

THE UNIVERSITY OF CALGARY

XML Document Classification Using Structural and Textual Features

by

Mohammad Khabbazzhaye Tajer

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

August, 2008

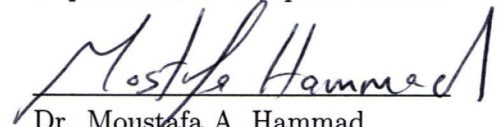
© Mohammad Khabbazzhaye Tajer 2008

**THE UNIVERSITY OF CALGARY**  
**FACULTY OF GRADUATE STUDIES**

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "XML Document Classification Using Structural and Textual Features" submitted by Mohammad Khabbazhaye Tajer in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.



Supervisor, Dr. Reda Alhajj  
Department of Computer Science



Dr. Moustafa A. Hammad  
Department of Computer Science



Dr. Vahid Garousi  
Department of Electrical and  
Computer Engineering

8/26/2008

Date

## Abstract

This thesis addresses XML document classification by considering both structural and content based features of documents. This leads to more informative feature vectors that better represent documents from different perspectives. To manage the feature space better, we integrate soft clustering and feature reduction into the process. In order to extract structural information, we use existing rule mining algorithms to capture frequent structural patterns in the form of rules and later convert them to structural features. However, for extracting content information of XML documents, we propose a new method based on soft clustering of words and using each cluster as a textual feature. We show that the classifier built only using our textual features outperforms some well-known information retrieval (IR) based document classification technique. Further, the combination of structural and textual features results in an accurate and robust classifier. We demonstrate the efficiency and effectiveness of incorporating both structural and content information by performing a comparative analysis of our classifier model and several other document classifiers including XML document classifiers.

**Index Terms:** XML document, classification, structural features, content features, soft clustering, information retrieval.

## Acknowledgements

This project could not have been accomplished without the gracious help and supervision of my advisor, Dr. Reda Alhajj. I would like to thank him for his time and patience and also for his example, as a professor and as a man. I am also grateful to Dr. Moustafa Hammad and Dr. Vahid Garousi, my committee members, for their time and effort in reviewing this work and their valuable feedback.

I would like to acknowledge my great friend, Keivan Kianmehr, who has never had any reservation to share his experience and deep knowledge with me.

I owe thanks to my family, my parents and sister, who have always encouraged me and been supportive of me in pursuing my dreams.

# Table of Contents

Approval Page	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Definition and Motivation . . . . .	1
1.2 The Proposed Solution . . . . .	4
1.3 Contribution . . . . .	9
1.4 Thesis Organization . . . . .	10
<b>2 Background and Related Work</b>	<b>12</b>
2.1 XML Basics . . . . .	12
2.2 Classification and Feature Reduction . . . . .	16
2.2.1 Classification . . . . .	16
2.2.2 Feature Reduction . . . . .	18
2.3 Content-based XML Classification . . . . .	22
2.4 Structure-based XML Classification . . . . .	26
2.5 Our Framework in Comparison with Previous Work . . . . .	29
<b>3 Feature Extraction from Unstructured and Semi-Structured Documents</b>	<b>30</b>
3.1 Feature Spaces for XML Document . . . . .	30
3.2 Structural Feature Construction . . . . .	33
3.2.1 Structural Rule Mining . . . . .	33
3.2.2 Constructing Structural Feature Vectors . . . . .	35
3.3 Textual Feature Construction . . . . .	36
3.3.1 Document Preprocessing . . . . .	38
3.3.2 Original Document Set vs. Inverted Index . . . . .	39
3.3.3 Supervised Word-Vector Formation . . . . .	40
3.3.4 Clustering of words . . . . .	42
3.3.5 Fuzzy Clustering of words and Constructing the Final Feature Vector Set . . . . .	44
3.4 Building Classifier Model . . . . .	46

<b>4</b>	<b>Creating a Hierarchy of Subjects Using Support Vector Machines and 10 Fold Cross Validation</b>	<b>47</b>
4.1	Creating Classifier models for Soft Classification . . . . .	48
4.2	Calculating the Distance between Subject Categories . . . . .	49
4.3	Hierarchical Clustering of Subject Categories . . . . .	50
<b>5</b>	<b>Experimental Analysis</b>	<b>54</b>
5.1	The Datasets . . . . .	54
5.2	Feature Extraction from XML Datasets and Classification . . . . .	57
5.3	Text Classification on 20NewsGroups Dataset . . . . .	63
5.4	Automatical Hierarchy Generation from the 20NewsGroupos Dataset	69
<b>6</b>	<b>Summary, Conclusion &amp; Future Work</b>	<b>76</b>
6.1	Summary . . . . .	76
6.2	Conclusion . . . . .	77
6.3	Future Work . . . . .	78
	<b>Bibliography</b>	<b>81</b>

## List of Tables

5.1	Subject categories of 20NewsGroups dataset partitioned into 6 top level classes . . . . .	55
5.2	Characteristics of XML datasets . . . . .	57
5.3	Classification results using the whole space and different methods . .	61
5.4	Comparison of IR classifiers . . . . .	62
5.5	Comparing the accuracy of Feature based classification to structural rule based classifiers . . . . .	63
5.6	Most similar classes in 20NewsGroups dataset based on our SVR-based distance measure . . . . .	70
5.7	Steps of the Single-linkage hierarchical clustering algorithm . . . . .	71
5.8	8 high level categories of 20NewsGroups dataset proposed in [58], created by linear projection. . . . .	73
5.9	8 high level categories of 20NewsGroups dataset created by our algorithm using Support Vector Regression Machine. . . . .	74

## List of Figures

1.1	Using clusters as textual and frequent sub-structures as structural features . . . . .	6
1.2	The Architecture of The Automatic XML Document Classifier Model . . . . .	8
2.1	Structure of a sample XML document . . . . .	14
2.2	Bookstore elements with different number of books . . . . .	14
4.1	Process of Hierarchical Clustering of Objects . . . . .	52
5.1	Number of Clusters vs. Accuracy on LOG1 Dataset . . . . .	59
5.2	Combining Textual and Structural Features . . . . .	60
5.3	Comparison of Naive-BOW and Cluster-BOW using 10-Fold Cross Validation and different number of words . . . . .	64
5.4	Size of the training set using different number of words for classification . . . . .	67
5.5	Automatically generated hierarchy from 20NewsGroups dataset . . . . .	72
5.6	Automatically generated hierarchy from 20NewsGroups dataset . . . . .	75



# Chapter 1

## Introduction

A growing volume of structured and unstructured documents do exist and should be better organized for effective usage. This requires the development of some systematic and automated techniques capable of classifying the documents based on their features. One such effective technique is described in this thesis. Before diving into the details of the proposed approach, this chapter is dedicated to introduce the problem, motivate for the work done, and enumerate the contributions.

### 1.1 Problem Definition and Motivation

XML (eXtensible Markup Language) has emerged as a recommendation of World Wide Web Consortium (W3C) and is being gradually accepted as the standard format for data exchange between different platforms and partners over the web. Further, because of its semi-structural nature, there has been great interest to store large data repositories, *e.g.*, digital libraries, as XML documents. As the amount of information available in XML format grows, data mining techniques have been widely considered to better organize XML documents for effective browsing and efficient search. In this regard, classification has been recently applied to XML documents [28, 29, 44, 45].

Classification is known as supervised learning technique which involves considering some predefined classes and significant features of a set of data objects to build a

classifier model capable of predicting the class of new data objects having the same set of features used in building the classifier. When the classes are not known, the process turns into unsupervised learning and the target is to find the classes in a process known as clustering.

The goal of XML document classification is to build a classifier model that can automatically assign XML documents to some existing categories. Most of the existing XML document classifiers either work solely based on information retrieval methods and ignore the structural patterns existing in the document tag-structure or the other way around. In general, feature reduction [17] has been successfully utilized as a preprocessing phase to reduce the complexity of classification models and to improve the accuracy of the classifier.

The work described in this thesis attempts to capture both content and structural information of XML documents in a way to integrate them into the process of learning and predicting the classifier model. Our main target is to convert document classification into a classical classification problem by effective feature extraction from different aspects of the document, and then apply any model or improved technique in order to build the desired classifier. Feature extraction gives us the flexibility of approaching the problem using a wider range of applicable techniques and taking advantage of the extensive research described in literature on classical classification problems.

Building a reliable and accurate classifier system that could be used in practice requires considering all important aspects of different document types. Furthermore, a practical document classifier or classification framework has to be extensible and scalable enough to perform in dynamic environment as the number of existing doc-

uments of different types is growing every moment. Developing dynamic classifiers requires extra effort to keep the classifier capable of maintaining high accuracy. Accuracy degrades as the new documents deviate from the characteristics of the set of documents used in the training. A dynamic classifier watches the change in the characteristics and adapts itself to cover the corresponding documents; such a dynamic classifier is out of the scope of this thesis and could be considered in future to expand the static classifier into dynamic one after we tuneup the static classifier and analyze all its aspects as covered in this thesis. Actually, most of the previous research in the area has focused on static classification of text documents because they concentrate on testing the outcome using existing datasets, which are generally static.

Our general idea is also applicable to other kinds of documents that have a structure; although the details of feature extraction would be different for other types of documents. For example, if the task is HTML document classification, we could also take into account features of HTML documents other than text (like links to other pages in the document or images used in the document, etc.). However, we have particularly focused on the classification of XML documents in this thesis. We want to show how it is possible to extract from XML documents informative features that could be used for building an accurate classifier model. The conducted experiments demonstrate the effectiveness and applicability of the proposed integrated approach.

Although our framework is extensible for dynamic document classification, we have concentrated on static classification where the set of documents is assumed to be fixed in order to initially validate our proposed ideas. We have briefly explained the additional steps required for online and dynamic classification in section 6.3 and

left dynamic document classification as future work.

Finally, we use feature vector representation of documents combined with Support Vector Regression (SVR) machines [56] for creating a hierarchy of documents. Using SVR we will define a distance measure between classes and use distance measure to hierarchically cluster the categories. There are several reasons for converting the flat structure of categories to a hierarchy; among which are: 1) Instead of querying a flat set of documents, many users prefer to browse hierarchical directories and issue queries in specific sub-areas. 2) It can help us to create high level categories of documents and cluster the subjects by cutting the hierarchy at any desired level. Clustering of the subjects could reduce the complexity of classification problems by grouping the similar classes of documents in one class. This technique could be useful in problems where all of the low level classes are not required and a high level grouping of subjects is enough.

## 1.2 The Proposed Solution

In this section, we describe our approach for building an automatic XML document classification model. Our main attempt is to incorporate both structure and content information of XML documents into the process of building the classifier model by constructing structural and textual features.

The feature extractor has two main components, each covers one aspect of the document. Resulted feature vectors from different aspects are concatenated later, and every document is represented by a single vector. On the other hand, the content-based feature extractor component clusters the words according to their distribution

over the set of class labels, and then computes the degree of association between documents and clusters. Since every cluster represents a feature or a group of words, the choice of clustering method could greatly affect the accuracy of classification.

Hard clustering algorithms assign every object to only one cluster, while soft clustering algorithms try to assign every object to several clusters with different membership degrees. We believe that the choice of soft or hard clustering of words depends on the average number of words per document. In case of a corpus that has much more words than documents, *i.e.*, documents are very large (simply contain tens of thousands of words), we could argue that a single word is a small atomic unit, and therefore hard clustering is preferable. On the other hand, if documents are not large, soft clustering seems to be more natural choice, mainly because the closeness of documents to clusters (features) is not underestimated or overestimated due to ignoring partial memberships in clusters.

Since the size of typical XML documents available on the web (such as RSS News Feeds) is not huge in practice, we would rather use *Fuzzy C-Means Clustering* [21] as a soft clustering algorithm after mapping every word to a vector according to its distribution over the set of class labels. If the number of class labels is small soft clustering of words will spread the feature space over a larger number of clusters and separate the words from each other according to their association with categories of subjects. On the other hand, when the number of class labels is large, each word vector will have more dimensions and it will result in a degradation in the quality of clusters. In such cases, we prefer to use the normalized distribution of words over classes as soft clusters and skip the clustering phase.

As for the structural feature extraction, we convert the tag-structure of the doc-

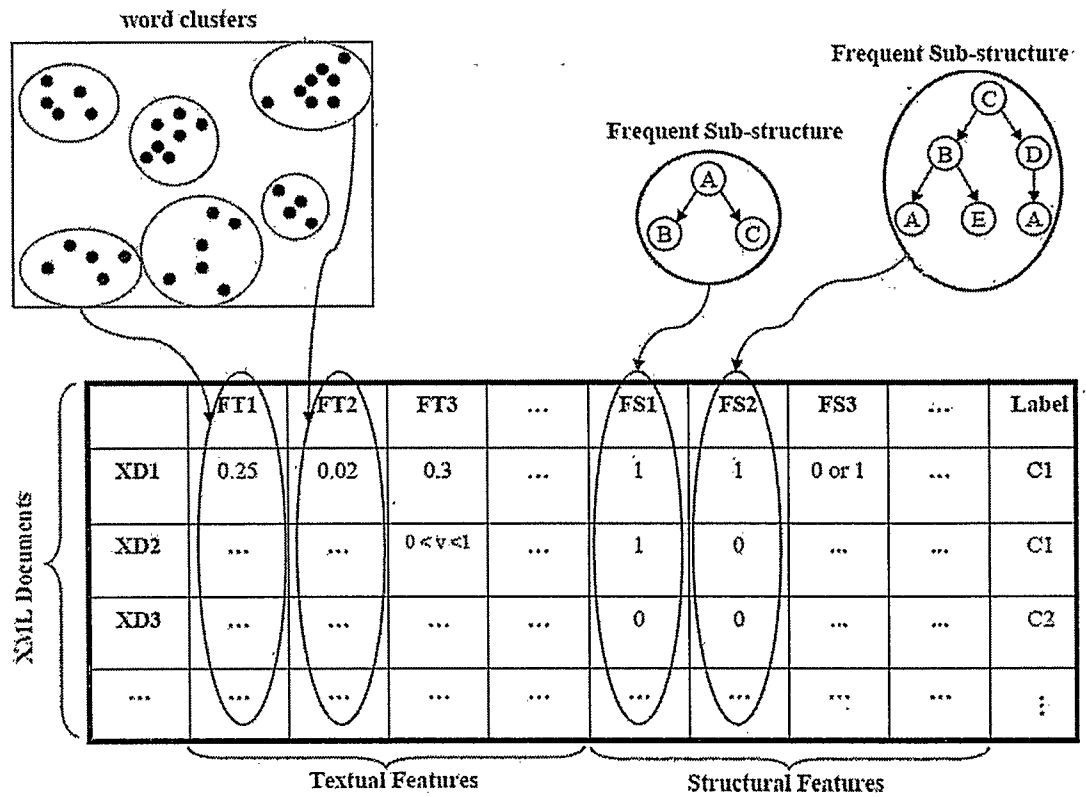


Figure 1.1: Using clusters as textual and frequent sub-structures as structural features

ument into a tree according to parent-child relationships, mine frequent sub-trees existing in the document tree and use each frequent sub-tree as a feature. Frequent pattern mining concepts have already been successfully adapted and used in the context of frequent tree mining [34]. Figure 1.1 depicts our general idea of feature based XML document classification and also a very high level view of feature extraction process in a nutshell. We have assigned hypothetical values to features in order to show the range of values for different types of features.

