CLASSIFYING WEB SERVICES WITH AND WITHOUT ASSOCIATION RULES

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

Thapar University, Patiala

By:
SHAILJA
(80732017)

Under the supervision of:
Ms. Shalini Batra
Lecturer (SS)

MAY 2009

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Classifying Web Services With and Without Association Rules"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Shalini Batra and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(SHAILJA)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Ms. SHALINI BATRA)
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

for 26/06/2009

(Dr. RAJESH BHATIA)
Astt. Professor & Head
Computer Science & Engineering Department
Thapar University,
Patiala

9/7/09

(Dr. R.K.SHARMA)
Dean (Academic Affairs)
Thapar University,
Patiala.

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from **Ms. Shalini Batra,** Lecturer (SS), Computer Science and Engineering Department. I am thankful for their continual support, encouragement, and invaluable suggestions. She not only provided me help whenever needed, but also the resources required to complete this thesis report on time.

I am also thankful to **Dr. Rajesh Bhatia**, Head, Computer Science and Engineering Department for her kind help and cooperation.

I would also like to thank all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would like to say thanks to all my friends for their consistent support. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

I am highly grateful to my parents and Rama for the inspiration and ever encouraging moral support, which enabled me to pursue my studies.

Shailja

SHAILJA

ABSTRACT

The transition of the World Wide Web from a paradigm of static Web pages to one of dynamic Web Services raises a new and challenging problem of locating desired Web Services. With the expected growth of the number of Web Services available on the web, the need for mechanisms that enable the automatic categorization to organize this vast amount of data becomes important.

Web Services classification is the task of automatically sorting a set of documents into categories from a predefined set. Automated Web Services classification is attractive because it removes the need of manually organizing document bases, which can be too expensive, or simply not feasible given the time constraints of the application or the number of documents involved. The process involves text mining and classification of WSDL (Web Service Description Language) documents based on Association rules. Text mining techniques are used at the first stage, namely preprocessing, to extract relevant information from a WSDL documents. Textual documentation, operations and arguments accompanying descriptions of Web Services are preprocessed. Association rules are applied to analyze the degree of dependency between contents of WSDL and category of the Web Services. A machine learning classifier is used at the last stage to categorize the documents under different categories. This classifier deduces a sequence of candidate categories for a preprocessed Web service description. Here the concept of Association rules in context of Web Services is used and its performance is compared with the primitive algorithms for classification.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Web Services are loosely coupled reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over the internet. Automatic text classification is an important task that can help people finding information on huge online resources. Classification or categorization is the task of assigning objects from a universe to two or more classes or categories. Text classification research and practice has exploded in the past decade.Current technologies for publishing Web Services, for example UDDI, enable providers to manually assign a category to their Services from a number of predefined choices such as business, educational, finance, scientific, etc .

Assigning a proper category to a Service can be a tedious and error prone task due to the large number of categories usually present in Web Services registries. Service consumer has to manually search published services by category. Since the categorization of services and maintenance of repositories has to be done manually by human entities, the classification task becomes considerably difficult, heavy and error-prone in practice, due to several issues (e.g. huge size of taxonomies in real-world applications, multiple people involved in maintaining or sharing services in a common repository, several distributed repositories being shared, etc.).

Automatic mechanisms can help in assisting service publishers in the categorization task, in order to reduce the effort required, and promote globally consistent classification decisions, even when several users are involved. Users will put a query and an automatic classifier will determine the most suitable categories where to look for the needed functionality. As a result, both service providers and consumers will be able to exploit Web Services technologies in a better manner.

1.1 The Notion of the Web Services

Web Services is given as: "any process that can be integrated into external systems through valid XML documents over Internet protocols" [1]. This definition outlines the general idea Web Services are built for. Unlike Services in general, Web Services are based on specifications for data transfer, method invocation and publishing. This is often misunderstood and when a Web Services is mentioned it sometimes refers to a general service provided on the Web, like the weather forecast on a Web page for example. The weather forecast is a service and provides its functionality for a variety of users but unless it comprises an interface to communicate with other applications via SOAP it is no Web Services by definition.

1.1.1 Communication Architecture

Web Services can be seen as software components with an interface to communicate with other software components. They have certain functionalities that are available through a special kind of Remote Procedure Call. Furthermore Web Services cannot be viewed or used with an ordinary browser. They require a unified form of messaging embedded in a XML document. This communication architecture contains three subcomponents:

1. **Consumer:** This denotes the entity utilizing the Web Services. This is another application in most cases.
2. **Transport:** It defines the means for the communication the consumer uses while interacting with a service.
3. **Provider:** The service provider.

1.1.2 Platform Elements

Four basic technologies that facilitate Web Services are:

- SOAP (Simple Object Access Protocol)
- XML((Extensible Markup Language)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language

XML is a format for data exchange and description, improving or eliminating marshalling and unmarshalling [2]. It provides interoperability between different platforms. SOAP is a simple XML-based protocol to let applications exchange information over HTTP [3]. It is important for application development to allow Internet communication between

programs. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. Web Services description is given by the Web Services Description Language. (WSDL) describes services as collections of network endpoints, or ports [4]. It is an XML document with a defined grammar where the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. Universal Description Discovery and Integration (UDDI) solve the problem of finding the corresponding description (WSDL document). UDDI provide a method for publishing and finding service descriptions [5]. It is a directory where Web Services can be registered and assigned to a service provider so that service consumer can search and locate the service. Once the WSDL file is downloaded from the UDDI directory, it can be used as an interface to invoke the Web Services.

1.2 Data Mining Concept

Data mining is the discovery of interesting, unexpected or valuable structures in large datasets. Association rule discovery is part of a larger field of study called data mining a field that consists of techniques to automatically find interesting patterns and trends in large collections of data [6]. There are different functions of data mining:

- Sequence analysis for time dependent data,
- Association Rule analysis which tries to determine relations between the data,
- Summarization which describes subsets of the datasets by computing the median and standard deviation,
- Classification which map datasets to one or more predefined classes and
- Cluster analysis which, similar to classification, groups datasets into clusters, by means of similarity metrics

Data mining techniques are used to explore the relationship between the Web Services description and category.

1.3 Introduction to Association Rules

In a database of transactions D with a set of n binary attributes (items) I, a rule is defined as an implication of the form

$X \Rightarrow Y$ where $X, Y \subseteq I$ and $A \cap B = \phi$.

The sets of items (for short item sets) X and Y are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively. The support, supp(X), of an item set X is defined as the proportion of transactions in the data set which contain the item set.

The confidence of a rule is defined

$Conf(X \Rightarrow Y) = supp(X \cup Y) / supp(X)$.

Association rules are implication rules that inform the user about items most likely to occur in some transactions of a database [7]. It is advantageous to use them because they are simple, intuitive and do not make assumptions of any models. Their mining requires satisfying a user-specified minimum support and a user-specified minimum confidence from a given database at the same time. To achieve this, association rule generation is a two-step process. First, minimum support is applied to find all frequent item-sets in a database. In a second step, these frequent item-sets and the minimum confidence constraint are used to form rules. While the second step is straight forward, the first step needs more attention.

Steps in finding the association rules:

Suppose one of the large item-sets is L_k, $L_k = \{I_1, I_2,\ldots,I_k\}$, association rules with this item-set are generated in the following way: the first rule is $\{I_1, I_2, \ldots, I_{k-1}\} \Rightarrow \{I_k\}$, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine their usefulness. Those processes iterated until the antecedent becomes empty.

1.4 Classifying Web Services

The classification of Web Services can be carried out with respect to the content of the WSDL documents to be classified, and is done in a two-steps process:

- Retrieval of keywords in the WSDL documents;
- Classification of documents

Text mining is the discovery by computer of new, previously unknown information, by automatically extracting information from a usually large amount of different unstructured textual resources [8]. In Web Services, text mining is used for extracting term that is relevant to a particular category of services. It deals with accessing a Web

Services description to extract category related terms. Textual description of Web Services might be in the form of Web Services Description Language (WSDL) documents, coming from UDDI registries, as well as any other textual document provided as a documentation support for the service itself. Words are extracted from documents and then preprocessed. Successively, words are filtered by means of a stop–list, and normalized. The stop–list contains articles, prepositions, and in general words that are frequent in each query, and therefore cannot be discriminated. During the stemming phase, verbs are brought back to infinitive, plurals to singulars.

