.html

[46] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995. [Online]. Available: citeseer.ist.psu.edu/dietterich95solving.html

[47] E. B. Kong and T. G. Dietterich, "Error-correcting output coding corrects bias and variance," in *International Conference on Machine Learning*, 1995, pp. 313–321. [Online]. Available: citeseer.ist.psu.edu/kong95errorcorrecting.html

[48] N. S. Ketkar, L. B. Holder, and D. J. Cook, "Subdue: compression-based frequent pattern discovery in graph data," in *OSDM '05: Proceedings of the 1st international workshop on open source data mining*. New York, NY, USA: ACM Press, 2005, pp. 71–76.

[49] J. Rissanen, "Stochastic complexity in statistical enquiry," *World Publishing Company*, 1989.

[50] H. Bunke and G. Allerman, "Inexact graph match for structural pattern recognition," *Pattern Recognition Letters*, pp. 245–253, 1983.

[51] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection." *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

BIOGRAPHICAL STATEMENT

Aravind Venkatachalam was born in Fahaheel, Kuwait in 1982. He did his primary schooling in Kuwait and his secondary schooling in India. He received his Bachelors of Technology degree from the University of Madras in 2004 where he won the 'Best Student Award'. His interest in research brought him to The University of Texas at Arlington where he later obtained his Masters degree in Computer Science and Engineering in 2007. His interest includes Data Mining, Information Retrieval and Information Integration.