Figure 1.2 depicts the architecture of the automatic XML documents classifier model and the data flow within the system. Our proposed model consists of three main components. The first component handles structural feature construction. A rule mining algorithm extracts all frequent structural patterns from XML documents and builds structural rules. Structural rules are then converted to features to form a set of structural feature vectors representing structural information of XML documents. Textual feature construction is performed by the second component of the proposed model. Inverted index representation of the given XML documents is used to build a set of word vectors according to the occurrences of words in different documents. Fuzzy C-Means clustering algorithm is then applied to word vectors to generate clusters of words. These clusters are later used as textual features to create a set of textual feature vectors describing content information of XML documents. Eventually, the third component takes feature vectors as input to a supervised learning algorithm to build the final classifier model.

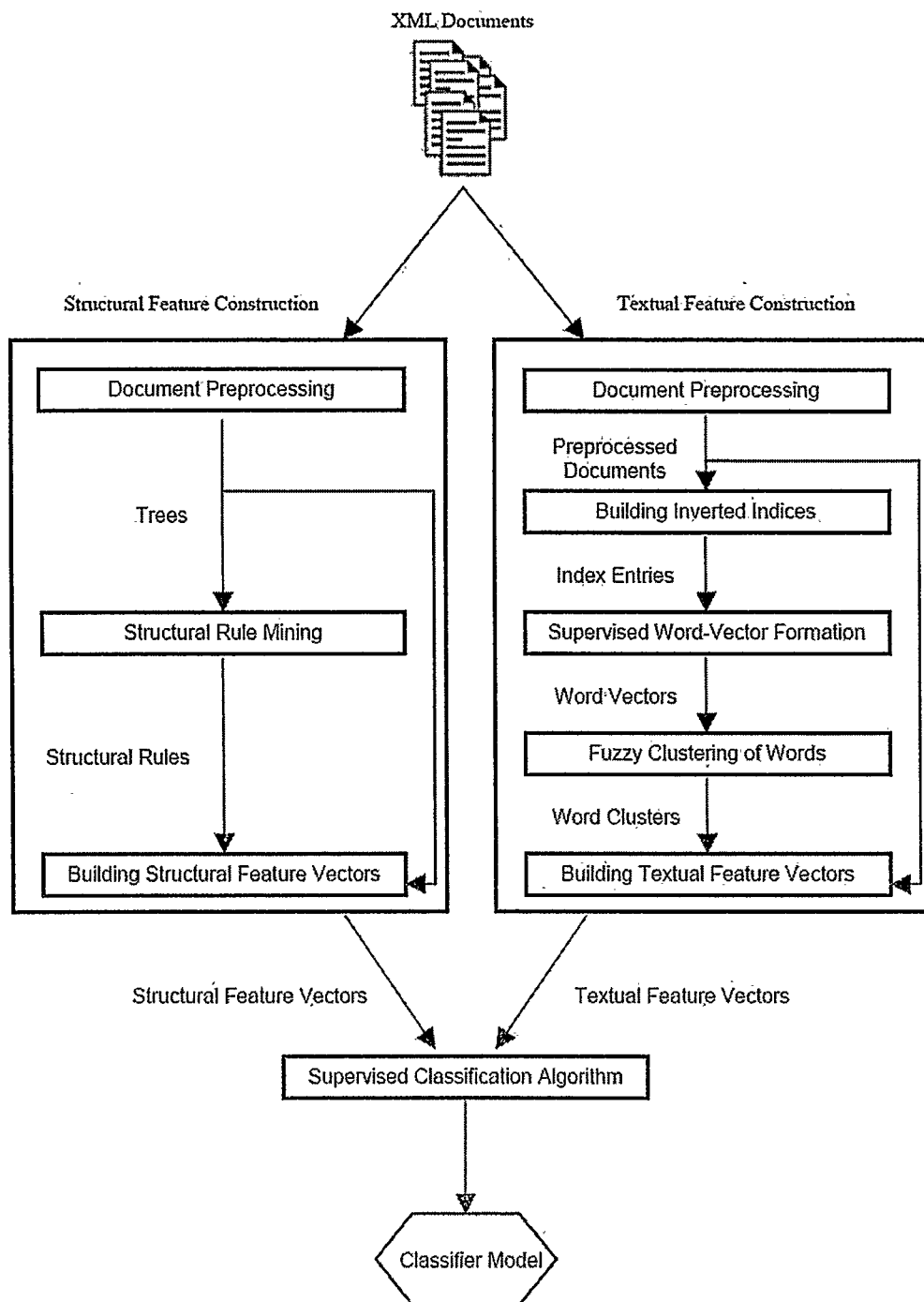


Figure 1.2: The Architecture of The Automatic XML Document Classifier Model



### 1.3 Contribution

The model described in this thesis might be considered as a major contribution in feature reduction and document classification. The main contributions of this thesis could be enumerated as follows:

- A general feature based framework is proposed for document classification. The proposed framework could be used for any kind of document. We will demonstrate the applicability of our framework on plain text and also semi-structured XML documents.
- A new approach for feature extraction from text documents is proposed based on using the distribution of words over the set of class labels and Fuzzy C-Means clustering. Experiments show how soft clustering achieves better compression of the feature space when compared to a hard clustering algorithm such as K-Means clustering.
- We adapt structure based classification techniques for structural feature extraction from documents, and convert the coverage of rules on documents to features. The extracted features are then concatenated with textual features and form the final feature vector for XML documents.
- Using feature based representation of documents, we propose a way of automatically building a hierarchy of subjects and clustering of subject(*class labels*). Automatic hierarchy creation facilitates browsing and searching for users and also can group small categories of subjects into larger ones according to content

similarities and could reduce the complexity of classification when the number of class labels is large.

## 1.4 Thesis Organization

The material presented above in this chapter motivated for the tackled problem and enumerated the major contribution of the proposed solution. For the reader to get the global picture, this section highlights the basic content of the other chapters of this thesis.

Chapter 2 is an overview of existing XML classification techniques and also provides the background required to understand the rest of the thesis. We will describe different classification approaches some of which are text based and some are structure based. We also give an overview of several text classification approaches and feature reduction techniques in the context of text classification.

Chapter 3 describes the feature spaces designed to capture structural and content information of XML documents and shows how textual and structural features are defined and extracted from documents. Textual feature extraction technique described based on soft clustering of words could be also used for text classification.

Chapter 4 explains our approach to build an automatic hierarchy of classes using the feature vector representation of documents obtained from previous chapter using SVR to measure the similarity of classes and hierarchical clustering.

Chapter 5 presents the selected evaluation model, the conducted experiments and the achieved results. The results of several experiments are reported in this chapter to show the effectiveness of clustering over other feature reduction techniques, as well

as the strength of our proposed technique for XML document classification compared to previous work. The conducted experiments also show how Fuzzy-C-means as a soft clustering technique achieves better compression of feature space compared to hard clustering algorithms.

Chapter 6 is summary, conclusions and future research directions.

## Chapter 2

### Background and Related Work

Data mining is the process of extracting hidden knowledge underlying the data accumulated over time [3]. Knowledge could be extracted in several predefined forms and used to make predictions or decisions for the future. In order to extract knowledge, machine learning techniques and models are frequently used. Different forms of data mining tasks involve classification, clustering, prediction, association rule mining, outlier mining *etc.* [4].

In this thesis we focus on the problem of Document Classification. Specifically, XML document classification. Among the data mining tasks mentioned above we have employed classification, feature reduction and clustering. As described in the literature, XML document classification includes two major techniques: content-based classification and structure-based classification. Alternative approaches have been also proposed to combine different techniques in order to improve classification performance and efficiency. In the rest of this chapter, we will provide the necessary background information and describe several XML classification work within each of the two categories.

#### 2.1 XML Basics

A document in XML format by definition is a well-formed document that contains one or more elements. In other words, a document starts from exactly one element

called the root element and all other elements have to start and end in the scope of other elements to keep the document well-formed; every non-root element has a parent. There could also be other attributes associated to each element for further description. Each element of the document contains either other elements or data that could be in different formats [2].

Elements of the document are opened and closed by tags; and the tag structure could continue to any level without any restriction as long as it conforms to the Document Type Definition (DTD) stored in a file with DTD extension. In this sense, XML documents are called to be semi-structured documents in comparison with fully structured data stored in relational databases. The structure of the XML document has to be well-formed, but there is no general form that all elements must follow. Every XML document has a tree-like structure according to the parent-child relationship existing between its attributes. Let's see what we mean by tree structure and semi-structured nature of XML documents. Figure 2.1 shows an example of the tree structure.

Every *element* node represents a tag, and associated with each tag there are attributes. The whole tree structure represents the schema of the document and parent child relationships. As it is shown in Figure 2.1, nodes that are under the same parent are called sibling nodes. Based on the given XML schema intended to store data about bookstores, there could be several documents each of which has its own specific structure. Root element of the document is `< bookstores >`, and under the root element there could be several `< bookstore >` elements. Figure 2.2 demonstrates two different bookstores.

From the XML schema, we know that under `< bookstore >` element we should

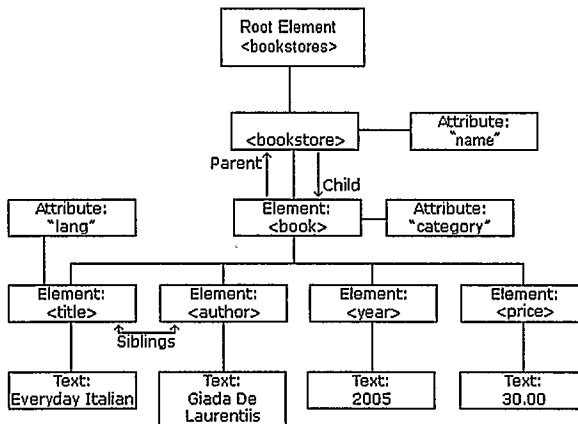


Figure 2.1: Structure of a sample XML document

<pre> &lt;bookstore name = "Chapters"&gt; &lt;book category="COOKING"&gt;   &lt;title lang="en"&gt;     Everyday Italian   &lt;/title&gt;   &lt;author&gt;     Giada De Laurentiis   &lt;/author&gt;   &lt;year&gt;2005&lt;/year&gt;   &lt;price&gt;30.00&lt;/price&gt; &lt;/book&gt; &lt;book category="CHILDREN"&gt;   &lt;title lang="en"&gt;     Harry Potter   &lt;/title&gt;   &lt;author&gt;J K. Rowling&lt;/author&gt;   &lt;year&gt;2005&lt;/year&gt;   &lt;price&gt;29.99&lt;/price&gt; &lt;/book&gt; &lt;book category="WEB"&gt;   &lt;title lang="en"&gt;     Learning XML&lt;/title&gt;   &lt;author&gt;Erik T. Ray&lt;/author&gt;   &lt;year&gt;2003&lt;/year&gt;   &lt;price&gt;39.95&lt;/price&gt; &lt;/book&gt; &lt;/bookstore&gt; </pre>	<pre> &lt;bookstore name = "Amazon"&gt; &lt;book category="COOKING"&gt;   &lt;title lang="en"&gt;     Everyday Italian.   &lt;/title&gt;   &lt;author&gt;     Giada De Laurentiis   &lt;/author&gt;   &lt;year&gt;2005&lt;/year&gt;   &lt;price&gt;30.00&lt;/price&gt; &lt;/book&gt; &lt;book category="CHILDREN"&gt;   &lt;title lang="en"&gt;     Harry Potter   &lt;/title&gt;   &lt;author&gt;J K. Rowling&lt;/author&gt;   &lt;year&gt;2005&lt;/year&gt;   &lt;price&gt;29.99&lt;/price&gt; &lt;/book&gt; &lt;/bookstore&gt; </pre>
--	--

Figure 2.2: Bookstore elements with different number of books

be expecting  $\langle book \rangle$  elements. However, every bookstore could have its own number of books; and unlike records in relational database, bookstore elements do not necessarily have exactly the same structure. XML documents could also have nested and recursive structures; this is why despite conforming to a predefined schema they are not fully structured and are called semi-structured documents.

Besides the tree structure, every XML document also contains text, mainly in leaf nodes and also as attribute values. As it is visible, every XML document has two main components: 1) Structure, 2) Text; each of which is a valuable source of data. Therefore, every information retrieval or knowledge extraction method has to take into consideration both of these sources of information in order to be precise. The following list provides a summary of reasons for XML usage and why it has become an inseparable constituent of today's information systems.

- XML is frequently used as a way of displaying dynamic data on HTML pages. HTML is also a markup language, but it is not capable of modeling and storing data since its tags and elements are predefined. Unlike HTML, we can define the structure and meaning of XML tags, use HTML only for displaying data and XML for storing dynamic data on web-pages. Stored data in XML files could be read, parsed and used by client-side script that runs on browsers to fill the empty HTML template [1].
- XML is a standard way of data sharing and exchange between applications. Data stored in XML format is plain text and platform independent. Every software system could read and process the document. Therefore, applications can send and receive necessary messages and data in this standard format [15].

- Since XML is an easy way of data sharing and exchange, there has been a great interest in using XML databases and storing data in XML format. Stored database could be queried using XML query languages and the result is an XML document containing the desired data. Retrieved data could be then used locally or exchanged between applications. Almost all of the modern database systems have added support for XML data type. Some XML databases are native, and work completely independent from relational databases. They have their own way of storing and querying XML documents. Some other XML databases are non-native, which means they store XML data in relational databases and query the document using relational engine, then convert the result relation back into XML format [15].

## 2.2 Classification and Feature Reduction

### 2.2.1 Classification

Every data object stored in a database is identified by its attributes either numerical or nominal. The main idea of classification is to explore through existing data objects called training set to discover a set of classifier rules which determine the class of each object according to its attributes [6]. These rules form a classifier model and are used to predict the class or missing attribute value of unseen objects whose class might not be known.

As we can see from the above definition, classification consists of two main steps. In the first step, which is also known as the supervised learning process (i.e., the class labels of the training samples are provided), the aim is to build a model that captures



the characteristics of data classes and attributes of the training set by generating a set of rules that could also be in the form of mathematical or machine learning models. The second step is to verify the accuracy of the created model through testing. The model is run over a test set to predict the classes of objects or data whose classes are not provided. By comparing the actual labels against the ones predicted by the classifier model, we can measure the accuracy of the model on the test set. If the accuracy is satisfactory, the model could be used in practice for the prediction of class labels of unseen objects whose labels are not known.

A more formal definition of the classification problem may be stated as follows [5]. Let  $D$  be a database of records  $\{R_1, R_2, \dots, R_n\}$  and  $C = \{C_1, C_2, \dots, C_n\}$  be the set of all class labels. The classification problem is a mapping  $f : D \rightarrow C$ , where each tuple  $R_i$  in  $D$  belongs to one and only one of  $m$  classes. There are some rare problems in which a record belongs to more than one class, but let us concentrate on the classical problem with single label data objects. A class,  $C_j$  contains precisely those records mapped to  $C_j$ , i.e.,  $C_j = \{R_i | f(R_i) = C_j, 1 \leq i \leq n, \wedge R_i \in D\}$ . This mapping function can then be used later to predict the classes of the tuples in a testing set.