1.4.1 Service Classification

Document classification, is the process of automatically assigning documents to one or more predefined categories based on the contents of these documents [9]. Web Services classification is the task of assigning predefined categories to Web Services. Instead of manually classifying documents or hand-crafting automatic classification rules, statistical service classification uses machine learning methods to learn automatic classification rules based on human-labeled training documents.

Let C = {c1, c_2, c_m} be a set of Web Services categories (classes) and D = {d_1, d_2, d_n} a set of WSDL documents. The task of the Web Services classification consists in assigning to each pair (ci, d_j) of C x D (with $1 \leq i \leq m$ and $1 \leq j \leq n$) a value of 0 or 1, i.e. the value 0, if the document d_j doesn't belong to $c_{i..[10]}$

This mapping is sometimes referred to as the decision matrix:

	d_1	**...**	**d_j**	**...**	**d_n**
c_1	a_{11}	...	a_{1j}	...	a_{1n}
...	...	...	...	...	...
c_i	a_{i1}	...	a_{ij}	...	a_{in}
...	...	...	...	...	...
c_m	a_{m1}	...	a_{mj}	...	a_{mn}

Figure 1.1: Decision Matrix

The main approaches to solve this task are:

- Naive Bayes

- Support Vector Machine
- Rocchio
- Neural networks
- Decision rules

The classification of Services into domain–specific classes can be performed using any of the above classification approach. Prior to application of any classifier, sequences of words are obtained by preprocessing. All words are weighted with *TF-IDF* metric. In this way the whole document set is encoded in a matrix, where rows represent documents (vectors) and columns are the weighted words. No information about the position or the meaning of the words is used, i.e., no semantic is known using this matrix. Once the importance to the terms is given by the TF-IDF method, classification algorithm is applied over the data.

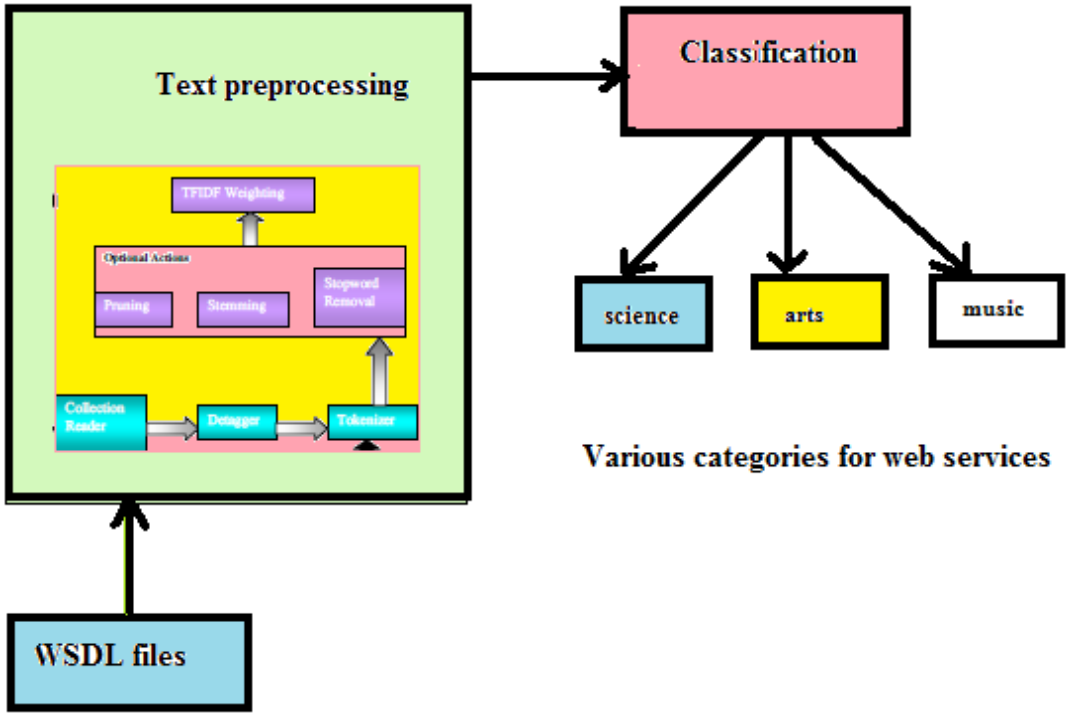

Figure 1.2: Classification process as a whole.

1.5 Structure of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 reviews the recent research in the area of the association rules and automatic and semiautomatic classification system for Web Services under the heading of Literature Review.

Chapter 3 gives an overview of the problem statement and describes the methodology used for solving the problem.

Chapter 4 focuses on the basic concepts of Web Services, association rules and its algorithms, Web Services classification process and various types of classifiers. It also describes the tools used during implementation phase.

Chapter 5 presents the results. It demonstrates the implementation, provides an overview of the main features.

Chapter 6 focuses on the conclusion of the work presented in this thesis and ideas for future research in this area are presented.

Chapter 2

Literature Review

In this chapter we review the published research related to our work in both of the issues described in introduction-namely finding association rules in Web Services and classification of Web Services. The work on association rule mining began with the development of the Apriori algorithm [11], and was further modified and extended. Since then, several attempts have been made to improve the performance of these algorithms. The partition algorithm [12] partitions the data into disjoint groups, processes each individually, and merges the intermediate results. It improves the overall performance by reducing the number of passes needed over the complete database to at most two. The incremental mining algorithm [13] incorporates the concept of data addition to validate the existing rules. The task of deriving new rules for data sets that grow incrementally is supported by the incremental algorithm. However, most of these algorithms are applicable to data stored in flat files.

The main characteristic of these algorithms is that they are main memory algorithms. In these algorithms, the data is either read directly from flat files or is first extracted from the DBMS and then processed in main memory. The algorithms implement their own buffer management schemes and the performance varies depending on the specialized data-structures used for buffer management. A few attempts have been made to build Web Services based mining approaches. Work in the field of Web Services mining has focused on integrating the mining functions with the machine learning.

Web Services are self contain, modular business applications that have open, Internet-oriented, standards-based interfaces [14]. During the past few years some efforts and research have been placed on assisting the developer to classify Web Services. As a result, some semiautomatic and automatic methods have been proposed. The different approaches are based on argument definitions matching document Classification techniques and semantic annotations matching.

MWSAF [15] is an approach for classifying Web Services based on argument definitions matching. First, MWSAF translates these definitions into a graph. MWSAF also translates real word formal descriptions of categories that have been specified by using Semantic Web languages. Then, MWSAF uses graph similarity techniques for comparing both. The main limitation of these matching approaches is that they do not attempt to reduce the distance between different coding conventions. In fact, MWSAF achieves low accuracy, which is shown in its implementation.

METEOR-S [16] describes a further improved version of MWSAF. The problem of determining a Web Services category is abstracted to a document classification problem. The graph matching technique is replaced with a Naive Bayes classifier. To do this, METEOR-S extracts the names of all operations and arguments declared in WSDL documents of pre-categorized Web Services. The main limitation of this approach is that it assumes independence between the name of an operation and its arguments. Clearly, the name, or header, of an operation and the name of its arguments might be related, e.g., print (Style, document) method signature. Therefore, this classifier seems to be based on a false premise. Although METEOR-S proposes a document classification approach, natural language documentation, usually present in WSDL files and service registries, is not considered. However, METEORS implementation results, for the same small data-set previously mentioned, shows an accuracy improvement with respect to MWSAF.

Assam [17] is an ensemble machine learning approach for determining Web Services category. Assam combines the Naive Bayes and SVM machine learning algorithms to classify WSDL files in manually defined hierarchies. Assam takes into account Web Services natural language documentation and descriptions. As reported, this approach is more accurate than similar approaches, even though its authors used a repository of 391 Web Services divided into 11 categories for implementing, which makes the validation of Assam stronger than the others. Previous efforts for classifying Web Services have several shortcomings. First, the classification approach proposed by MWSAF has shown low accuracy. Even though this work was evaluated with a set of only 24 services divided in two categories, the resulting accuracy was 0.625. Second, the classification-based version of MWSAF shown better accuracy, but this version is based on the false premise that an operation and its argument names are independent.