Classification has several different applications, such as preventing theft and saving lives, medical diagnosis, increasing revenue and market analysis, better decision making, predicting customer behavior, etc. Fraud-detection is an example of a real world application of classification. The training set includes complete records of both valid and fraudulent activities, each classified as either valid or fraudulent activity. Data could be obtained from the previous cases happened in the past whose condition is known and certain. Appropriate attributes for each record are identified by domain experts and values are assigned. The classifier training algorithm uses these

pre-classified examples to generate a set of rules and parameters required for building a classifier model. Once the classifier is created, it is applied to a test set of classified samples to determine the quality of the model. The quality and efficiency of the model is usually evaluated by a combination of statistical techniques and domain expertise. When the quality of the classifier is approved, it is used to discriminate between the validity and fraudulent of future data.

There are many popular and powerful techniques for classification [6]. Decision trees [7] and Bayesian networks [8] have been widely researched for classification. Statistical models, such as nearest neighbor methods, have been used for classification as well. Many other machine learning techniques, such as neural networks and support vector machine, are used to automate the process of rule induction for building a classifier model [8].

### **2.2.2 Feature Reduction**

Feature reduction refers to the process of selecting a subset of features (attributes) of the dataset; the target is to locate the minimum set of the most significant features. As data mining has become more popular and widely used for solving problems characterized by high dimensionality, such as text processing of internet documents and gene expression array analysis, feature extraction has become one the key tasks in data mining. The main target is to reduce the dimensionality of these kind of problems to the best manageable size possible without sacrificing the accuracy; actually, the accuracy is mostly improved. A corpus of documents usually consists of at least tens of thousands of words each of which is one feature in the bag of words representation. Gene expression datasets are also quite high dimensional with

thousands of genes that identify a disease [17].

Three main reasons could be enumerated for feature reduction:

- Improving the prediction accuracy of predictor models; When there are tens of thousands of features, it is expected to have redundant and uninformative or noisy variables as well. A good example of this could be gene expression array data, where a few features (usually less than 20) out of the whole feature set are enough to build a classifier; and the classifier built based on these few features usually outperforms the one created based on all features [18]. Feature reduction has also been proven effective on text data. Since the whole word-set is extremely large compared to the number of words that appear in one document, there will be lots of 0 values in each feature vector and it will make the dataset sparse and noisy. A number of different studies have shown how feature reduction techniques, specifically feature clustering, improves the accuracy of text classifiers, e.g., [47, 48, 49, 54].
- Feature reduction could help us create a faster and less costly classifier. Obviously, when the number of variables involved in building the classifier and testing the model is smaller, the complexity decreases and it is easier and faster to build the model. However, we must be careful not to sacrifice the classification accuracy for time complexity if the accuracy is of greater importance.
- Identifying the most effective features for classification. When there are several thousand variables, we want to see which of them are most important in identifying the class label. Especially in the case of gene expression array data, we want to find the deciding genes on a disease [19, 20].

Feature extraction techniques from datasets have been focused on four main categories [17].

1. Filter methods
2. Wrapper methods
3. Embedded methods
4. Feature clustering

Filter methods refer to the methods where we don't build any classifier in order to measure the effectiveness of the feature. Instead, a measure is defined for the goodness of the feature. For this purpose, statistical measures and Information Gain have been used in combination with a threshold based on which we decide to select the feature or filter it out. Wrapper methods take another approach and build classifier models based on individual features as well as the combination of features. Then, the rise or fall of the accuracy on the training dataset is used as the deciding factor for keeping or removing the feature from the set of selected features. Embedded methods take advantage of both techniques. They usually filter the large set of features using a filter method, then the wrapper method is applied to the smaller set of features to further remove the irrelevant features. Feature clustering has also been quite popular specifically in feature selection and extraction from text documents. In this approach, first we cluster the features (words in text documents) and all of the words that are similar to each other will fall under the same cluster. Then few features from the cluster which are closest to the cluster centroid are selected. In our case, we use clustering of words to compress the feature space by directing all

of the words that are similar to each other into the same clusters and then use each cluster as a feature; all of the words in that cluster will be used in order to avoid 0 values.

The clustering problem is generally the process of exploring through data objects we have in a dataset and grouping similar objects in the same categories, while the number of categories is either not known or is provided as an input parameter. It has been widely studied in the literature [16], and several techniques have been proposed. Clustering techniques could be categorized into different categories regarding the similarity measure they define for data objects. From another viewpoint, we could have either soft or hard clustering result.

In soft clustering, every data object can belong to more than one cluster with different membership degrees; however the sum of all membership degrees must be 1 for each object. In hard clustering every object is assigned only to one cluster. Unlike classification, clustering is an unsupervised task in nature. Which means the categories are not known and there is no training and learning. However, there are cases where we have extra information about clusters or data objects besides their numerical vector as well, or we know a few points that belong to certain clusters. In those cases, using the extra information available could facilitate the process and improve the reliability of clustering. The class of clustering algorithms that use these extra information are called semi-supervised or supervised clustering techniques.

### 2.3 Content-based XML Classification

In this approach, only the content of XML documents are used in building a classifier model; document structures are completely ignored, *i.e.*, all tags within the XML documents are removed in a pre-processing step. After removing tags, the XML documents are treated as a set of labeled pure text documents and the XML classification problem is simplified to the classical problem of automated text classification.

Many standard information retrieval (IR) and machine learning methods have been successfully applied to the text classification problem. For instance, Rocchio's algorithm [10] is a classical IR method that has been adopted [11] and evaluated for text classification. In Rocchio's algorithm [10], each class is represented by a prototype vector, and a document vector is assigned to the closest class by measuring the similarity between a particular document and each of the prototype vectors. For example, the similarity can be simply computed by the dot product or by using different similarity measures. From the family of machine learning algorithms, Naive Bayes, k-nearest Neighbor, Decision Trees and Support Vector Machines have been widely applied to real world text classification problems.

Machine learning algorithms generally apply an inductive process in which an automated text classifier is built by learning, from a set of labeled documents, the characteristics of different categories. The final classifier model contains a set of decision rules that are obtained via explicit error minimization. The advantages of the above approaches are: 1) accuracy, which is comparable to that achieved by human experts, and 2) efficiency in terms of time spent for training and prediction. However, in the context of web documents classification, where the number of cat-

egories is significantly large, the above methods do not perform properly as they are all flat classifiers, *i.e.*, they ignore the structural relationship between the categories (classes) by flattening the class structure. To cope with this problem, a few hierarchical classification methods have been proposed to assign a document to an appropriate category from a hierarchical category space, *e.g.*, [12, 13, 23, 24, 25].

In most of the hierarchical classification methods, the category space is organized in tree structures [26]. For instance, Fuhr *et al.* have proposed a hierarchical content-based classification method for XML documents classification. In their work, to assign a new XML document to an existing category, the classifier works in a top-down manner starting from the root of a hierarchical taxonomy. A binary classifier, which is built for each category in the tree, decides whether a document fits in that particular category or not. A document can be classified into different categories with different confidence measures.

The most successful previous approaches proposed for text categorization have applied feature reduction techniques, specifically feature clustering, to the bag of words representation of documents in order to reduce the dimensionality. Word clustering has been a popular technique for language modeling and word co-occurrence [46] as well as feature reduction in text classification [47, 48, 49, 54]. Most of the previous work published around this topic has been concentrated on using distributional clustering of words and Information Bottleneck, presented first in [55].

The authors of [55] represent every word by its distribution of context where it has appeared. Words are divided into two groups of nouns and verbs, and the co-occurrence frequencies are measured. Relative entropy of word distributions is then used as a dissimilarity measure for clustering. Later on, the idea of distributional

clustering and Information Bottleneck has been very popular and has been used as one of the major feature reduction techniques for text classification. Authors of [47] use the class label information and represent every word by its distribution over the set of class labels. Thus, their clustering is more aggressive and they achieve a higher compression of feature space compared to unsupervised distributional clustering. In their experimental results, they achieve close but lower classification accuracies using a Naive Bayes classifier after word clustering as opposed to bag of words representation of documents. The work described in [49] takes a similar approach and shows that the classification accuracy could also be improved as a result of word clustering provided that training samples are small.

The work described in [48] combines distributional clustering and classification power of SVM for the multi-class document classification problem, and achieves higher classification accuracies in many cases compared to Naive Bayes classifiers. The idea of distributional clustering of words has emerged from natural language processing, and the aim is to provide a possible way of clustering when the only information we have is co-occurrence of words. In other words, distributional clustering is a powerful soft clustering technique when we have lack of useful information, and works best for problems where word representations are highly complicated and dimensionality is so high that using conventional clustering techniques in data mining is not accurate. In contrast, if word representation is simple and low dimensional, using conventional distance-based algorithms is preferable. We believe that the word representation after projection of words on the set of class labels is greatly informative and rather simple because the number of class labels in typical text classification problems is small (for instance, 20 in the 20 newsgroups problem [52]). Therefore, we



consider Fuzzy FCM [21] to be a more suitable soft clustering technique compared to distributional clustering for feature reduction in XML document classification problem for variety of reasons. First of all, fuzzy systems have demonstrated their effectiveness in soft computing and data mining [53], specifically in soft clustering of complex data such as Microarray and intrusion detection data [50, 51, 22]. It has also been successfully applied to mining web access data, which has a similar nature in some ways to word clustering. Secondly, fuzzy FCM has low computational cost, and when it comes to implementing real life text classification systems, computational cost becomes a serious issue. In the end, as mentioned above, word clustering problem for the purpose of document classification is a much simpler problem than the general case of unsupervised word clustering studied in natural language processing where we have little useful information available.

All of the described text classification approaches, despite their good performance in many text applications, do not perform quite effectively for XML document classification because there is a considerable amount of information buried in the structures of XML documents; such information is ignored by content-based text classification algorithms [28]. A XML document is a semi-structured data that contains not just text content, but also structures. Further, structures in XML documents are believed to be chosen more carefully than the content [29]. Consequently, structures represent more precise characteristic of XML documents that can be employed in the classification process.

## 2.4 Structure-based XML Classification

The second alternative for XML document classification is to use the information hidden within the documents' structure for building the automated classifier model. In this case, the tags and their hierarchical structure within the XML documents are learned by the classifier algorithm, and are used to assign a new incoming XML document to an existing category.

Recently, there have been several attempts to propose structure-based classifiers that solely use the structure of the document. For instance, XRules [28] is a well-known structured-based XML classification method, which works based on the basic idea of associative classifiers. Associative classification algorithms usually adapt an association rule discovery technique to extract class association rules (rules whose consequents are class labels) from a data set; this is the training process. The extracted rules should have frequencies above user specified constraints, *i.e.*, *minsup* and *minconf* thresholds. Then, a classifier is built by using a subset of the most discriminative and high quality rules generated in the learning stage. CBA [30], CMAR [31] and CPAR [32] are well-know methods from the family of associative classifiers.

In XRules [28], the developers suggested that associative classifiers can be adapted to generate structure-based rules by flattening the XML structure, and later these rules can be used to classify the XML documents. The developers also argue that if associative classifiers are to be directly applied to the XML document classification problem, they result in loss of structure information as the rule miner component of the utilized classifiers do not take into consideration the hierarchical structure of

XML documents while extracting the rules. However, the advantage of XRules over the classical associative classifier is that it has an efficient rule mining component, namely XMiner, that extracts all structural rules related to any class according to the hierarchical structure existing within the XML documents. The qualified rules, *i.e.*, rules that satisfy the minimum support and confidence values, are used later to perform the structural classification task. While XRules classifier shows improvement in accuracy compared to content-based classification methods, it suffers from several issues. First, the rule generator algorithm returns a huge number of rules, and it is difficult to store the rules, retrieve the related rules, and sort the rules. Second, in most cases, XRules achieves a high classification accuracy by using a significantly large number of rules in the classifier (by choosing very small support threshold value such as 0.03%), which in turn can cause the overfitting problem, especially for a small training data set. Further, the XRules method completely ignores the contents of XML documents. Therefore, there is still the possibility to improve the performance of the classifier model by learning from information presented by both content and structure of the XML documents.

In addition to the above approaches, some recently proposed alternative methods make use of both document content and structure in order to improve the XML document classification efficiency. The work described in [33] presents an inductive learning system for XML documents. A higher-order logic formalism suitable for representing individuals with complex structures is used for structural information representation. Structure features are constructed by composing transformations of XML document structure. A structured feature selection method, which works based on analyzing DTD, is applied to remove irrelevant text content from document ele-

ments. A new text document corresponding to every element is formed by collecting text contents of that particular element. Finally, every word in the training set is represented as a content feature. The learning algorithm of this method is a decision-tree learning algorithm driven by precision and recall. The main disadvantage of this approach is that it highly depends on the XML document schema, e.g., DTD, and can not classify schema-less XML documents where the data does not conform to any fixed schema.

Theobald *et al.* [29] proposed a classification method for schema-less XML documents. In their model, structure features are constructed according to XML paths for the document elements to represent the document structure. For content features, instead of using words as features, they combine tags with text terms that appear in the corresponding element content, and each tag-term pair represents a content feature. Finally, they use SVM algorithm to build their classifier model. The main advantage of their approach is that they map content features into an ontological concept space and construct expanded textual features. This results in better handling of documents that include heterogeneous vocabulary. Although showing accuracy improvement, this approach suffers from efficiency issues when tag-path features are constructed by considering any length within the XML document. This may generate a huge number of structure features, which in turn can degrade the performance of the classifier algorithm. In their paper, the authors limit tag-path features to path length 2 to cope with efficiency matters. However, ignoring the tag-path features with length more than 2 will result in loss of structural information.

## 2.5 Our Framework in Comparison with Previous Work

The above analysis and identified drawbacks and weaknesses of the existing approaches motivated the development of our approach described in this thesis.

The main contribution of this study is an automatic text and XML document classifier model that incorporates both structural and content information of XML documents into the process of learning and prediction. In order to extract the structural information, we use the existing rule mining algorithms to capture the frequent structural patterns in form of rules and later convert them to structural features. However, for extracting the content information of XML documents, we propose a new method based on *Fuzzy C-Means Clustering* [21] of words and using each cluster as a textual feature. We show that the classifier built only using our textual features outperforms most well-known IR-based document classification technique. Further, the combination of structural and textual features will result in an accurate and robust classifier. We demonstrate the efficiency and effectiveness of incorporating both structural and content information by performing a comparative analysis of our classifier model and several XML and text document classifiers.

## Chapter 3

# Feature Extraction from Unstructured and Semi-Structured Documents

In this chapter, we describe our techniques for feature extraction. We discuss the feature vector construction process, rule mining, document preprocessing, clustering and classification. If the approach is to be used for semi-structured document classification, both components have to be applied to capture features from both aspects of the document. Otherwise, if the purpose is only to classify plain text documents structural feature construction phase could be skipped.

### 3.1 Feature Spaces for XML Document

In order to precisely characterize both content and structure of XML documents for the learning process of XML classification, every XML document in our model is represented by a feature vector. Features in a feature vector are usually representatives of data item attributes in the form  $f_1 = v_1 \wedge f_2 = v_2 \wedge f_3 = v_3 \wedge \dots \wedge f_n = v_n$ , where  $f_i$  is a feature (attribute) and  $v_i$  is its value.