After going through the literature related to Web Services classification it has been analyzed that majority of approaches don’t consider Web Services interface description and its associated textual documentation, so we thought of working in this direction and

apply various information retrieval techniques to the data extracted from interface and textual description associated with the Web Services.

Chapter 3

Problem Statement

3.1 Problem Definition

During the literature survey it was found that some approaches have been proposed for automatic or semi-automatic classification of Web Services. However, these approaches have shown some limitations. Some approaches have low accuracy while some propose to classify Web Services on the basis of the definitions of operation arguments that belong to a particular category. It has been further observed that some of these methods do not exploit a Web Services interface description and its associated textual documentation. Finally, some of these methods assume the existence of additional sources of information, such as semantically annotated services, which might be either more expensive or more challenging to produce than manually classifying Web Services.
The main limitation of these approaches is that they do not attempt to reduce the distance between different styles for defining arguments present in standard descriptions. In this thesis we will exploit dependency between the Web Services category and its interface described by WSDL document for classifying Web Services. Association rules have been used for building Web Services Classifier for automatically classifying Web Services; this is, determining the category of a Web Services, given a set of predefined categories.

The main goals of this thesis are to

1. Build a classification system using association rule applied on the category of a Web Services, its operations and its textual documentation, namely argument definitions and comments written by developers.
2. Based upon this system we aim to analyze whether this system gives better accuracy than the primitive methods used for classification or not.
3. To analyze how the importance of a term to a particular category varies with its frequency and appearance in other documents.

3.1 Methodology

The step-by-step methodology to be followed in solving the automatic classification problem is given below:

1. Download WSDL files from the different UDDI repositories.
2. Apply text mining over these WSDL files. Text mining process involves four logical steps:
    a. First the names and comments are extracted.
    b. Combined names are split to generate different words.
    c. The terms are then filtered to remove the non relevant words, called stop words.
    d. Stemming is done to reduce terms to their stems.
3. Importance each word within a document is calculated through TF-IDF method.
4. .csv and .arff files are prepared using these terms.
5. Association rules are applied on the data set using data mining tools.
6. Classification algorithms are applied on same data set using tools of data mining.

We have analyzed the results using WEKA and Rapid miner, Open source data mining software.

Chapter 4

Web Service Classification

4.1 Publishing and Finding Web Services

Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web [18]. They are loosely coupled reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over the internet.

Web Services have primarily been designed for providing inter-operability between business applications. They can be used alone or in conjunction with other Web Services to carry out a complex aggregation or a business transaction [19]. A Web Services is described using a standard that provides all of the details necessary to interact with the service such as, message formats, transport protocols, and location. Web Services Description Language (WSDL) is an XML language that contains information about the interface, semantics of a call to a Web Services.

There should be a communication scenario between service requester, provider and broker. A Client is a software that makes use of a Web Services, acting as its ‘user’ or ’customer’. A Message is the basic unit of communication between a Web Services and a Client; data to be communicated to or from a Web Services as a single logical transmission. In this regards the service requester invokes services which are provided by service provider and service broker enables the service requester to dynamically find service provider or registry where services can be published or advertised. These can be performed through three activities including

publish means how the provider of a Web Services registers itself,

find means how an application finds a particular Web Services, and

bind means how an application connects to, and interacts with, a Web Services after it's been found [18].

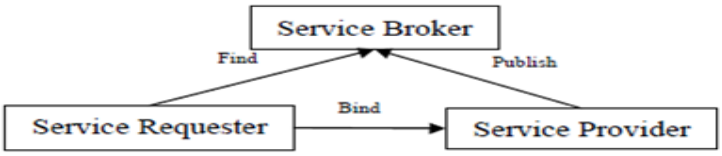

Figure 4.1: The basic model of service interaction [18].

4.1.1 UDDI (Universal Description Discovery and Integration

UDDI provides a specification that permits the publication and location of services in a universal service registry. It publishes an available service along with its call interface and semantics, thereby providing a discovery and meeting point for service providers and consumers. It basically provides a method for publishing and finding service descriptions. It is a directory where Web Services can be registered and assigned to a service provider, therefore forming a structure that resembles a yellow pages directory. Any user can browse the directory to search for a desired service and download the description in case of a match. UDDI is an industry initiative (ARIBA, IBM and Microsoft) to enable businesses to quickly and dynamically find and transact with each other. Searching a UDDI registry can be done by any UDDI browser or by the registry's Web interface if present. Microsoft for example, provides both a Web interface and the original UDDI registry [20] to search for an entry. A query with 'weather' as the search string returns about 10 results at the Microsoft UDDI registry. The corresponding WSDL File can be found in the overview document if it was specified. The UDDI registry is implemented as multiple peer nodes where the registration across all nodes is replicated by node operators. The data structure of the registry is XML yet a WSDL document cannot be published without additional precautions. The structure of a UDDI registry differs from a WSDL file because UDDI also supports business and service information. As a result, UDDI has no direct support for WSDL or any other service description mechanism. The UDDI specification defines four key types of information that might be needed to facilitate service use:

1. **Business information:** Contains information such as the business name, contact details, and so on. It might also include information categorized in a way similar to the format of the Yellow Pages.
2. **Service information:** Contains information about services relevant to a particular business function or a given service category. Each businessEntity might hold several businessServices.

3. **Binding information :** Contains information required to call a Web Services, such as network access point, supported interfaces, and so on. Each businessService might hold several bindingTemplates.
4. **Specification information:** To invoke a Web Services, you must know a service's location and the kind of interface the service supports. The bindingTemplate indicates the specifications or interfaces a service supports through references to specification information [12].

Once a Web Service is developed, its description is published and a link is send to a UDDI (Universal Description, Discovery and Integration) repository so that potential users can find it. When someone wants to use Web Services, request is send to UDDI repository for the WSDL file to find the location of the service, the function calls and how to access them. Then users can find this information in WSDL file for invoking the Web Services.

4.2 Need of WSDL

A central purpose of WSDL is to describe interfaces (formerly known as port-types) to Web Services. Actual Web Services may be viewed as remote implementations of these interfaces. In general, service providers/implementers could use a standard interface, extend a standard interface or develop there own. An interface contains a set of operations and each operation has a signature, which includes an operation name, input, output and exception messages. These messages have types that are defined using some XML-based schema language. Indeed, interface descriptions may be used to find candidate Web Services. Such descriptions are therefore critical for proper discovery and use of Web Services [3].

Figure 4.2 shows an e.g. WSDL document and its corresponding relevant information inside the box:

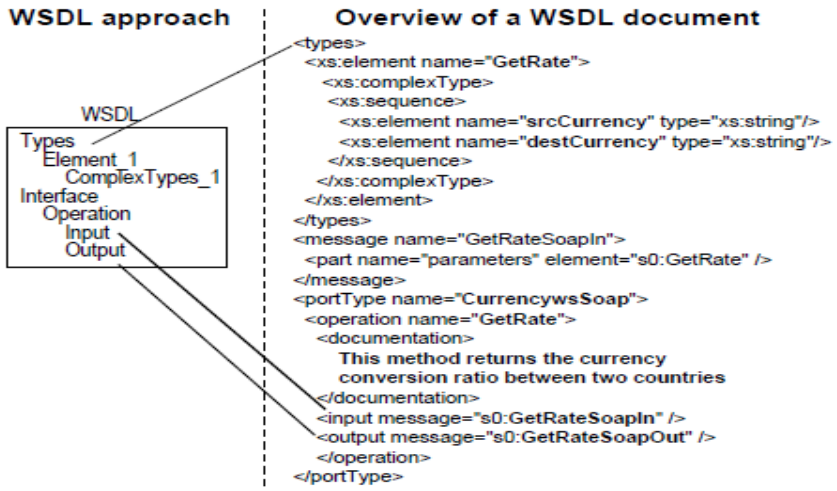

Figure 4.2: WSDL document [22]

WSDL describes four aspects of the Web Services.

- **Interface information**
  - What are its publicly available functions?
- **Data type information**
  - What information is required/produced by message requests and responses?
- **Binding information**
  - Which transport protocols can be used?
- **Address information**
  - Where is the service located?

4.2.1 How WSDL Works:

When a service provider wants to publish its Web Services, it first generates a Web Services Description Language (WSDL) file to describe its Web Services. WSDL documents are created using an IDE or a WSDL editor which provides templates for WSDL creation. Developers add information and comments to the WSDL document through any IDE like Netbeans XML schema editor. Then, the service is registered in the Universal Description, Discovery, and Integration (UDDI) repository and made available for invocation. The UDDI repository now contains all necessary information to identify this Web Services along with a URL that points to its corresponding WSDL file. Once a service requestor has queried the UDDI repository and found that this Web Services best suited its need, it can download the WSDL file of this service and use it to generate messages to interact with the Web Services [6].

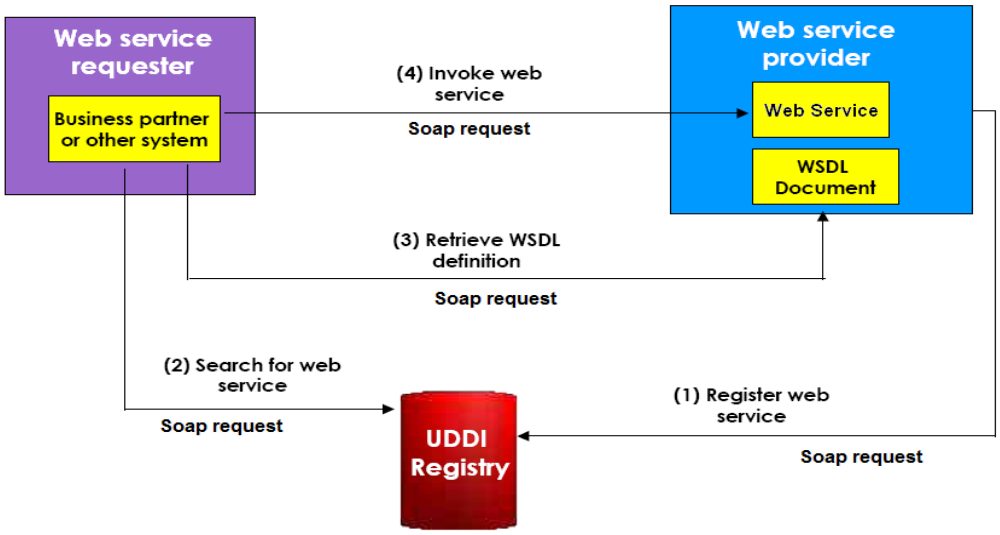

Figure 4.3: Web Services publishing, search and invocation [23].

The wealth of information available on the internet is currently being complemented by an ever-increasing number of services. These services offer the possibility not only to gain more specific types of information but also to interact with the sources of the information, changing the state of these systems and causing real world processes to occur. Data mining is the discovery of interesting, unexpected or valuable structures in large datasets. Association rule discovery is part of a larger field of study called data mining a field that consists of techniques to automatically find interesting patterns and trends in large collections of data.

4.3 Data Mining Techniques

4.3.1 Classification

Classification is the process of partitioning a given dataset into disjoint classes using a class attribute. The goal of classification is to analyze the training set and to develop an accurate description or model for each class using the attributes presented in the data. Many classifications models have been developed such as neural networks, genetic models, and decision trees etc.

4.3.2 Association Rules

Association rule mining [7] is a data mining technique used to find interesting associations among a large set of data items. A typical application of association rule mining is market-basket analysis. In market-basket analysis, buying habits of customers

are analyzed to find associations between the different items that customers place in their "shopping baskets". The discovery of such associations can help retailers develop marketing and placement strategies as well as plan on logistics for inventory management. Items that are frequently purchased together by customers can be identified. An attempt is made to associate a product "A" with another product "B" so as to infer "whenever A is bought, B is also bought", with high confidence (i.e., the number of times B occurs when A occurs).

4.3.3 Clustering

Clustering is the process of grouping the data into clusters with high intra-cluster similarity and low inter-cluster similarity. A similarity measure needs to be defined and the quality of the cluster, to a large extent, depends on the appropriateness of the similarity measure for the data set or the domain of application. The technique of clustering, for example, can be used to divide the market into distinct groups, so that each group can be targeted with a different strategy. There are several clustering techniques: partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model based methods.

The basic difference between classification and clustering is that classification is a supervised learning method, which assumes predefined class labels, while clustering is an unsupervised learning method that does not assume any knowledge of classes.

4.3.4 Prediction

Prediction techniques are based on some continuous valued attributes. Previous history of the attributes is used to build the model. This technique is commonly used for predicting product sales.

4.3.5 Deviation Analysis

This technique compares current data with previously defined normal values to detect anomalies. Deviation analysis tools are useful in security systems, where authorities can be warned about deviation in resource utilization. From the vast areas of data mining we have used association rules and classification techniques for finding the degree of dependency between the category name and description of Web Services.

4.4 Introduction to Association Rules

Association rule mining makes correlation among items that are grouped into transactions, deducing rules that define relationships between item sets. The rules have a user-stipulated support, confidence, and length. Association rule mining has attracted tremendous attention from data mining researchers and as a result several algorithms have been proposed for it . Let $I = \{i_1, i_2, \ldots, i_m\}$ be the collection of all the items and D be the set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \phi$. They are two terms associated with association rules. These are: Support and Confidence. If the support of itemset {AB} is 30%, it means "30% of all the transactions contain both the itemsets – itemset A and itemset B".

Support of itemset {AB} = Count Of the transactions containing the itemsets A and B
Total Number of Transactions

If the confidence of the rule $A \Rightarrow B$ is 70%, it means "70% of all the transactions that contain itemset A also contain itemset B".

Confidence of the rule $A \Rightarrow B = \dfrac{Support\ A}{Support\ AB}$

An association rule-mining problem is broken down into two steps:

1. Generate all the item combinations (itemsets) whose support is greater than the user specified minimum support. Such sets are called the frequent itemsets and
2. use the identified frequent itemsets to generate the rules that satisfy a user specified confidence. The frequent itemsets generation requires more effort and the rule generation is straightforward.

4.4.1 Association Rule Mining Algorithms

Many algorithms have been devised for finding the relation between different itemsets. Some of them is given below.

4.4.1.1 Apriori Algorithm

The Apriori algorithm [11] is based on the frequent itemsets and rule generation phases. Frequent itemsets are generated in two steps. In the first step all possible combination of items, called the candidate itemset (C_k) is generated. In the second step, support of each candidate itemset is counted and those itemsets that have support values greater than the user-specified minimum support form the frequent itemset (F_k). In this algorithm the database is scanned multiple times and the number of scans cannot be determined in advance. The apriori algorithm is depicted below.

```
F1 = {frequent 1-itemsets}
For (k = 2; Fk-1 ≠ 0; k++) do
Ck = generate (Fk-1)
For all transactions t ∈ D do
Ct = subset (Ck, t)
For all candidates, c ∈ Ct do
c.count++
End for
End for
Fk = {c ∈ Ck | c.count ≥ minsup}
End for
Answer = ∪k {Fk}
```

The problem with the above algorithm is that it generates too many candidates that turn out to be small (or not frequent) resulting in wasted effort.

4.4.1.2 Partition Algorithm

The partition algorithm is a fast and efficient algorithm for mining association rules in large databases. The algorithm differs from the other mining algorithms in terms of the number of passes it makes over the database. The algorithm makes just 2 passes over the input database to generate the rules. The partition algorithm executes in two phases: In the first phase of the algorithm the database is divided into non-overlapping partitions and each of the partitions are mined individually to generate the local frequent itemsets. At the end of the first phase the local frequent itemsets are merged to generate the global candidate itemsets. In the second phase of the algorithm the support is calculated for all the itemsets in the global candidate itemset and the global frequent itemsets are generated

as the set of itemsets, which satisfy the support. The database is scanned completely once in each of the phases of the algorithm. The algorithm can be executed in parallel to utilize the capacity of many processors with each processor generating the rules for a particular set of transactions and merging the frequent itemsets obtained from all the processors.

The Partition algorithm [12] differs from the Apriori algorithm in terms of the number of database scans. The partition algorithm scans the database at most twice. The algorithm is inherently parallel in nature and can be parallelized with minimal communication and synchronization between the processing nodes the algorithm is divided into 2 phases:

i) During the first phase, the database is divided into n non-overlapping partitions and the frequent itemsets for each partition are generated.

ii) In the second phase, all the local large itemsets are merged to form the global candidate itemsets and a second scan of the database is made to generate the final counts.