In a learning system, such as the one proposed in this thesis, a feature vector is used to describe the key properties of XML documents in the training set to be learned by the classification algorithm. Here, the feature set designed for XML classification includes structural features and textual features. The former feature set represents frequent structural patterns, and the latter feature set describes the

content information in XML documents. Structural features are constructed based on frequent structural patterns hidden in XML documents. Frequent structural patterns can be captured in the form of structural rules by adapting a rule mining algorithm.

In the context of XML documents, a rule is an entity that relates a frequent structure on the left hand side to a class variable on the right [28]. The rule mining algorithm extracts all structural rules having support and confidence greater than a pre-determined threshold; support is the percentage of documents that contain the structure in the right hand side of the rule, and confidence is the percentage of documents belonging to the class variable, *i.e.*, the consequent of the rule, by considering only documents that contain the antecedent of the rule. Such rules are able to extract useful associations between frequent structures in XML documents and class variables. Structural rules are then incorporated into the learning process in the form of structural features, *i.e.*, every rule is represented by a feature. The corresponding feature value indicates whether the rule is covered by a particular XML document from the training set or not. Structural features provide the classification algorithm with extra discrimination knowledge obtained from the structural information available inside the document, which are completely ignored by content-based classifiers. They also help the classifier to deal with XML documents that come from a large number of heterogeneous sources with different structures. Structural feature construction is covered in more detail in Section 3.2.

On the other hand, textual features are constructed based on XML document content. In context of content, every XML document is a set of words, each of which has a frequency of occurrence in the document. The straight forward way to

extract textual features would be by considering every word as a feature and the corresponding word frequency as the feature value. This representation results in a unique feature vector for every document and every feature value is set to the normalized frequency of the word. Although this way of feature construction is easy and requires few computational steps, it has the following major drawbacks. First, when every single word is modeled as a feature, obviously the resulting feature vector will have many dimensions because the number of existing words in a set of documents is very large. High dimensionality increases the complexity of the problem and reduces the efficiency and accuracy of the classifier model. Second, when new words are added to the system as a result of adding new documents to the system, this change will greatly affect the system since new words need to be added to the feature vector of all existing documents. Third, each document contains only a small portion of the words in  $\omega$ , which represents the complete set of words existing in the training documents; this means that a large portion of the feature values in the feature vector will be mostly 0. Having the final textual feature vectors so sparse will definitely overshadow the accuracy of the final classifier model. In order to cope with these issues and take a more sophisticated approach in feature extraction, we propose a new method based on *Fuzzy Clustering* of words and using each cluster as a feature. The corresponding feature (cluster) value for each document will be related to the number of words in that document that belong to the cluster. In other words, the corresponding feature value describes the dependency of the document on a cluster of words. Among all clustering techniques that could be possibly used for our purpose, *Fuzzy Clustering* is preferable since it can associate every word to multiple clusters with different membership degrees, and hence avoids crisp assignment of words to



clusters. Our proposed method for textual feature construction is described in more detail in Section 3.3.

## 3.2 Structural Feature Construction

As described in Section 3.1, structural features are constructed based on frequent structural patterns to characterize structural information of XML documents. Structural feature construction includes two major phases: generating structural rules and building structural feature vectors based on the generated rules. In the first phase, the task is to extract from the given XML documents all structural rules that satisfy user-specified support and confidence thresholds. In the second phase, the validity of rules for every XML document is used to generate a set of structural feature vectors. Every XML document in the dataset is represented by a feature vector. Every feature in the vector corresponds to an individual rule. The value of every individual feature for any particular document is set to 1 if the rule is covered by that particular document, otherwise the value is set to 0. The generated feature vectors represent coverage distribution of structural rules over the given XML documents.

### 3.2.1 Structural Rule Mining

A structural rule is formally defined as  $X \xrightarrow{s,c} Class_i$ , where  $X$  is a frequent substructure in a given collection of XML document structures  $D$ ,  $Class_i$  is one of  $k$  classes,  $s$  and  $c$  are support and confidence measures, respectively; they are used to evaluate

rule goodness and are computed as follows:

$$s(X \Rightarrow Class_i) = \frac{s(X \cup Class_i)}{|D|} \quad (3.1)$$

$$c(X \Rightarrow Class_i) = \frac{s(X \cup Class_i)}{s(X)} \quad (3.2)$$

The rule  $X \stackrel{s;c}{\Rightarrow} Class_i$  implies that a particular XML document  $j$  with structure  $D_j$  more likely belongs to  $Class_i$  if  $D_j$  contains frequent substructure  $X$ . The problem of structural rule mining involves extracting all structural rules that satisfy the user-defined thresholds for support and confidence. It is basically divided into two subproblems. The first subproblem aims at finding frequent substructures from the given data collection  $D$ . Frequent structures refer to substructures whose support exceed the support threshold value (minimum support). The second subproblem is to generate structural rules from frequent structures by considering the fact that rules should satisfy the minimum confidence threshold.

Most rule mining algorithms require the input data to be in transactional form in order to perform the mining process. The issue in structural rule mining for XML classification arises mainly because of the semi-structural nature of XML documents that contain both tags and content. The general approach used to address this issue is document format transformation as a preprocessing step. First, all document content is removed. Second, the tag structure of the XML document is transformed into a tree structure to preserve the hierarchical form of the document. Finally, the tree structures, each of which represents a XML document, are flatten to form a transactional dataset appropriate for rule mining algorithms.

The method we have used for structural rule mining follows the rule generator approach, called XMiner [28]. In XMiner, the TreeMiner algorithm [34] is adapted to

find all structural rules that have support and confidence greater than given thresholds. In order to apply XMiner, XML documents are modeled as ordered, labeled, and rooted trees. Later, tree structures are converted into transactions. It is worth mentioning that this conversion preserves the hierarchical structure of the trees. For further information about XMiner, the reader should refer to [28].

### 3.2.2 Constructing Structural Feature Vectors

In order to make use of the structural information discovered by structural rules in the learning process, every structural rule is modeled by a structural feature and every XML document from the training set is represented by a structural feature vector.

A structured feature in our model is defined as a predicator indicating whether every individual rule from the extracted structural rule set is covered by a particular XML document from the training set. Feature vectors constructed using our method describe the distribution validity of structural rules over the training set. The process of building the final structural feature vectors is as follows. We check the validity of rules against XML documents. If a rule is valid for a document (*i.e.*, the rule is covered by the document), the value of the feature representing that rule in the feature vector is set to 1; otherwise the value is assigned to 0. A pair <feature, value> in a structural feature vector takes the following form:

$$f^{r_i} = \begin{cases} 1 & \text{if } r_i \text{ is covered by the document } d \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

where  $f^{r_i}$  is a feature that represents rule  $i$  in the structural rule set and  $d$  is a XML

document from the training set  $D$ .

In order to maintain consistency between learning and prediction, we need to generate structural feature vector for the given test document. However, there is a difference in the process of generating structural feature vector for a test instance compared to the same process when applied to a training sample. In case of a training sample, we can check to see if a structural rule is covered by the sample as we already know its class label. However, for a given test instance, the class label is not known. As a result, it is not possible to check the validity of a particular rule for the given test instance using the same strategy that we apply for training samples. To address this issue, we make a minor change in the algorithm as follows. If the antecedent of a structural rule is covered by a test instance, we assume that the rule is valid for the given test item and set the value of the feature representing that rule in the corresponding feature vector to 1; otherwise, it will be set to 0. Using this approach, every test instance can be represented by a structural feature vector as well. At this stage, we can apply the classifier model to predict the class label of a given test instance by taking into consideration consistency of the model.

### 3.3 Textual Feature Construction

Although there are different types of documents which are used for different purposes, all these documents have one main element in common which is text as the major content. In this sense, every document, including XML documents, can be looked at as a bag of words and the text is definitely one of the main sources where we can extract valuable content information from documents, in the form of textual

features.

This section is devoted to describe our approach for extracting textual features from XML documents. There are different approaches for extracting features from text. The simplest representation of text documents is to use every existing word in the dictionary as a feature and frequency of the word in documents as feature values. It is also possible to apply feature reduction techniques, such as information gain based feature reduction, on the set of words and restrict the feature space to a subset of words. Another possible way of feature extraction or reduction as mentioned in Chapter 1 is clustering of words.

One of the major contributions this work is extracting textual features from text documents through soft clustering of words. We will explain in detail how it is possible to cluster the words in our approach and also experimentally show how clustering of words could improve the accuracy of the classifier model compared to the other feature extraction and reduction methods used for document representation.

To form textual feature vectors, four main steps are involved in the model as follows.

1. Document preprocessing (if necessary);
2. Building an Inverted index (if it is not already available);
3. Clustering of words;
4. Building the feature vector for every document.

Here, it is worth mentioning that since most IR systems use inverted index and preprocess the documents before building the index, the first two steps are not nec-

essary to be done in most cases. However, in order to preserve the integrity of this document as a complete thesis, we describe the first two preliminary steps as well as the last two steps which are the main components of our system in constructing textual features.

### 3.3.1 Document Preprocessing

In preparing the documents for textual feature construction, three main pre-processing steps are required. First, all non-textual components of a document need to be removed to form a plain text document. For instance, in the context of XML documents, we need to get rid of all XML tags. After converting all documents into plain text format, we perform the two well-known preprocessing steps applied in every IR system, known as *Stop Word Removal* and *Stemming* [12].

*Stop word removal* refers to the process of removing all common words that can appear in every document regardless of the context. After removing all stop words from the document, stemming is performed.

*Stemming* refers to the process of converting every word into its root in the English language. For instance, there are different variations of the word “go” such as *go*, *went*, *going*, *gone*, etc. All different forms of any particular word more likely denote the same concept when they appear in a document. Therefore, in our IR feature extraction approach, we are not concerned about the tense or the form of the word; only the root meaning of the word is considered. Since there are many different variations for words in the English language, stemming is usually a tricky step.

In this work, an existing open source library has been used for the stemming task it is described in Section 5. Hereafter, wherever the word “document” appears in this chapter, it refers to a document in the plain text format.

### 3.3.2 Original Document Set vs. Inverted Index

The original representation of documents and words in a set of documents includes documents as the main entries and a set of words within every document. In order to find the occurrence of a given word  $W$  within the content of a set of documents, a linear search over all documents is needed. Every document  $D$  in the set can be represented as:

$$D = \{(w_i, f(w_i, D)) | w_i \in \omega\},$$

where  $w_i$  is a distinct word in the complete word set  $\omega$ ;  $f(w_i, D)$  denotes the normalized frequency of  $w_i$  in  $D$ , such that the sum of all word frequencies for every document is 1. In other words,  $f(w_i, D)$  gives the fraction of words,  $w_i$ ,  $i = 1, 2, \dots, m$ , existing in document  $D$ . Obviously, the frequency of all words that appear in some documents and exist in  $\omega$  but do not appear in  $D$  is 0 for  $D$ . Using this representation, every document can be modeled as a vector of  $\langle \text{word}, \text{word frequency} \rangle$  pairs. The normalized frequency of  $w_i$  in  $D$  can be calculated by dividing the number of occurrences of  $w_i$  in  $D$  over the total number of words in  $D$ . Later in this thesis, we will refer to the matrix representation of original documents as  $\Delta_{N \times |\omega|}$ , where  $N$  is the number of documents and  $|\omega|$  is the cardinality of the complete word set, *i.e.*,  $\omega$ . Every entry  $\Delta_{ij}$  represents the normalized frequency of the  $j^{\text{th}}$  word from  $\omega$  in the  $i^{\text{th}}$  document.

In addition to the document representation described above, the inverted index technique has been used to represent a given document set as well. An entry  $W$  in the inverted index could be represented as:

$$W = \{(d_i, f(W, d_i)) | W \text{ appears in } d_i\}.$$

Here, every index entry for a particular word  $W$  includes a list of all documents that contain the particular word, and  $f(W, d_i)$  denotes the normalized fraction of word  $W$  in document  $d_i$ .

Inverted index structure indicates the same concept in different IR systems in the way that it indexes documents by words. Different systems use different measures for frequency; here, we use the described normalized frequency measure and argue that it suits our feature vector extraction task better than the other ones. We use both inverted index and original representation of documents in order to build textual feature vectors. The following explanation describes how we benefit from these two representations to form the textual feature space.

### 3.3.3 Supervised Word-Vector Formation

Every clustering technique requires a similarity measure to estimate the closeness among objects that are going to be clustered. Since we are interested in *Fuzzy Clustering* of words, we need to represent every word by a vector which is constructed by considering the *inverted index* and class labels of documents in our training set. Since we use the class label information to form word vector, our final clustering model will be *supervised* in the sense that the class label information will be involved in calculating the similarity between objects (words). In the rest of this section, we



articulate the process of building word vectors.

To perform fuzzy clustering of words, the algorithm requires every word to be presented in a vector form. For this purpose, we use class label information existing in the training set. Every word is mapped to a vector of length  $n$ , where  $n$  denotes the number of distinct class labels in the training set. There is one coordinate corresponding to each of the  $n$  class labels in the vector. In other words, every coordinate shows the importance of the word for the corresponding class of documents, and the vector is normalized such that the sum of all the values associated with the coordinates is 1.

As defined earlier in this section, every entry  $W = \{(d_i, f(W, d_i)) | W \text{ appears in } d_i\}$  in the *inverted index* contains all documents in which the word  $W$  has appeared as well as their corresponding frequencies. Further, the class label for every document is also available along with that document. Thus, the inverted index structure provides us with all required information about the given document set. Equation 3.4 shows how the unnormalized value of each coordinate is computed.

Let  $w_i = \{(d_k, f(w_i, d_k)) | w_i \text{ appears in } d_k\}$  be the inverted index for word  $w_i$  and  $V_i = (v_1, v_2, \dots, v_n)$ , be the corresponding word vector, where  $n$  is the number of classes; the coordinate value is computed as follows:

$$V_{ij} = \frac{\sum \{f(w_i, d_k) | d_k \in C_j\}}{\sum f_{ik}} \times \frac{\sum_{\ell=1}^n |C_\ell|}{|C_j|} \quad (3.4)$$

The value of the  $j^{\text{th}}$  coordinate is calculated as the frequencies of  $w_i$  in all documents from class  $C_j$  in which  $w_i$  appears, over the sum of all frequencies existing in the *inverted index* entry for  $w_i$ . This fraction shows the dependency of class  $C_j$  on word  $w_i$ . The only drawback of using this fraction is that in many datasets, instances

are not distributed uniformly among the classes. In such datasets, the dependency of dominant classes to the word will be overestimated. To neutralize this effect, we multiply each coordinate value by the number of all documents over the number of documents in that specific class; this will push down the coordinate value for all dominant classes and will boost up the coordinate values for all classes with fewer instances. Therefore, the value of each coordinate will get closer to the dependency of the corresponding class to word  $w_i$ , regardless of the distribution of instances in classes. Every coordinate of the vector is then normalized using Equation 3.5.

$$VN_{ik} = \frac{V_{ik}}{\sum_{j=1}^n V_{ij}}, n \text{ is the number of coordinates} \quad (3.5)$$

Normalized vector  $VN_i$  is the final vector that will represent word  $w_i$  in the next *Fuzzy Clustering* step.