The partition algorithm is designed to generate rules in parallel and utilize the power of a number of processors.

4.4.1.3 Incremental Mining

Association rules represent an important class of knowledge that can be discovered from data warehouses [13]. As new data is added, previously discovered rules have to be verified and new rules may have to be added to the knowledge base. Changes to the data can also invalidate existing patterns. The task of deriving new rules for data sets that grow incrementally can be done in several ways. Re-executing the algorithms from scratch each time a database is updated can result in excessive computation and I/O. Re-execution is not an efficient process, since it ignores previously discovered rules and repeats the work that has already been done. Incremental mining is a useful technique for discovering new rules as the data distribution patterns change, obviating the need for re-computation of old rules.

4.5 Web Services Classification Process

Web Services Classification is the act of determining a category of a Web Services, from several pre defined categories [24]. The automatic classification is done on the basis of information provided by the WSDL documents.

There are two stages in the classification process:

1. Text preprocessing or text mining
2. Classification

Automatic Web Services classification process uses text mining techniques at the first stage, namely preprocessing, to extract relevant information from a WSDL document. These techniques have been designed for preprocessing textual documentation, operations and arguments accompanying descriptions of Web Services. The process uses a supervised document classifier at the second stage, namely classification. This classifier deduces a sequence of candidate categories for a preprocessed Web Services description.

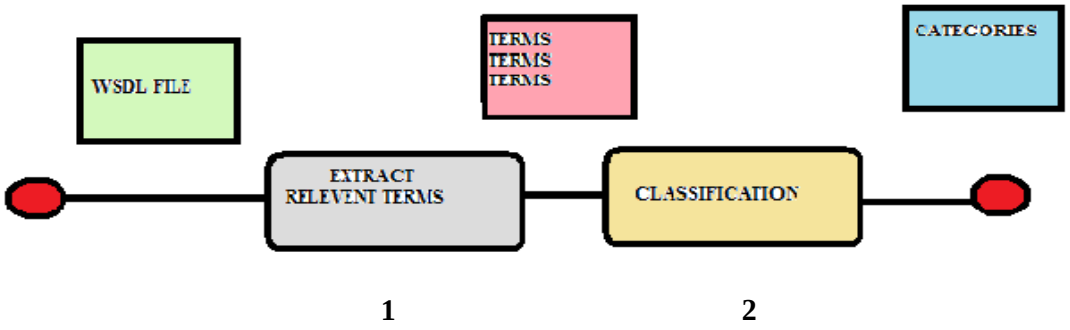

Figure 4.4: Two Step Classification Process

4.5.1 Text Preprocessing for WSDL Documents

Text mining, also known as intelligent text analysis refers to the process of extracting interesting and non-trivial information and knowledge from unstructured text. First the names and comments are extracted then the combined names are split to generate different words. By splitting combined words, process attempts to bridge different naming conventions. In general, developers combine a verb and a noun for denoting the name of an operation, such as getQuote or get quote. Then, every distinct operation and message that follows each naming denomination would be treated as a different word. To bridge different naming conventions, combinations of words are searched and split into verbs and nouns. Combinations of more than two words is also considered, e.g., for "getQuoteFor" the tool dumps "get", "quote" and "for". Table 1 summarizes the rules for splitting combined words.

Notation	Rule	Source	Result
Java Beans	Splits when changing text case.	getZipCode	Get Zip Code
Hungarian	Splits when changing text cases.	ulAccountNum	Ul Account Num

Special symbols	Splits when either "_" or "-" occurs.	get_Quote	Get Quote

Table 4.1: Rules for splitting combined words.

The terms are then filtered to remove the non relevant words, called stop words. Thereafter, stemming is done to reduce terms to their stems. The whole process is explained by the Figure 4.5 shown below.

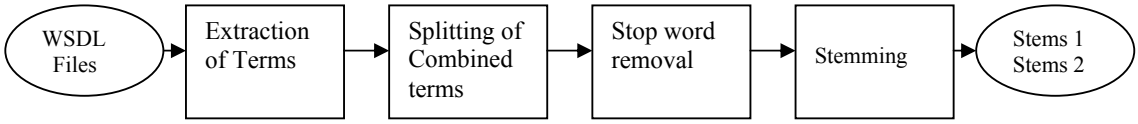

Figure 4.5: Text Mining Process

4.5.2 Techniques for Preprocessing

1. Detagger
2. Tokenizer
3. Stop word removal
4. Stemming

- **Detagger:** Detagger is used for removing all the tags from the XML document. Basically it is used for tag removal but along with that it can also strip off the section of code. There are various tools available online for detagger.
- **Tokenizer:** The stream of characters in a natural language text must be broken up into distinct meaningful units (or tokens) before any language processing beyond the character level can be performed. A token is a categorized block of text. A lexical analyzer processes lexemes to categorize them according to function, giving them meaning. This assignment of meaning is known as tokenization. A token can look like anything; it just needs to be a useful part of the structured text. A tokenizer is not always required, it heavily depends on the following processing steps and how powerful those methods, like stemming methods, are. Semantic approaches should be able to handle the most problems. It needs a good consideration what a tokenizer should process. Otherwise, it might slow down the whole information retrieval process.
- **Stopword removal**: Stop words are the irrelevant terms that do not have any importance in the document classification. Stop word removal allows for the discarding of certain key phrases and words that may otherwise taint the

algorithms (for example, those present as headers or those which represent concepts that are plentiful but irrelevant).

- **Stemming:** Stemming is a technique for the reduction of words into their root . Many words in the English language can be reduced to their base form or stem e.g. **agree**d, **agree**ing, dis**agree**, **agree**ment and dis**agree**ment all belongs to the stem **agree.** Furthermore are names transformed into the stem by removing the"'s". Following a selection of suffixes and prefixes for removal during stemming:

  Suffixes: ly, ness, ion, ize, ant, ent , ic, al , ical, able, ance, ary, ate, ce, y, dom , ed, ee, eer, ence,ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ty,ship, some, ure

  Prefixes: anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tri, ultra, un.

- **TFIDF Weighting:** After extracting the terms from the WSDL document every document will be represented as a vector ~v = (e_0, ..., e_n). Each element e_i represents the importance of a distinct word w_i for that document. This importance will be calculated according to the selected word weighting method, TF-IDF. TF-IDF is a word weighting heuristic that determines that a word is important for a document if it occurs often on it. Instead, words that occur in many documents are rated as less important because of their low inverse document frequency. For each term ti of a document d weight can be calculated by using the formula:

$$\text{Term Weight} = \quad w_i = tf_i * log\left(\frac{D}{df_i}\right)$$

Where

tf_i = term frequency (term counts) or number of times a term i occurs in a document.

df_i = document frequency or number of documents containing term i

D = number of documents in the database [2].

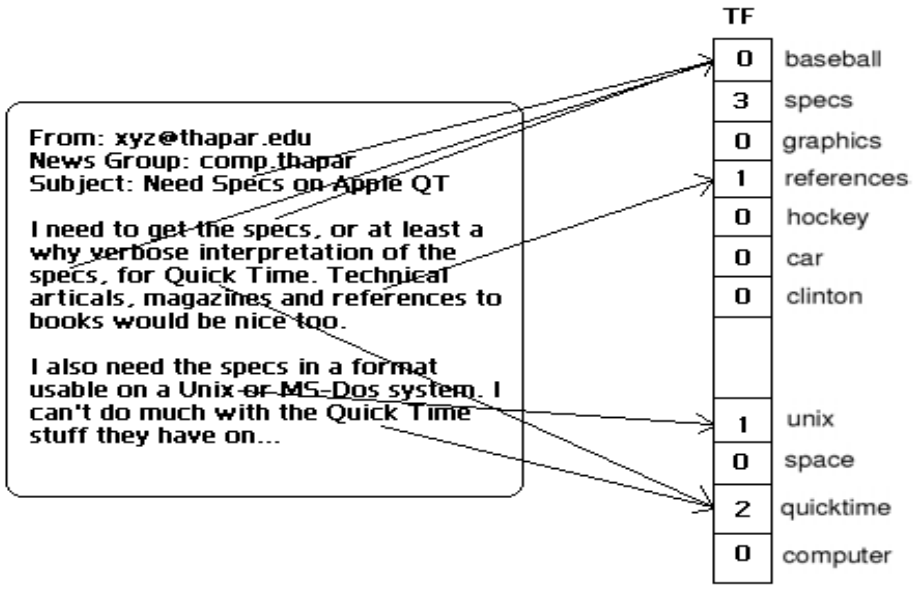

Figure 4.6: Term frequencies with in a document.

4.5.3 WSDL Classification

The classification of services into domain–specific classes can be performed using the any of the method stated below. As mentioned above, automatic service classification helps during:

1. **Service publication** (to classify the new service) and
2. **Service retrieval** (to identify the class(es) where to restrict the focus of the query) [24].

Document classification refers to the process of assigning an electronic document to one or more categories based on its contents [18]. Automatic document classifiers support classification of documents seen as objects characterized by features extracted from their contents. In general, when some external mechanism, such as human feedback, provides information on the correct classification for documents, we talk about supervised document classification. This approach consists of two phases:

1. Training phase.
2. Classification phase.

During the training phase, such a learning system receives a collection of categorized documents and builds a classifier. Then, during the classification phase, this classifier deduces one or more categories for a new document. Automatic classification assumes that:

1. The category of a Web Services depends on its textual comments in its WSDL.
2. Method signatures i.e. the operation name and arguments

It is desirable to measure the dependencies between categories and arguments, i.e., operation interfaces. By finding this dependency we can create category vectors for each category in the pre defined set. Figure below shows an example of the training set data based upon which the category for the new document can be inferred.

	Training Set					**Test Set**			
	d_1	…	…	d_g		d_{g+1}	…	…	d_s
c_1	ca_{11}	…	…	ca_{1g}		$ca_{1(g+1)}$	…	…	ca_{1s}
…	…	…	…	…		…	…	…	…
c_i	ca_{i1}	…	…	ca_{ig}		$ca_{i(g+1)}$	…	…	ca_{is}
…	…	…	…	…		…	…	…	…
c_m	ca_{m1}	…	…	ca_{mg}		$ca_{m(g+1)}$	…	…	ca_{ms}

Figure 4.7: Training Set and Test Set [15].

ca_{ij} will be equal to 1 if document d_j belongs to the category c_i,

ca_{ij} will be equal to 0 if document d_j does not belongs to the category c_i,

A document d_j is often referred to as a positive example of c_i if $ca_{ij} = 1$, a negative example of c_i if $ca_{ij} = 0$.The classifier is first trained with the training set and then each new document can be categorized based upon the cosine similarity between the new weight vector of the new document and the pre defined categories. The document is assigned to the category with which its vector has highest cosine angle.

4.6 Categorization Algorithms

A variety of different algorithms for text categorization have been developed. We list here the main approaches:

4.6.1 Naïve Bayes

The Naïve Bayes categorization approach is a simple probabilistic technique. The probability that a particular document belongs to given class is determined by relying on the assumption that word distributions are independent variables, i.e. that the presence of one word has no effect on the distribution or presence of other words in the same document, an assumption that is most likely not seen in practice. From estimated probabilities that words belong to different categories, the probabilities that documents belong to various categories are determined .Documents are often represented by vectors that account only for the presence or absence of words, not for their frequency in each document. Overall, the Naïve Bayes algorithm is simple to implement, but usually outperformed by the other techniques explained below [25].

4.6.2 Rocchio

The Rocchio technique builds for each category a single prototypical document. The category profile consists of a weighted list of words or terms formed from the word distribution within the category. To decide which categories a new document belongs to, its word distribution is compared to those of the prototypical category documents. When the similarity is high enough, the new document is assigned to the category in question. While extremely simple to implement, difficulties arise with this algorithm when the documents in a category are bunched in several disjoint groups. The algorithm then compares a new document to an average prototypical document that may be quite different from those in each bunch of the category, leading to obvious categorization errors.

Refinements to improve the Rocchio algorithm consist in including negative training examples, taken as documents just outside each category, and using them as negative indicators when computing prototypical document vectors [26].

4.6.3 Neural Networks

An artificial neural network consists of a network of many simple units, usually positioned in successive layers. Communication channels that carry numeric data connect

the units, with varying connection strengths. A network layer receives input, in the form of a collection of terms and weights representing a document, intermediate layers process the weights, and an output layer suggests a relevant category. A large number of variations exist in this architecture [17].

4.6.4 SVM (Support Vector Machine)

Very accurate text classifiers can be learned automatically from training examples using simple linear SVMs. The SMO method for learning linear SVMs is quite efficient even for large text classification problems. SVMs also appear to be robust to many details of preprocessing [28].

4.6.5 Decision Rules

Decision rules algorithms classify a document by following a set of classification directives or rules. The rules indicate when a word, or a collection of words, or the absence of a word, is a good indicator that the document belongs to a given category. The rules may be combined in the form of a complex decision tree.

Such category decision rules can be learned automatically, by examining which words discriminate between categories, or experts can formulate them manually. In this respect, this approach is unique as it allows document categorization without the availability of a training set of pre-classified documents, provided experts can formulate the categorization task accurately.

When no explicit knowledge about categorization rules is available, machine-learning algorithms that rely of word statistics, such as those presented in the preceding paragraphs are eminently suitable [29].

4.7 Tools Used

During implementation phase following two tools were used:

4.7.1 WEKA

The Weka workbench is a collection of state-of-the-art machine learning algorithms and data preprocessing tools. It provides extensive support for the whole process to

implement data mining, including preparing the input data, evaluating learning schemes statistically, and visualizing both the input data and the result of learning.

Weka has three principal **advantages** over most other data mining software:

1. It is open source, maintainable, and modifiable.
2. It provides a wealth of state-of-the-art machine learning algorithms that can be deployed on any given problem.
3. It is fully implemented in Java and runs on almost any platform.

The main **disadvantage** is that

1. Most of the functionality is only applicable if all data is held in main memory. However, for most of the methods the amount of available memory imposes a limit on the data size, which restricts application to small or medium-sized datasets. If larger datasets are to be processed, some form of sub sampling is generally required.
2. A second disadvantage is the flip side of portability: a Java implementation is generally somewhat slower than an equivalent in C/C++.

4.7.1.1 Methods and algorithms in WEKA

Weka contains a comprehensive set of useful algorithms for data mining tasks. These include tools for data engineering (called "filters"), algorithms for attribute selection, clustering, association rule learning, classification and regression. In the following subsections the most important algorithms in each category are listed:

- **Classification.** Implementations of almost all main-stream classification algorithms are included. Bayesian methods include naive Bayes, complement naive Bayes, multinomial naive Bayes, Bayesian networks, and AODE, decision tree learners, decision stumps,ID3, a C4.5 clone called "J48,"decision trees, random trees, OneR, Ripper called JRip, decision tables, single conjunctive rules, and Prism.
- **Regression.** There are implementations of many regression schemes. They include multiple and simple linear regression, pace regression, a multi-layer perceptron, support vector regression, locally-weighted learning, decision stumps, regression and model trees (M5) and rules (M5rules).

- **Clustering.** At present, only a few standard clustering algorithms are included: KMeans, EM for naive Bayes models, farthest-first clustering, and Cobweb. This list is likely to grow in the near future.
- **Association rule learning.** The standard algorithm for association rule induction is Apriori, which is implemented in the workbench, and there is also Tertius, which can extract first-order rules.
- **Attribute selection.** Both wrapper and filter approaches to attribute selection are supported. A wide range of filtering criteria are implemented, including correlation-based feature selection, the chi-square statistic, gain ratio, information gain, symmetric uncertainty, and a support vector machine-based criterion Filters [21].

4.7.2 Rapidminer

Rapid Miner uses XML (extensible Markup Language), a widely used language well suited for describing structured objects, to describe the operator trees modeling KD processes The graphical user interface and the XML based scripting language turn Rapid-Miner into an IDE and interpreter for machine learning and data mining. Rapid Miner provides more than 400 operators including:

- **Machine learning algorithms:** a huge number of learning schemes for regression and classification tasks including support vector machines (SVM),decision tree and rule learners, lazy learners, Bayesian learners, and Logistic learners. Several algorithms for association rule mining and clustering are also part of Rapid Miner. Furthermore, we added several meta learning schemes including Bayesian Boosting.
- **Data preprocessing operators:** These include discretization, example and feature filtering, missing and infinite value replenishment, normalization, removal of useless features, sampling, dimensionality reduction, and more.
- **Feature operators:** selection algorithms like forward selection, backward elimination, and several genetic algorithms, operators for feature extraction from time series, feature weighting, feature relevance, and generation of new features.
- **Meta operators:** optimization operators for process design, e.g. example set iterations or several parameter optimization schemes.
- **Performance evaluation:** cross-validation and other evaluation schemes, several performance criteria for classification and regression, operators for parameter optimization in enclosed operators or operator chains.