### 3.3.4 Clustering of words

It was mentioned earlier how clustering of words could be useful to compress the feature space. Here, we review and categorize the possible approaches to word clustering from different perspectives. Clustering approaches we consider for our application could be categorized into two categories: soft and hard clustering from one perspective. From another point of view, we could choose an unsupervised, semi-supervised or a supervised clustering approach. we believe unsupervised clustering of words could be helpful for applications where the clustering of words is done for purposes other than classification. But for the classification purpose, we don't con-

sider unsupervised clustering of words an option because it ignores useful class label information and focus on semi-supervised and supervised clustering of words.

In general, our approach suggests semi-supervised clustering using the distributions of words over the class labels as vectors to be clustered and soft clustering. It is worth mentioning that soft clustering algorithms are unsupervised in nature. What makes our clustering of words semi-supervised is the combination of supervised word vector formation with regard to distribution of words over class labels and an unsupervised clustering algorithm.

Word vector formation itself could also be considered as a way of fully supervised clustering of words. Every class label will represent one cluster and the number of clusters is the same as the number of class labels, which will also solve the problem of parameter setting for us. On the other hand, when the number of class labels is rather large, like many real word applications and as in many benchmark datasets (Reuters with 188 classes), the quality of clustering usually goes down. Therefore, when the number of document categories is large, we would rather use the distribution of words over the categories as a matrix representing the soft clustering of words. This will solve the problem of parameter setting as well as the problem of dealing with low quality clusters as a result of high dimensionality.

Clustering algorithms could also be categorized into soft and hard. We already mentioned that the use of soft clustering is preferable. However, we will experimentally compare soft and hard clustering algorithms for feature extraction and show how soft clustering could be more effective for our application.

### 3.3.5 Fuzzy Clustering of words and Constructing the Final Feature Vector Set

Given the number of clusters  $K$ , *Fuzzy C-Means* returns for every object a vector that shows the membership degree of the object in different clusters. For a given set of word-vectors that has the cardinality of  $|\omega|$ , *Fuzzy C-Means* returns a  $|\omega| \times K$  matrix that we call *Words Membership Matrix*, denoted  $WMM$ . Every row  $i$  of  $WMM$  represents membership degrees of a single object, simply a word, in different clusters and every column  $j$  represents membership degrees of different objects in the  $j^{th}$  cluster.

The process of constructing the final textual feature vectors, after mapping every word to a vector, can be summarized in the following main steps:

1. Fuzzy clustering of words and forming the Words Membership Matrix  $WMM$ ;
2. Matrix Multiplication of  $\Delta$  and  $WMM$  (the original set of document vectors);
3. Normalizing the multiplication result.

The first step could be skipped if the number of categories is large, and we decide to use the distribution of words in classes as soft clustering of words. Equation 3.6 shows how the final textual feature vector set is formed after matrix multiplication; recall that  $\Delta$  is the matrix representation of the original documents.

$$FVSet_{N \times K} = \Delta_{N \times |\omega|} \times WMM_{|\omega| \times K} \quad (3.6)$$

Each row in  $FVSet_{N \times K}$  represents feature values for a particular document from the document set. Every document's corresponding vector is normalized such that

the sum of all feature values in the vector is 1. The number of features  $K$  can be set to any desired number; it is equal to the number of clusters. We adjust this parameter experimentally and show that we can capture enough textual information as a result of using a supervised word-vector formation and fuzzy clustering of words, even using a very small number of features compared to the actual number of words in each of our benchmark datasets (approximately 14,000). The achieved accuracy is comparable to the most successful previous work described in the literature. We use  $K = 10$  in all of our experiments. This parameter setting is discussed in more detail in Chapter 5.

Finally, feature vector construction is performed in the same way for both training and test samples. The only difference could be stated as follows. Although the range of words used in a specific domain is limited, there could be words that exist in the training document set, but do not appear in the new test document. For every new test document, first the preprocessing step is performed to make the words consistent with the set of words in  $\omega$ . Document representation vector for every new test document  $t$  will have  $f(t, w_i)$  for every word  $w_i$  that exists in  $\omega$  and 0 for all other coordinates that represent a word that exist in  $\omega$ , but not in  $t$ . All other words that might be in  $t$  and have not appeared in any of the documents in the training set will be disregarded as we did not have access to these words while we were creating the classifier model. After converting the test document to a form consistent with the document representation in our training set, we have a  $1 \times |\omega|$  matrix and we can perform the same matrix multiplication with  $WMM_{|\omega| \times K}$  to form the feature vector for test document  $t$ . The normalized feature vector could be passed to any classification model that we build from the training feature vector set for class

prediction.

### 3.4 Building Classifier Model

In order to incorporate both structural and textual features into the learning process, structural and textual feature vectors associated with any particular XML document are first consolidated to one feature vector. The resulted feature vector indicates both structural and content characteristics of the XML document. The task of building a classifier model in our approach is then to employ the combined feature vectors, each of which represents a XML document from the training set, into an effective learning algorithm which uses these feature vectors to train a classifier model. There exist many classification algorithms for supervised learning. The performance of different learning algorithms strongly depend on the classification domain and the dataset. In this study, we have used support vector machines and decision tree algorithms to build the classifier model. To evaluate the accuracy of our classifier model, we use the train versus test validation technique.

## Chapter 4

# Creating a Hierarchy of Subjects Using Support Vector Machines and 10 Fold Cross Validation

In this chapter, we describe how it is possible to exploit feature based representation of documents along with SVM in order to create a hierarchy of subjects. While text classification is necessary to automatically assign new documents to appropriate subject(s), creating a hierarchy of documents helps users to query and browse the document set more efficiently.

Most of the existing hierarchies are created manually based on the conceptual similarities between subject categories. Manual hierarchy creation is subjective and users from different domains might end up with different hierarchies of categories. Although manually created hierarchies are subjective, we believe they can still provide a good framework for user queries if they are created by domain experts in every sub-domain. However, this way of clustering subjects might not be possible for modern applications and document sets since the number of documents and subjects are growing exponentially every day. There are also situations in which we have no good regarding idea how the hierarchy should be built based on the conceptual similarities between subjects. Automatically generated hierarchies have been proposed in order to avoid the confusion and provide a unique hierarchy of documents that could be cut at any level to get top level document groups. The process involves scanning the whole document set in which every document is represented as a bag of words and

measuring the similarity between different categories.

To avoid the manual construction, we propose an automatic solution for hierarchy creation from the training dataset using the feature vector representation of documents described in Chapter 3, and also Support Vector Machines. We repeat the experiments involved in hierarchy creation 10 times through 10 fold cross validation to insure the reliability of our created hierarchy. Our approach to hierarchy creation is first calculating the distance between classes. For this purpose, we use a supervised classification model that could assign every sample to two different classes with different probabilities. In other words, we could also use support vector machine as a way of soft classification that computes the degree of similarity of each document to different classes. If the model is being used for the purpose of classification, then the class which has the maximum membership probability will be the predicted class for the given document. Here our purpose from this experiment is only hierarchical clustering and grouping the large number of class labels into smaller top level categories. Automatically created hierarchy could be further studied and labeled by domain experts for future usage and for facilitating user queries. The rest of this chapter summarizes the steps involved in automatically creating a hierarchy of subjects from feature vector representation of documents.

## 4.1 Creating Classifier models for Soft Classification

The first step for clustering of class labels is to compute the distance between them. For this purpose, we use Support Vector Regression [56] that could compute the membership probability of a sample in each of two classes. Obviously, creating a



hierarchy for a dataset with two classes makes no sense. Therefore, we are always dealing with a classification problem of classifying data samples into  $|C|$  classes where  $C$  is the set of all existing categories. We construct  $|C|$  SVR's each for classifying one of the classes say  $c_i$  against the other ones. For creating SVR model  $S_i$ , all of the samples from class  $c_i$  are used with their labels changed to 1 and the rest of the dataset is sampled randomly to select  $|c_i|$  samples distributed evenly over the  $|C| - 1$  classes except  $|c_i|$ . Then, we label all of these randomly selected samples by 0.

Using each SVR,  $S_i$ , we could compute the probability of membership of every test data sample in  $c_i$ . Please notice that by testing data samples here we mean the portion of training dataset which is used for testing in 10 fold cross validation. We are not allowed to use samples from test dataset for hierarchy creation since those samples are only used for validation and measurement of accuracy. After applying all regression models on all samples, we get a  $|D| \times |C|$  matrix, where  $|D|$  is the size of the training document set. Every element  $j_i$  of the matrix shows how likely it is for the  $j$ th document to belong to class  $|c_i|$ . Let's call this matrix the membership probabilities (*MP*) matrix. The next step in hierarchical clustering would be calculating the distance between classes using the membership probabilities matrix.

## 4.2 Calculating the Distance between Subject Categories

Besides the calculated probabilities in *MP*, every sample in the training set also has a real class label. Equation 4.1 shows how the distance between two different classes  $i$  and  $j$ ,  $d(i,j)$ , is found using the membership matrix.

$$d(i, j) = 1 - \frac{\frac{\sum \{MP_{kj} | D_k.label=i\}}{|c_i|} + \frac{\sum \{MP_{ki} | D_k.label=j\}}{|c_j|}}{2}} \quad (4.1)$$

Basically, the distance or dissimilarity of classes in Equation 4.1 is defined to be *1 - similarity*. Similarity is always a number in the range [0,1] because all membership probabilities are between 0 and 1 and so is the every average value on them; it is found using membership probabilities. Every row in *MP* as mentioned before, represents memberships of the corresponding document from document set in different classes. In order to find the distance between classes  $c_i$  and  $c_j$ , first we find average membership probability of all documents that belong to  $c_i$  in class  $c_j$ . Then, we repeat the computation the other way around, and see how likely it is for documents of class  $c_j$  to belong to class  $c_i$ . Similarity of two categories is then the average of the two similarities mentioned above.

Ten fold cross validation does not make any significant difference to the process. It is only to improve the reliability of the distance measure defined. Experiments are repeated 10 times on the training set and each time one of the 10 splits of the dataset is used for testing and creating the membership matrix. Ten different distance values between classes is then found and the average of these 10 values would be used as the final distance. Having computed the distance between categories, now we can use hierarchical clustering for building a hierarchy of class labels.

### 4.3 Hierarchical Clustering of Subject Categories

Hierarchical clustering is a clustering approach that converts the flat structure of existing objects into a hierarchy according to their distances from each other. When

the distance between objects is known, regardless of the distance measure used, it is very simple to apply hierarchical clustering. The general process may be summarized as follows:

1. The distance between objects is known.
2. Define a distance measure between clusters and also between clusters and objects.
3. Consider every cluster as an object.
4. In each step of the algorithm find the shortest distance between objects and merge them into one cluster. Merged objects could be two basic objects, two clusters or a basic object and a cluster.
5. Continue combining objects until there is only one object left and that is the whole set of objects.
6. Hierarchy of objects could be obtained by logging during the execution of the algorithm and tracing the log backwards.
7. In the end, the hierarchy could be cut at any level to get the desired number of top level clusters.

This algorithm is called Hierarchical Agglomerative Clustering [57] and the output is a dendrogram that shows the steps of the algorithm on the hierarchy at different levels. There are several approaches for defining the distance between clusters and also for defining the distance between a single object and a cluster. Single-linkage measure uses the minimum distance between objects in two different clusters

as distance between two clusters. There are also other variations that use maximum and average distance. Here, we prefer to use the single-linkage measure because we believe if there are any two classes that are similar to each other, they have to be put in one cluster in order to minimize misclassification; our main purpose here from hierarchy generation is to divide classes into top level groups in such a way that improves classification accuracy.

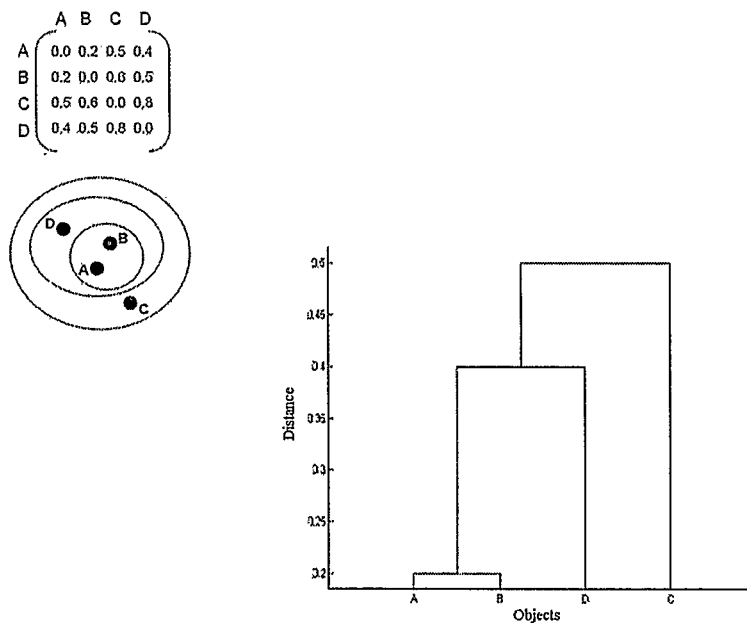


Figure 4.1: Process of Hierarchical Clustering of Objects

To illustrate how a hierarchy of objects could be created using a given distance matrix, assume we have four objects with their distances from each other as given in Figure 4.1. From the distance matrix it is simple to follow the steps of the algorithm.

First objects  $A$  and  $B$  are grouped into one cluster because 0.2 is the shortest distance between two objects. Then  $D$  joins the newly created cluster because 0.4 is the next smallest value in the matrix; and finally the whole group of objects forms the highest level of the hierarchy. The algorithm applied here followed the single-linkage measure. We might get a different dendrogram using other types of linkage measures.

## Chapter 5

### Experimental Analysis

We have implemented the whole system architecture presented in Figure 1.2. Some of the components have been adapted from existing systems developed either by our research group or by other research groups as described in Chapter 2. To pre-process documents for textual feature extraction, we used the implementation from [35] for stemming and a complete list of stop words from [36]. For structural feature extraction, we obtained the Xminer engine from the authors of [28]. Our implementation of textual feature extraction as described in Section 3.3 could also be used independently for IR classification. We have done experiments on XML documents in Log Markup Language (LOGML) [37] and have compared our results to three classification methods XRules, CBA and IRC [38]. In order to evaluate our text classification approach, we have also reported the results we achieved from running our IR component after ignoring all the structural information on XML datasets as well as the 20NewsGroups [52] dataset. Before comparing our results to other approaches, we will briefly describe our datasets. We will also suggest a possible way of parameter setting (number of features to be extracted from each aspect) for our approach.

#### 5.1 The Datasets

We have conducted extensive experiments on Text and XML datasets (*all publicly available*) to evaluate both our text classifier built only based on textual feature

extractor and also our proposed XML classification framework. The 20NewsGroups dataset is used for text classification, which is one of the most popular benchmark datasets for text mining research projects. The version of this dataset we have used is a collection of 18,828 documents distributed almost evenly among 20 different classes each of which is a newsgroup. Each document is an email message sent under one of the subject categories that also contains header information. In the version of this dataset we are using, all of the duplicates and header fields are removed except for *From* and *Subject* fields. We have further removed the *From* field from all the messages to use only subject and body of the documents for classification. Table 5.1 shows these 20 classes divided into 6 top level categories. Top level division of this dataset is only to show how it is organized and how there exist classes so similar to each other. For the purpose of training and testing our system, we have used all of the classes and the classification problem is to assign one of the 20 labels to every test document based on the model created from documents in the training set.