- **Visualization:** operators for logging and presenting results. Create online 2D and 3D plots of your data, learned models and other process results.
- **In-** and output: flexible operators for data in- and output, support of several File formats including
    1. arff
    2. C4.5
    3. csv
    4. bibtex
    5. dBase
    6. Reading directly from databases [30].

Chapter 5

Implementation and Results

In order to implement association rules and classification algorithms on our data set WEKA and Rapid Miner toolkits are used. We tried evaluate the performance of a classifier based upon association rules and to analyze whether this gives better results than the primitive classification algorithms.

5.1 Preprocessing

Our data set contains words extracted from plain text description for each service and the WSDL document associated with each service.

Data sets: we used here a collection of 165 Web services whose WSDL documents from downloaded from different UDDI repositories.

1. www.Xmethods.com
2. www.SALCentral.com
3. www.service-repository.com

All the WSDL files were preprocessed using the following steps:

1. First we pulled out argument declarations and comments from each WSDL document.
2. Tokens were generated using the tokenizer.
3. Stop words were removed from the file. A text file “stopword.txt” containing list of stop words was given as a input to the toolkit. Based upon this text file all stop words were removed.
4. Stemming was done to bring the words to their base words. porter stemmer was used. steps 1 to 4 are shown in Figure 5.1 shown below.

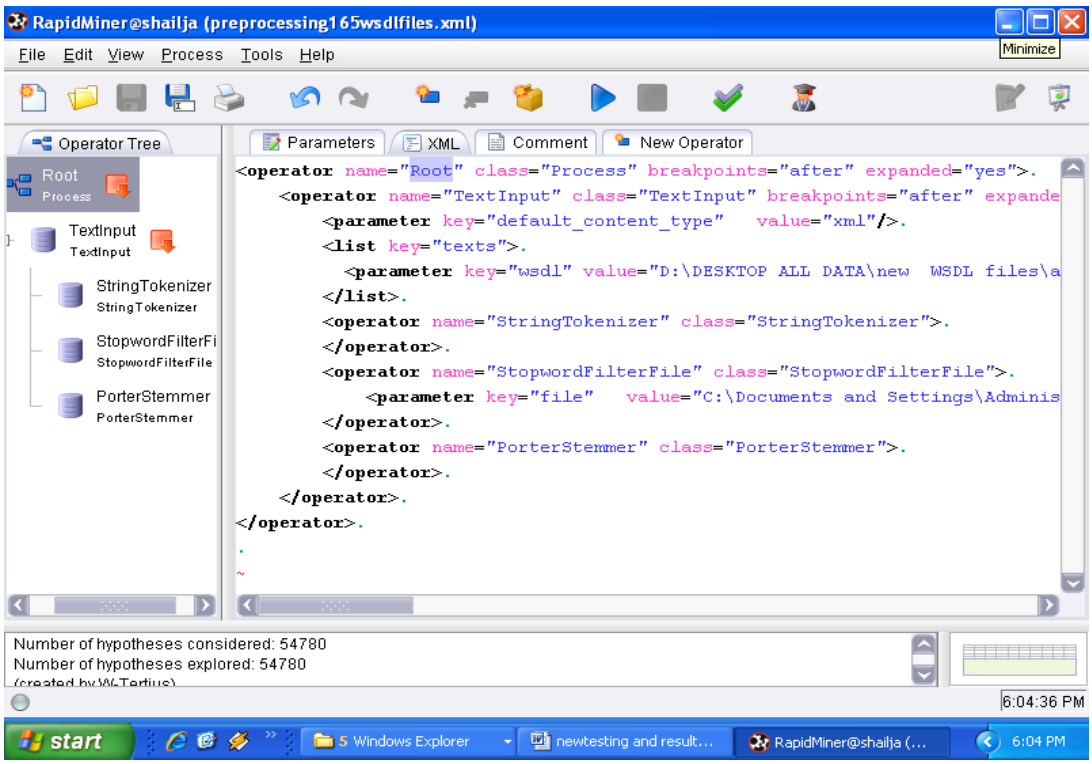

Figure 5.1: Preprocessing Of WSDL files.

5. TF-IDF was generated as shown in Figure 5.2 and Figure 5.3 from this TF-IDF matrix the words with high importance were extracted out.

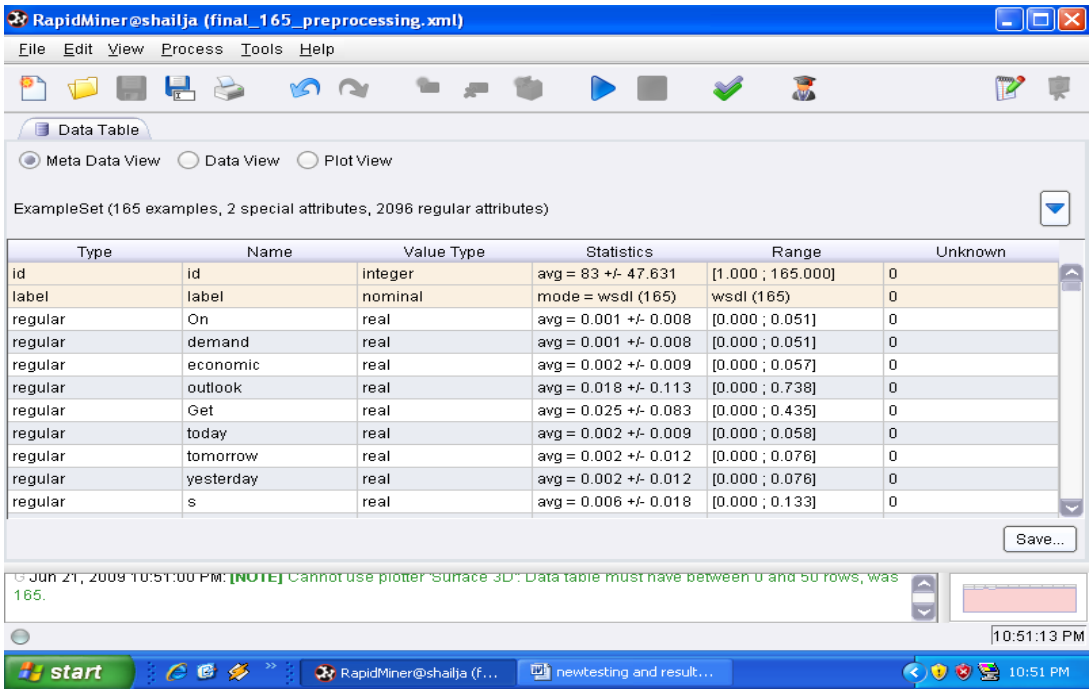

Figure 5.2: Meta data view of files.