Table 5.1: Subject categories of 20NewsGroups dataset partitioned into 6 top level classes

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

A XML document in LOGML describes the web-server log of a user session in a specific compact format. Every document contains a graphical representation of the

log file as well as the content of the visited pages. Along with every document, there is also information about the domain from which the users have visited the page on the main Internet domain or one of its sub-domains. Our main purpose is to build a classifier capable of classifying users who have visited from an academic domain (*edu* or *ac*) against users who have visited from other domains. We have downloaded the log files from [39]; it consists of user sessions for three consecutive weeks on the web-server on *http://www.cs.rpi.edu* domain. In all our experiments, we have used the log of one week for training and the log of another week for testing. Table 5.2 describes the characteristics of our datasets. Our main reason for choosing XML datasets in this format is being consistent with the experimental framework presented in [28], which has been one of the most successful XML classification systems so far and have a basis for comparing our system to previous research in the area.

Although there are only two classes, by looking at the distribution of documents in the classes as shown in Table 5.2, it is obvious that expectedly most of the documents belong to the *other* class because it covers a wider range of internet domains. This complicates the classification process. Besides the distribution, due to the nature of the documents that contain log file information, it is quite complicated for the classifier model to distinguish between classes. This complexity originates from the fact that there are many general pages on every webserver that all users may visit. Although users from academic domain are more likely to visit scientific pages, it does not mean that they never visit pages from other categories.



Table 5.2: Characteristics of XML datasets

Dataset	# of Documents	Distribution in classes	
		educational	other
LOG1	8074	1962	6112
LOG2	7047	1686	5721
LOG3	7628	1798	5830

## 5.2 Feature Extraction from XML Datasets and Classification

Although we have described in detail how we extract textual and structural features, yet there are some other issues to address for feature extraction. In Section 3.3.5, we have already mentioned that the number of textual features is equal to the number of clusters ( $K$ ), which we determine experimentally in this section. As for the structural features, we mentioned that XRules works based on frequent pattern mining concepts, and in order to mine frequent structural patterns we need to set the support threshold. XRules achieves its accuracy by setting the latter parameter to very small values such as  $0.03\%$  when it uses XMiner, which results in extremely large rule set; in most cases more than 20,000 for the described datasets. This results in overfitting in the training dataset, and the difficulty of a meaningful interpretation from the classifier model later on. Furthermore, instead of suggesting a basis for parameter setting, authors of [28] built several rule sets by changing the support and confidence parameters and report their best achieved accuracy. Although this experimental approach based on changing parameters and reporting the best result has been widely used in the literature, it is not possible and realistic simply because we

have no idea about the parameters when we start the experiments. In this section, we describe how we have chosen our parameter values experimentally and how we can achieve high classification accuracy by extracting few features from either aspect of the XML documents before starting the final classification.

### **Number of Clusters:**

In order to set the number of clusters, we did experiments on LOG1 dataset. We split the dataset randomly into two halves for training and testing. We found experimentally that classifier models start to learn and extract knowledge for the values of  $K$  around 10. For very small values of  $K$ , say closer to 2, the classifier ends up classifying most samples in the dominant class, which is an indicator of lack of knowledge extraction. We continued adding features to the feature space by recreating the final dataset 26 times. We did this experiment for all integers between 2 and 20. For other cluster numbers greater than 20, we have increased the number of clusters by 10 in each step until 100; we used Linear-SVM to perform this experiment as SVMs have always demonstrated their reliability and strength for the classification problem, specifically text classification [14]; by running comprehensive experiments, we found Linear-SVM more accurate than others for our problem.

It can be easily observed from Figure 5.1 that very few clusters cannot capture enough textual information from the dataset for the final classifier model. As we increase the number of clusters from 2 to 10, there is considerable change in accuracy. The peak is at 9, and after 10 the accuracy remains around 84%; the classifier model seems to be stable. We increase the number of clusters by 10 from 20 to 100 to support the reliability of our choice of the number of clusters. Although doing

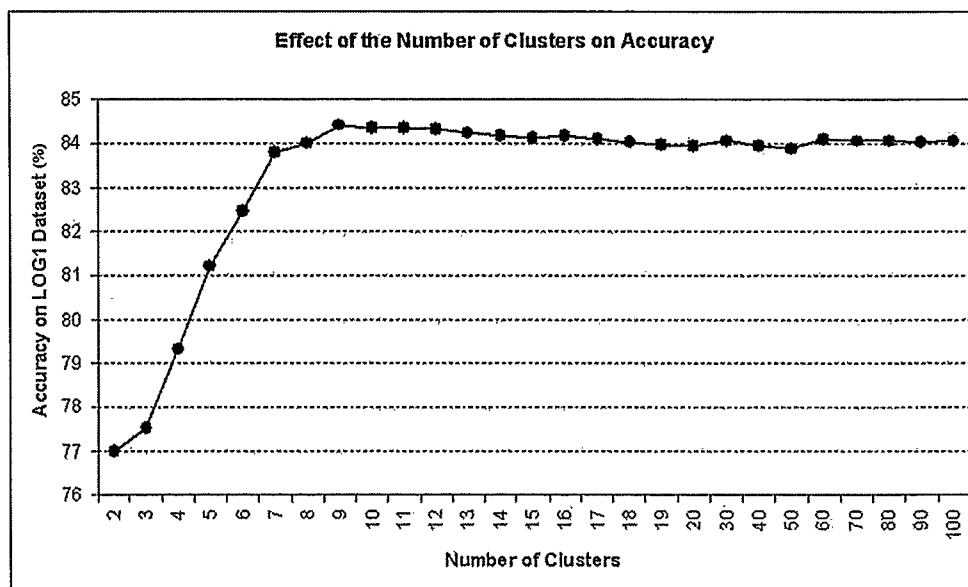


Figure 5.1: Number of Clusters vs. Accuracy on LOG1 Dataset

the same experiment on other datasets could result in having the peak at a different value, all of them become stable around 10; so, we have set  $K = 10$  for the textual feature extractor component in all experiments.

#### Selection of Frequent Structural Patterns:

Every frequent structural pattern mined by XMiner is considered a feature in the structural feature space. Similar to any other frequent pattern mining algorithm, XMiner also defines frequent patterns based on a minimum threshold value (*Support*). In order to avoid a huge structural feature space, we conducted experiments on the same split of LOG1 dataset used to set the number of clusters. We set the support value equal to  $0.7\%$  to mine the first 1,817 frequent patterns. We used Information

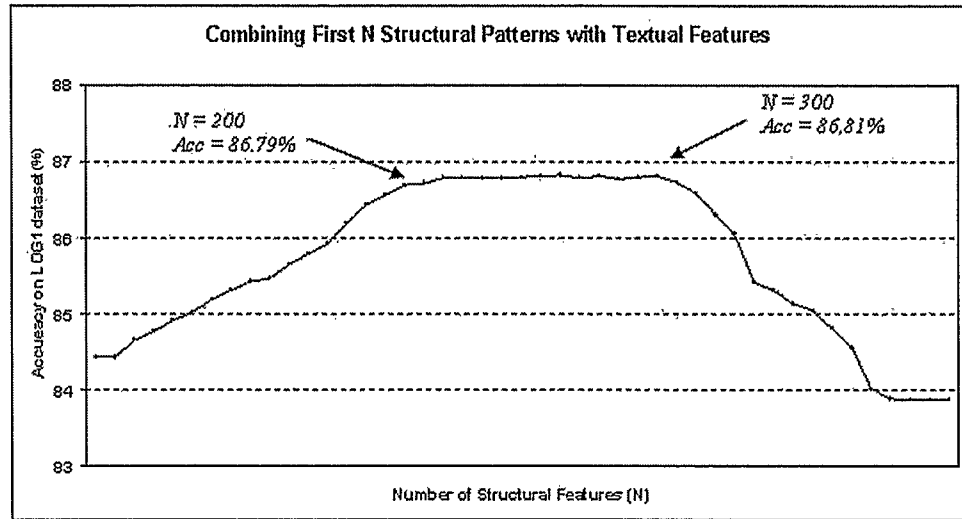


Figure 5.2: Combining Textual and Structural Features

Gain of features with respect to class labels to sort the features. In each step, we combined the first  $N$  structural features with 10 textual features derived from the previous section in order to find the best combination. Since the number of required frequent structural patterns is more due to the sparseness of these features, we know that we need to add more structural features than textual features. We start  $N$  from 10 and add the next 10 features to the feature space in each step until 300. From 300 to 1800, we increase the step size to 100. Figure 5.2 shows how combining the first  $N$  structural features with the 10 textual features discovered from the previous step changes the accuracy of Linear-SVM on LOG1 dataset.

Very few structural rules can not capture enough information from the dataset; however, they can still improve the accuracy when combined with textual features. The model gets so close to its highest accuracy (86.81%) when 200 rules are added;

there is no considerable change in accuracy as we keep adding features to the feature space in steps of 10 from 200 to 300. Figure 5.2 also shows that by adding large number of structural rules, the effect of textual features fades away gradually and the accuracy decreases. According to the feedback from the conducted experiments, we have set the number of clusters to 10 and the number of rules to 200 in all of our experiments.

Table 5.3: Classification results using the whole space and different methods

Datasets		Accuracy (%)	
training	test	Classification Method	
		DT	SVM
LOG1	LOG2	85.39	<b>85.88</b>
LOG2	LOG3	<b>87.04</b>	85.79
LOG3	LOG1	83.86	<b>84.56</b>

**Classification Accuracy and Comparisons** After fixing the number of features to be extracted from each aspect of the document, we can use the final feature based dataset to build our classifier model. As mentioned before, feature representation of documents gives us the flexibility to apply different techniques. We have explained how we benefit from information gain in the previous section. Here, we also take advantage of this flexibility and use two different techniques, Linear-SVM and Decision Trees (DT), that work based on totally different concepts to build classifier model. Table 5.3 shows how SVM and DT perform on the final feature based dataset, which is the result of concatenating feature spaces.

Extracted knowledge in the form of textual features overlaps structural knowledge in many cases, and repeating the same information is no help to improve the

classification accuracy. However, it is expected to observe improvement after using the whole feature space since it is very likely to have individual feature spaces that do not contain exactly the same information.

Table 5.4: Comparison of IR classifiers

Datasets		Accuracy (%)	
training	test	Feature-IR	IRC
LOG1	LOG2	<b>83.37</b>	74.81
LOG2	LOG3	<b>83.41</b>	77.64
LOG3	LOG1	<b>83.42</b>	73.76

We have compared the accuracy of our IR classifier to IRC as proposing a feature based IR classifier is one of the main contributions of this thesis. As summarized in Table 5.4, our IR classifier can outperform IRC by an average of 9% in accuracy. This improvement is due to the fact that we compress the whole feature space of words into clusters and use well known classification techniques; there is very little content-based information that we ignore compared to IRC, which works based on clustering of documents. Our classification accuracies are close in all of the experiments because all LOG datasets contain the same documents visited in different weeks, and the closeness of accuracies in different experiments is an indicator of successful and similar clustering of words, regardless of the corpus based on which we clustered the words.

At the end, we compare the accuracy of classification using the whole feature space to two of the most well known structure based classifiers. In two of the experiments, we have achieved our best accuracy using SVM and DT; this helped us creating a better model for the second experiment. Our proposed system works

Table 5.5: Comparing the accuracy of Feature based classification to structural rule based classifiers

Datasets		Accuracy (%)		
training	test	Feature Based	XRules	CBA
LOG1	LOG2	<b>85.88</b>	82.99	77.23
LOG2	LOG3	<b>87.04</b>	84.61	76.43
LOG3	LOG1	<b>84.56</b>	83.81	75.70

better than both XRules and CBA using the whole feature space. Table 5.5 reports the comparison and the amount of improvement we achieved; this highlight the advantage of our proposed approach.

### 5.3 Text Classification on 20NewsGroups Dataset

Besides XML classification, it is necessary to evaluate our proposed text classification approach based on word clustering from different perspectives, most importantly processing time and the improvement it achieves compared to a simple bag of words representation of documents. We mentioned earlier that for text classification, using all of the words in the process of model creation is not necessary and feature reduction has always been applied to text datasets. Our experimental results on the 20NewsGroups dataset show that although using all of the words is not necessary, using all of the words does not degrade the quality of our proposed classification approach. Furthermore, low computational cost of our implementation using Inverted Index structure enables us to use all of the words for classification with little time spent on feature extraction.

In order to compare our classification accuracy with Naive-BOW (naive bag of

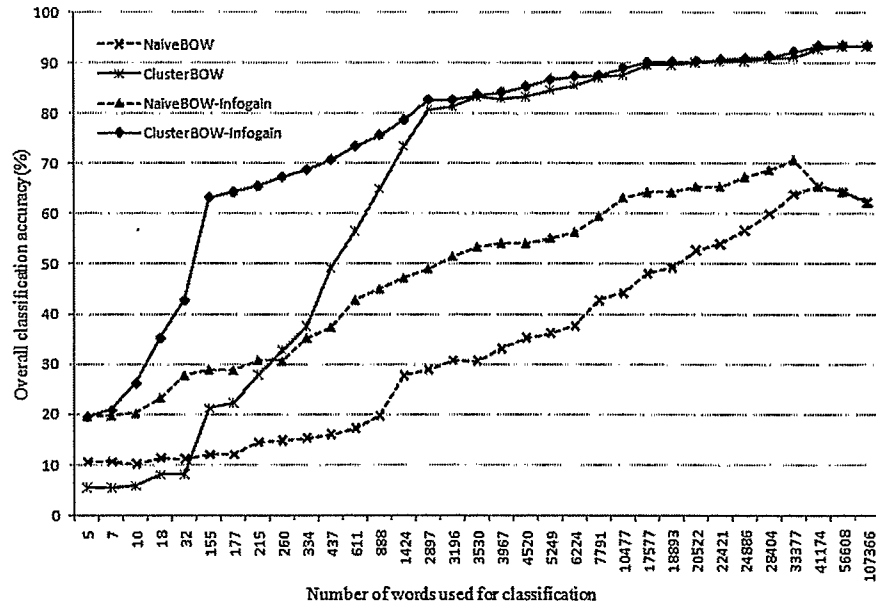


Figure 5.3: Comparison of Naive-BOW and Cluster-BOW using 10-Fold Cross Validation and different number of words

words) combined with Linear-SVM approach, several experiments are conducted using different number of words for training and different criteria for word filtering. In all experiments, we have chosen 20 as the number of classes for the number of features(clusters) in our approach; and we used multiple binary SVM classifiers for multi-class classification.

Figure 5.3 shows the average classification accuracy using different number of words. We have considered two different criteria for filtering the words based on: 1) word support (number of documents in which the word appears) and 2) Information Gain; since it is possible to filter any desired number of words using information gain, we choose the numbers on the  $X$  axis based on the first criteria. Minimum



support of the word is changed from 1 to 7,000 in order to find the number of words that pass the filter. We have to change the support value gradually when the support is small since there is a considerable change in the number of words that pass the filter in the initial steps. When the minimum support is set to 1, which means we are interested in all of the words, on average 107,366 words, from the training portion of cross validation splits pass the filter. Changing the support from 1 to 2 results in a significant drop, meaning that many words appear only once in the training set. Only 56,608 words pass the filter on average when the support is changed from 1 to 2.

Unlike the initial steps, in the consequent steps, there is no need to change the value by 1. For example, when we change the support from 400 to 500, the number of words changes from 611 to 437. Therefore, we could increase the step as the support gets higher. At the very end of the *Xaxis*, we have 5 and 7 words filtered after using 6,000 and 7,000 as minimum support, respectively. Every time, both the average classification accuracies of naive bag of words and clustering bag of words are measured. The best accuracy achieved by Linear-SVM on simple BOW representation of documents is 65.48% using 41,174 words and after that the accuracy goes down slightly as a result of extremely huge dimensionality and sparseness. In our approach, on the other hand, even adding the words that have appeared only twice in the training set increases the accuracy from 92.82% to 93.39%. Therefore, using minimum support as our filter, we achieve the best accuracy using 56,608 words, and the accuracy does not increase when the rest of the words are added in the final step.

Information gain was also used for filtering the words. In order to keep experi-

ments using this filter consistent with the ones performed based on minimum support, we choose the same number of top words as we had on the  $X$  axis after sorting the words based on information gain. In order to compute the information gain of the words with respect to the class labels, first we find the mutual information gain of every  $\langle \text{word}, \text{class} \rangle$  pair using Equation 5.1 first presented in [43].

$$I(w, c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)} \quad (5.1)$$

After finding the mutual information gain of every class and every word, we use the summation of over the set of classes to find the overall information gain of the word. Figure 5.3 shows how both systems can perform better using the information gain criteria with fewer number of words; still our feature extraction technique based on word clustering and minimum support outperforms the Naive-BOW even after the effective information gain feature reduction. Best performance is achieved by our approach after information gain filter whose accuracy curve is always over all the other curves. However, in order to get the highest possible accuracy, we still need to add most of the words to the clusters. The classifier built after feature reduction by information gain and then feature extraction through clustering achieves the same highest accuracy with 41,174 words on average while the highest possible accuracy we can get from Naive-BOW approach is 70.74% using top 33,377 words after sorting by the words based on the second criteria. It is worth to highlight that we have used all the 20 classes and all the documents for classification, and our reported results should not be compared with same projects that use only top level partitions of the 20NewsGroups dataset for classification.

Besides the experiments summarized in Figure 5.3, we also performed experi-

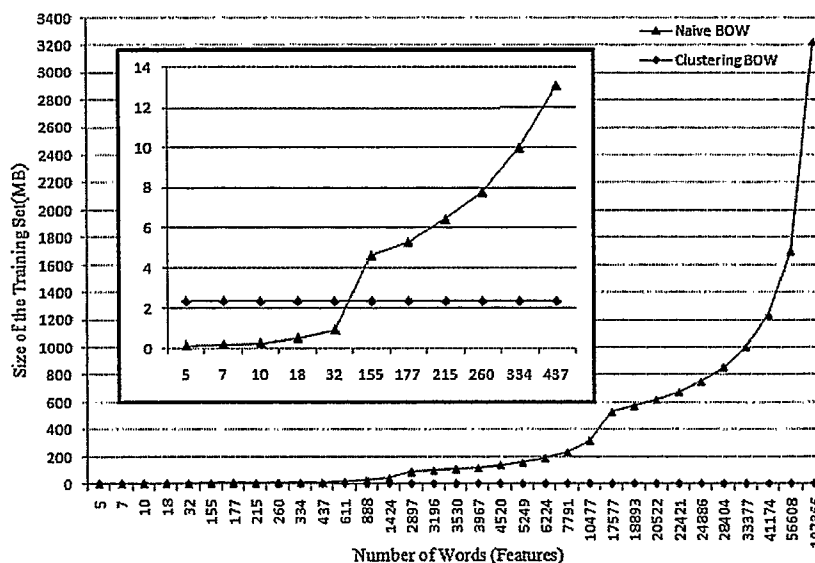


Figure 5.4: Size of the training set using different number of words for classification

ments on our approach, this time using K-Means hard clustering algorithm. While soft clustering of words achieves such high classification results, on the other hand, using K-Means clustering and same number of clusters (20) resulted in accuracies all less than 50%, using different number of words which means with hard clustering of words we either loose information and can't build an accurate classifier or the number of clusters required is much more than the ones in soft clustering. Considering either case supports the validity of our argument about the choice of clustering algorithm in the beginning of this thesis.

Figure 5.4 shows the size of the training set using different number of words by both Naive-BOW and our approach. It is visible that our training set size is very small and always constant due to the same number of features used in every experiment. On the other hand, the size of the training set using Naive-BOW approach

increases dramatically as we add more words to the feature vector, simply because the dimensionality increases and this will greatly affect the performance of the system in terms of both memory usage and processing time as presented in Table 5.3.

# of Words	SVM Train-Test (ms)		Clustering (ms)	Other (ms)	Total (ms)	
	Naïve-BOW	Feature-Based	Feature-Based	Feature-Based	Naïve-BOW	Feature-Based
100	31676	32694	685	2857	<b>31676</b>	36236
200	50259	33939	933	3325	50259	<b>38197</b>
500	91544	26231	1168	3578	91544	<b>30977</b>
1000	154079	21742	1282	4144	154079	<b>27168</b>
2000	288746	18245	2540	4947	288746	<b>25732</b>
5000	688622	15033	10870	6532	688622	<b>32435</b>
10000	1355100	13207	21957	8082	1355100	<b>43246</b>
20000	2688023	10982	41782	10909	2688023	<b>63673</b>

As we can see in Table 5.3, the processing time of our system on a typical personal computer (*2.1 GHz CPU, 2GB Main Memory*) is considerably faster than the Naive-BOW approach except only for the cases where we are using a very small percentage of words that could not result in acceptable accuracies whatsoever. The amount of time spent on training and testing by the SVM model decreases as we use more words for classification in our approach and it's no surprise since the size of the dataset is the same, but documents are easily separable when more words are used and SVM performs faster. Using Naive representation of the documents, the only processing time is SVM training and testing while in the feature based approach there are other computational steps as well; mainly clustering of the words and finding word vectors using the inverted index structure; also matrix multiplication of the clustering result

and document vectors; we have reported the sum of the latter two steps as other computational cost for the feature based approach. Although we are paying little preliminary computational cost, our model creation is much more faster than Naive-BOW approach when reasonable number of words is used. It is worth mentioning that we are achieving this fast processing time by assuming that an inverted index structure is already available which is a quite reasonable and realistic assumption since every modern information system uses this index structure for its information retrieval purposes.

#### 5.4 Automatical Hierarchy Generation from the 20News-Groupos Dataset

As described in Chapter 4, we can take advantage of our dense feature vector and combine it with SVR in order to measure the similarity of classes to each other; then, create a hierarchy of subjects from the training dataset. The created Hierarchy could be further cut at any level to get any desired number of clusters. In order to create the hierarchy of subjects and be consistent with the most recent similar publication and hence have a basis for comparison, we have used the “*by-date*” version of the 20NewsGroups dataset with 18,846 documents, which is already split into two parts of train and test. The rest of the characteristics, including the number of classes and class labels are the same. After computing the distance between classes, we get a 20 by 20 symmetric distance matrix with 190 actual distance values between classes, which are all possible permutations of 2 out of 20 class labels. Having calculated the distance between classes, we could hierarchically cluster objects. Table 5.6 shows the

5 smallest distance values between classes. Small distance between classes indicates similarity. Table 5.1 includes the full names of classes.

Classes	atheism	tlk.religion	politic.gun	os.win	ibm.hardware
	tlk.religion	soc.religion	politic.misk	ibm.hardware	mac.hardware
Distance	0.5204	0.6267	0.7599	0.8838	0.9036

Table 5.6: Most similar classes in 20NewsGroups dataset based on our SVR-based distance measure

Although we have only used bag of words representation of documents and have not used any semantical measure for similarity, it is observable that the most similar categories are also conceptually close to each other. For example, the category that covers documents about *Atheism* is very close to one of the categories covering *Religion* related discussions. Other religion related categories are also similar. On the other hand, we can see that classes under which documents related to computer systems are categorized among the most similar classes.

The single linkage measure, as described in Chapter 4, takes the distance vector as input and creates a hierarchy of objects. In each step, two of the objects (either cluster or single object) are merged into one cluster until we have a cluster of all objects. Because we have 20 classes and each step merges 2 objects into one cluster, obviously the clustering algorithm has 19 steps. Table 5.4 and Figure 5.5 show the steps of the hierarchical clustering algorithm. Class labels are alphabetically sorted and numbered from 1 to 20, and every newly created cluster (object) is assigned a new number starting from 21. Each step merges to objects until, at the end, object number 39 created as a result of the last step groups all of the class labels into one cluster.

We can see in the steps of the algorithm that besides religion and computer

Step	Cluster 1	Cluster 2	Distance	Result
1	alt.atheism(1)	talk.religion.misc(20)	0.5204	21
2	soc.religion.christian(16)	21	0.6267	22
3	talk.politics.guns(17)	talk.politics.misc(19)	0.7599	23
4	comp.os.ms-windows.misc(3)	comp.ibm.pc.hardware(4)	0.8838	24
5	comp.sys.mac.hardware(5)	24	0.9036	25
6	misc.forsale(7)	25	0.9098	26
7	comp.graphics(2)	comp.windows.x(6)	0.9185	27
8	26	27	0.9399	28
9	rec.sport.baseball(10)	rec.sport.hockey(11)	0.9418	29
10	22	23	0.9477	30
11	sci.electronics(13)	28	0.9504	31
12	rec.autos(8)	rec.motorcycles(9)	0.9549	32
13	31	32	0.9618	33
14	sci.crypt(12)	30	0.9710	34
15	sci.space(15)	33	0.9796	35
16	talk.politics.mideast(18)	34	0.9797	36
17	35	36	0.9831	37
18	sci.med(14)	37	0.9859	38
19	29	38	0.9908	39

Table 5.7: Steps of the Single-linkage hierarchical clustering algorithm

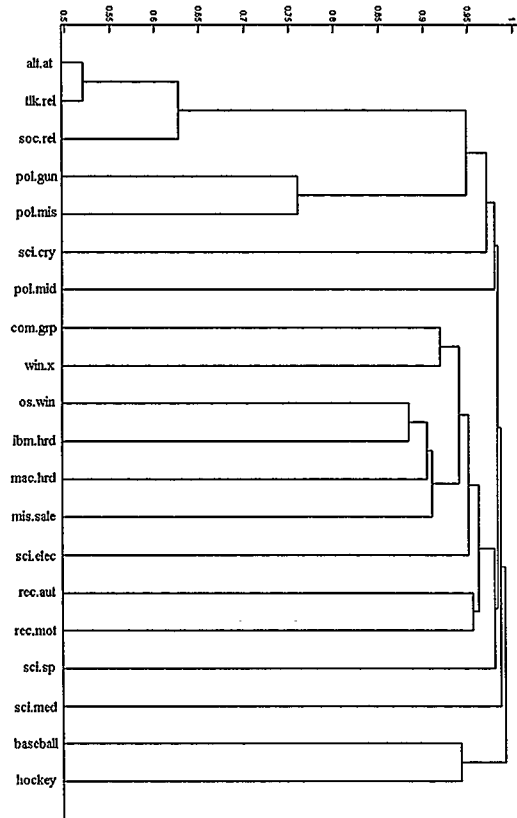


Figure 5.5: Automatically generated hierarchy from 20NewsGroups dataset

systems, sport categories and also automobile and motorcycle categories are similar to each other. This means that our suggested way of creating hierarchy of classes makes conceptual sense. Figure 5.5 is the dendrogram that shows the process visually.

Besides clustering of class labels and using the hierarchy for classification, as mentioned before, the created hierarchy could also be used for facilitating user queries after assigning labels to the larger categories in the intermediate levels.

Authors of [58] proposed another way of measuring the similarity between classes



based on the *linear discriminant projection*; and based on their similarity measure, they create a hierarchy of classes. By cutting their created hierarchy, they cluster the dataset into 8 top level categories and compare. Further, they use the first 1,000 words from the dataset according to the information gain for classification and report their accuracies. Table 5.4 shows their automatically created categories by linear projection.

Group	Members
1	<i>alt.atheism, talk.region.misc, talk.politics.guns, talk.politics.misc, talk.politics.mideas</i>
2	<i>comp.os.mswindows.misc, comp.sys.ibm.pc.hardware, comp.graphs, comp.sys.mac.hardware, comp.windows.x</i>
3	<i>sci.space, sci.med, sci.electronic</i>
4	<i>rec.sport.baseball, rec.sport.hockey, rec.motorcycles</i>
5	<i>misc.forsale</i>
6	<i>soc.religion.christian</i>
7	<i>rec.autos</i>
8	<i>sci.crypt</i>

Table 5.8: 8 high level categories of 20NewsGroups dataset proposed in [58], created by linear projection.

In order to compare our classification accuracy with the previous research in the area, we cut our hierarchy at an appropriate level to get 8 top level classes. Figure 5.6 shows the same dendrogram of Figure 5.5. Top level categories are shown with different line styles. Cutting the dendrogram in order to get 8 top level categories is similar to executing only the first 11 steps of clustering in Table 5.4. The first two groups of classes are similar to each other in both approaches. The first group mostly belongs to classes related to politics and religion, and the second group is the group of computer related categories. Baseball and Hockey are both in the same

Group	Members
1	<i>alt.atheism, talk.region.misc, talk.politics.guns, talk.politics.misc, soc.religion.christian,</i>
2	<i>comp.os.mswindows.misc, comp.sys.ibm.pc.hardware, comp.graphs, comp.sys.mac.hardware, comp.windows.x, sci.electronic, misc.forsale</i>
3	<i>rec.sport.baseball, rec.sport.hockey</i>
4	<i>rec.autos, rec.motorcycles</i>
5	<i>sci.med</i>
6	<i>talk.politics.mideas</i>
7	<i>sci.space</i>
8	<i>sci.crypt</i>

Table 5.9: 8 high level categories of 20NewsGroups dataset created by our algorithm using Support Vector Regression Machine.

categories in both tables. However, the linear projection approach also groups the *rec.motorcycles* class with these two while it doesn't seem to be relevant to these two topics. Our approach groups this class and *rec.autos* in one cluster, which makes much more conceptual sense. Linear projection groups all categories on science in one cluster, while in our clustering of subjects, each of these is a separate cluster by itself. From one perspective, we could consider them all science related topics that should be under the same categories; but from another perspective, looking at the specific scientific topics we will realize that they are not related to each other whatsoever (Medical Science, Cryptography and Space).

Comparing the accuracies of the classifier models created based on these categories could also give us a measure for the goodness of clustering. Authors of [58] have reported 96.3% as their best achieved accuracy with a Linear-SVM model created from the first 1,000 words according to information gain. We achieved 98.05%

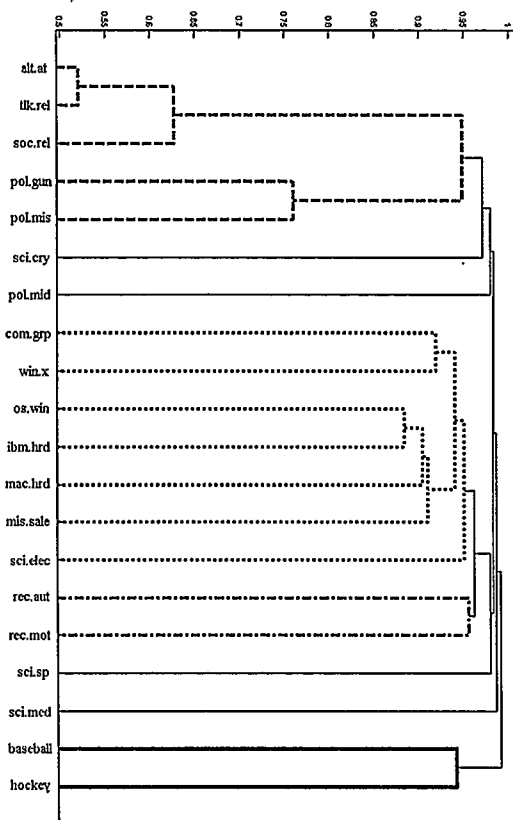


Figure 5.6: Automatically generated hierarchy from 20NewsGroups dataset

accuracy using the same classifier model and our feature extraction technique, altogether, there is no unique way of deciding which grouping of class labels makes more conceptual sense. However, our higher achieved accuracy could be an indicator of better grouping of classes using the SVR-based distance measure presented in chapter 4.

## Chapter 6

### Summary, Conclusion & Future Work

In this closing chapter we include a summary of what has been accomplished and our concluding remarks. The main points drawn out of the testing phase are presented in Section 6.2. Section 6.3 discusses some of the limitations and suggests some possible directions for future research.

#### 6.1 Summary

In this thesis, we proposed a new general framework for document classification based on extracting features from different aspects of the document. We described how it is possible to apply the proposed framework to text and XML document classification problems. We extract from each aspect of the document a feature vector that represents the document in a compact format and captures valuable information that can be used later on to build an accurate classifier using any of the well known classification techniques. One main advantage of this representation is that it converts the document classification problem to a classical classification problem that could be approached in many different ways using the extensive previous research in the area of classification. Although different types of documents have different elements, the only element which is in common among them is text. In this thesis, we proposed a novel approach for IR based classification based on word clustering and using each cluster as a feature. We took advantage of this compact feature based representation

and also proposed a solution for automatically generating a hierarchy of subjects. Comparison of our results proves the validity of our approach. Experiments have shown that combining textual and structural feature spaces improves the accuracy due to a richer source of knowledge we feed to any optimum and efficient classification technique.

## 6.2 Conclusion

By running experiments on XML and text datasets, we evaluated the applicability of our proposed framework. First, we conducted experiments on XML datasets and performed feature extraction. In order to extract features from text and structure of the documents, we conducted experiments to find the optimal number of features to be extracted from each aspect. Concatenation of feature vectors combined with well known classification techniques was able to outperform XML document classification techniques that work solely based on either text or structure. Furthermore, we showed how our novel textual feature extractor is successful to supersede IRC text classification technique that considers words based on clustering of documents. We also ran extensive experiments on the 20NewsGroups dataset and compared accuracy, memory requirements and also performance of our approach with Naive bag of words feature based representation of documents and observed great improvement in terms of all of the mentioned factors.

After classification on 20NewsGroups with all of the classes, we generated a hierarchy of classes using our compact feature based representation of documents and the method proposed in Chapter 4 for hierarchy generation. From the generated

hierarchy, we identified 8 high level categories of the dataset. Interpretation of the generated results shows that our clustering of class labels makes conceptual sense. Besides conceptual interpretation, we also evaluated the generated hierarchies in terms of classification accuracy and showed how it results in more accurate classification compared to the most recent similar research in literature. Altogether, our text classification results on XML datasets compared with IRC approach, our fast and accurate classification on the 20NewsGroups dataset compared to Naive bag of words approach and our high classification accuracy on 8 high level categories support the validity of our textual feature extraction. Experimental results on XML datasets also prove that the combination of our accurate text classifier with structural features extracted after frequent structural pattern mining results in a more accurate XML document classifier compared to previous approaches that mostly ignore one aspect of the document.

### 6.3 Future Work

This thesis mainly focused on creating a compact textual feature space from the original representation of XML documents and combining it with structural features extracted in the form of classifier rules. Our structural feature space construction could be also compressed by clustering of frequent structural patterns. For this purpose, it is necessary to define a similarity measure between sub-structures. This could be either done using the coverage of each rule on training documents which results in a high dimensional and sparse vector. Clustering of structural features based on this representation requires more sophisticated clustering algorithms such

as distributional clustering based on Information Bottleneck technique, otherwise using conventional clustering techniques does not seem to be promising according to high dimensionality and sparsity. Another alternative for clustering could be by considering every frequent structural pattern a word and repeating a similar process as we did for word clustering. Similarity measure for structural patterns could be also defined based on the similarity of tree structures. Investigating different alternatives for defining similarity between frequent structural patterns and using appropriate clustering technique in order to extract a compact structural feature vector for every document is one of the necessary extensions on the current work.

Our textual feature vector construction focuses only on the bag of words representation, which has been the most popular document representation in the literature. However, semantical closeness of the words could also be used as a factor for both word vector formation and feature vector construction after word clustering. Incorporating ontology in the process and considering semantical similarities between words and subjects could extend the scope of this work in order to create a feature vector that could be used in practical modern information systems.

We have already evaluated the effectiveness of our approach on the 20NewsGroups dataset, which is a multi-class document directory. However, most of the real world document directories, such as google directory, include documents that are multi-labeled as well meaning that each document could belong to more than one category. This could also be done using SVR and using the membership probabilities in classes as an indicator of the membership degrees of documents in classes.

Finally, online document classification is one of the requirements of every modern document classification system. Most of the existing search engines and information

systems explore the web using a crawler to discover new reachable documents from existing documents based on the links between them. With our feature vector representation of text documents, we can easily extend the scope of this work in order to create an online document classification system by constructing feature vector for newly discovered documents which is consistent with the existing representation. Further, we can use documents that are predicted to strongly belong to one of the classes for future training and improving the current model by adding recently observed words to the word set and dynamically grow the existing word clusters to include the new words. Each of these steps specifically, dynamically expanding the clusters requires extensive investigation and experiments which is beyond the scope of this thesis and will be our future research direction based on the current work.



## Bibliography

- [1] Extensible Markup Language (XML), Webpage:  
(<http://www.w3schools.com/XML/>)
- [2] T. Bray, J. Paoli and S. McQueen: "Extensible Markup Language (XML) 1.0 W3C Recommendation," *W3C*, February, 1998
- [3] A. Cavoukian: "Data Mining: Staking a Claim on Your Privacy," *Information and Privacy Commissioners Report*, Ontario, Canada, 1998.
- [4] M. H. Dunham: "Data Mining: Introductory and Advanced Topics," *Prentice Hall*, 2002.
- [5] M. H. Dunham: "Data Mining Techniques and Algorithms," *Prentice Hall Press*, 2000.
- [6] J. Han and M. Kamber: "Data Mining Concepts and Techniques," *Morgan Kaufmann*, 2000.
- [7] S. K. Murthy: "Automatic construction of decision trees from data: A multidisciplinary survey," *Data Mining and Knowledge Discovery, Vol.2, No.4*, pp.345-389, 1998.
- [8] R. O. Duda and P. E. Hart: "Pattern Classification and Scene Analysis," *Wiley and Sons, Inc*, 1973
- [9] M. A. Hall: "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning," *Proc. of ICML*, pp.359-366, 2000.

- [10] J. Rocchio, Relevance feedback in information retrieval, In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp.313-323 Prentice Hall Inc., 1971.
- [11] D. A. Hull, 1994. "Improving text retrieval for the routing problem using latent semantic indexing." *Proc. of ACM SIGIR*, Dublin, pp.282289, 1994.
- [12] S. DAlessio, K. Murray, R. Schiaffino, and A. Kershenbaum. "The effect of using hierarchical classifiers in text categorization." *Proc. of the Int. Conf. Recherche d'Information Assistee par Ordinateur*, Paris, pp.302313, 2000.
- [13] S. Dumais and H. Chen. Hierarchical classification of Web content. *Proc. of ACM SIGIR*, Athens, pp.256263, 2000.
- [14] T. Joachims, "Text categorization with support vector machines: learning with many relevant features." *Proc. of ECML*, Heidelberg, pp.137142, 1998.
- [15] A. Silberschatz, H.F. Korth, S. Sudershan: "Database System Concepts," *McGraw-Hill, Inc. New York, NY, USA*, 1998.
- [16] P. Berkhin: "A Survey of Clustering Data Mining Techniques", *Grouping Multidimensional Data*, Springer Berlin Heidelberg, pp.25-71, 2006.
- [17] I. Guyon, A. Elissee: "An Introduction to Variable and Feature Selection" *Journal of Machine Learning Research* 3, pp.1157-1182, 2003
- [18] C. Romualdi, S. Campanaro, D. Campagna, B. Celegato, N. Cannata, S. Toppo, G. Valle, G. Lanfranchi: "Pattern recognition in gene expression profiling using DNA array: a comparative study of different statistical methods applied to

cancer classification,” *Journal of Human Molecular Genetics*, Vol. 12, No. 8, pp.823-836, 2003.

- [19] L. M. Fu, E. S. Youn: “Improving reliability of gene selection from microarray functional genomics data,” *IEEE Transactions on Information Technology in Biomedicine*, pp.191-196, 2003
- [20] M. Khabbaz, K. Kiamehr, M. Alshalalfa and R. Alhajj, “Fuzzy Classifier Based Feature Reduction for Better Gene Selection,” *Proceedings of the International Conference on Data Warehouse and Knowledge Discovery, Springer-Verlag LNCS, Regensburg, Germany*, pp.334-344, 2007.
- [21] J.C. Bezdek, R. Ehrlich and W. Full, “FCM: The fuzzy c-means clustering algorithm.” *COMP. GEOSCI.*, Vol.10, no.2-3, pp.191-203. 1984
- [22] M.E. Futschik and N.K. Kasabov, “Fuzzy clustering of gene expression data.” *Proc. of IEEE FUZZ*, pp.414-419, 2002.
- [23] M. Sasaki and K. Kita. “Rule-based text categorization using hierarchical categories.” *Proc. of IEEE Int. Conf. on SMC*, pp.2827-2830, 1998.
- [24] K. Wang, S. Zhou, and Y. He. “Hierarchical classification of real life documents.” *Proc. of SIAM SDM*, Chicago, 2001.
- [25] A. S. Weigend, E. D. Wiener, and J. O. Pedersen. “Exploiting hierarchy in text categorization.” *Information Retrieval*, 1(3):193-216, 1999.
- [26] A. Sun and E. Lim, “Hierarchical Text Classification and Evaluation.” *Proc. of IEEE ICDM*, pp.521-528, 2001.

- [27] N. Fuhr and G. Weikum, "Classification and Intelligent Search on Information in XML." *Bulletin of the IEEE Technical Committee on Data Engineering*, 25(1), 51-58, 2002.
- [28] M.J. Zaki and C. Aggarwal, "XRULES: An Effective Structural Classifier for XML Data." *Proc. of ACM SIGKDD*, pp.316-325, 2003.
- [29] M. Theobald, R. Schenkel and G. Weikum, "Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data." *Proc. of WebDB*, pp.1-6, 2003.
- [30] B. Liu, W. Hsu and Y. Ma: "Integrating Classification and Association Rule Mining," *Proc. of ACM SIGKDD*, 1998.
- [31] W. Li, J. Han and J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules," *Proc. of IEEE ICDM*, 2001.
- [32] X. Yin and J. Han, "CPAR: Classification based on predictive association rules," *Proc. of SIAM SDM*, 2003.
- [33] X. Wu, "An Inductive Learning System for XML Documents." *Proc. of Inductive Logic Programming*, 2007.
- [34] M. J. Zaki. "Efficiently Mining Frequent Trees in a Forest," *ACM SIGKDD*, 2002.
- [35] <http://www.tartarus.org/~martin/PorterStemmer/index.html>
- [36] <http://www.lextek.com/manuals/onix/stopwords1.html>

- [37] J. Punin, M. Krishnamoorthy, M. Zaki. LOGML: Log markup language for web usage mining. In WEBKDD Workshop (with SIGKDD), August 2001.
- [38] C. Aggarwal, S. Gates, P. Yu. On the merits of using supervised clustering to build categorization systems. SIGKDD, 1999.
- [39] <http://www.cs.rpi.edu/~zaki/software/logml/> Last Accessed on December,9,2007
- [40] H. Lu, R. Setiono, and H. Liu, Effective data mining using neural networks, IEEE Trans. Knowledge Data Eng., vol. 8, pp. 957-961, Dec. 1996.
- [41] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. 1984. Classification and Regression Trees. Monterey, CA: Wadsworth International Group.
- [42] <http://people.csail.mit.edu/jrennie/20Newsgroups/>
- [43] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," *Proceedings of ACM-CIKM*, 1998.
- [44] Garboni, C., Masegla, F., Trousse, B.: Sequential pattern mining for structure-based XML document classification. In: Workshop of the Initiative for the Evaluation of XML Retrieval. (2005) 458-468
- [45] L. Candillier, I. Tellier and F. Torre, "Transforming XML trees for efficient classification and clustering." *Proc. of Workshop of Initiative for the Evaluation of XML Retrieval*. pp.469-480, 2005.

- [46] P.F. Brown, et al. "Class-based n-gram models of natural language." *Computational Linguistics*, 18(4):467-479, 1992.
- [47] L. D. Baker and A. McCallum. "Distributional clustering of words for text classification." *Proc. of ACM SIGIR*, pp.96103, 1998.
- [48] R. Bekkerman, R. El-Yaniv, Y. Winter and N. Tishby. "On feature distributional clustering for text categorization." *Proc. of ACM SIGIR*, pp.146153, 2001.
- [49] N. Slonim and N. Tishby. "The power of word clusters for text classification." *Proc. of ECIR*, 2001.
- [50] S. Russell and W. Lodwick, "Fuzzy clustering in data mining for telco database marketing campaigns," *Proc. NAFIPS*, pp.720726, June 1999.
- [51] H. Shah, J. Undercoffer and A. Joshi, "Fuzzy clustering for intrusion detection Fuzzy Systems." *Proc. of IEEE FUZZ*, vol.2, pp.1274-1278, 2003.
- [52] <http://people.csail.mit.edu/jrennie/20Newsgroups/>
- [53] S. Mitra, S. K. Pal, and P. Mitra. "Data mining in soft computing framework: A survey." *IEEE Transactions on Neural Networks*, pp.3-14, 2002.
- [54] I. Dhillon, S. Mallela and R. Kumar, "Enhanced word clustering for hierarchical text classification." *Proc. of ACM KDD*, pp.191-200, 2002.
- [55] F. Pereira, N. Tishby, and L. Lee. "Distributional clustering of english words." *Proc. of the Annual Meeting of ACL*, pp.183190, 1993.

- [56] A. J. Smola, B. Schlkopf, "A Tutorial on Support Vector Regression." *Statistics and Computing Journal*, pp.199-222, 2004.
- [57] A. K. Jain, R. C. Dubes, "Algorithms for clustering data." *Prentice Hall: New Jersey*, 1988.
- [58] T. Li, S. Zhu, M. Ogihara, "Hierarchical document classification using automatically generated hierarchy," *Journal of Intelligent Information Systems*, pp.211-230, 2007.