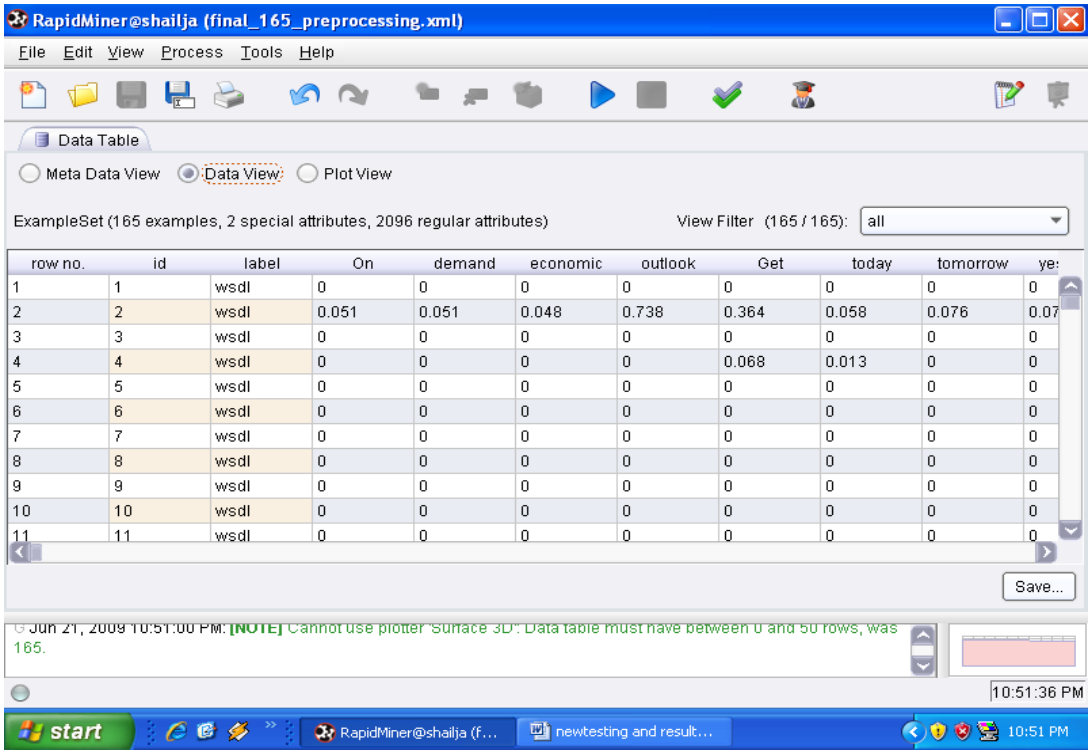

Figure 5.3: TF-IDF Matrix

First, we measured the dependencies between categories and WSDL contents. We have assessed that an argument is more significant to a category, if it has a high importance in the given category but a low importance in the whole collection of categories. Here, "importance" refers to the TF-IDF value for an argument. We have verified our hypothesis on a subset of the Web services collection. This collection is composed of 165 hand-classified Web services, as shown in Table 5.1.

Category of Web Services	**No. of Web Services**
Country (a)	62
Communication (b)	38
Business (c)	51
Finance (d)	14

Table 5.1: Categories for Web Services.

6. Once TF-IDF matrix is calculated then .comma separated files (.csv) and attribute relation file format (.arff) files are prepared whose first column represent column represents service parameters and last column represents the class. This data set is loaded to the WEKA and Rapid Miner Toolkit for further application of association rules.

5.2 Association Rules generation

Association rule Implementation in WEKA.

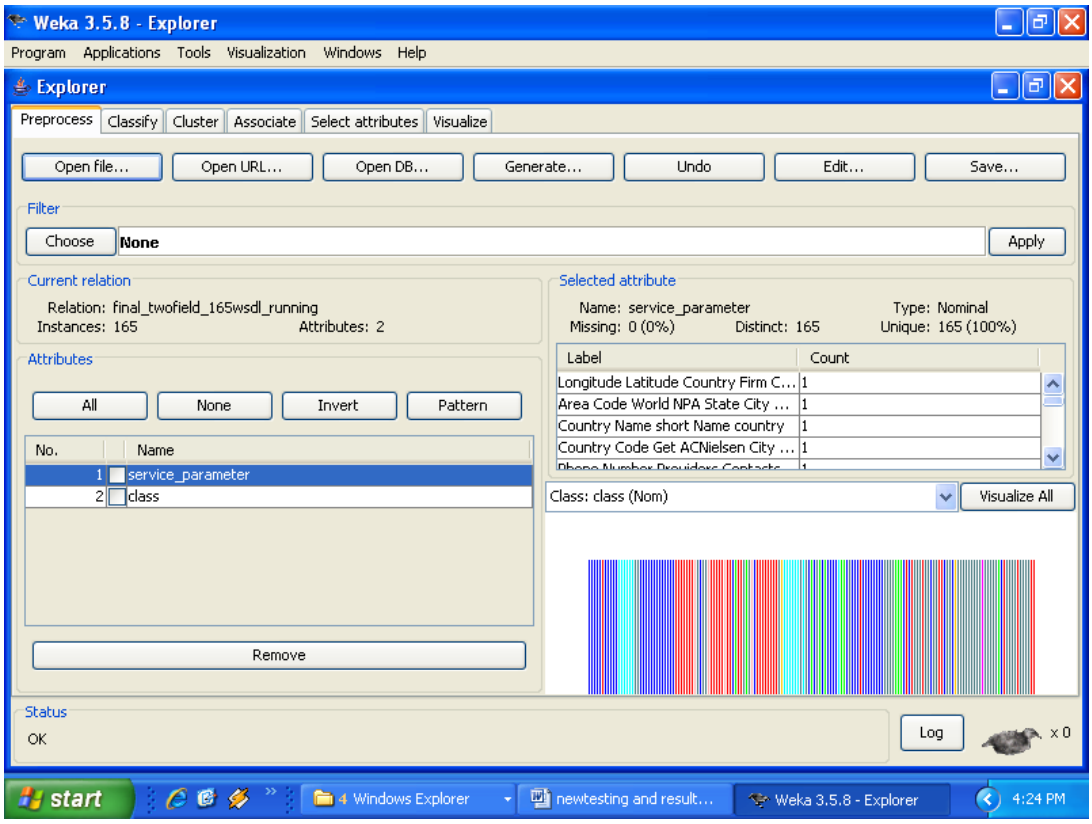

Figure 5.4: Loading data set to WEKA.

After the file has been preprocessed we tried to find out interface patterns within category related services by using Association Rules, specifically, the Tertius and PridictiveApriori algorithm from Weka and Rapid miner. The results using WEKA and Rapid Miner toolkits are shown in Figure 5.6, Figure 5.8 and Figure 5.10.

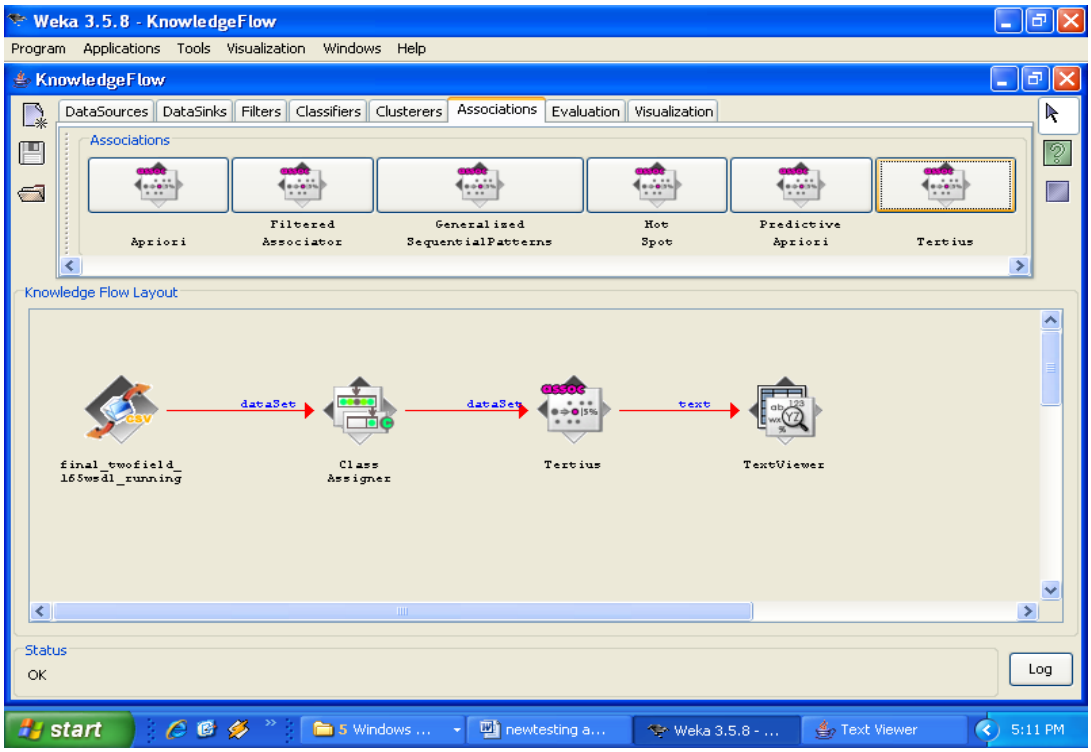

Figure 5.5: Knowledge flow set up for Association Rules

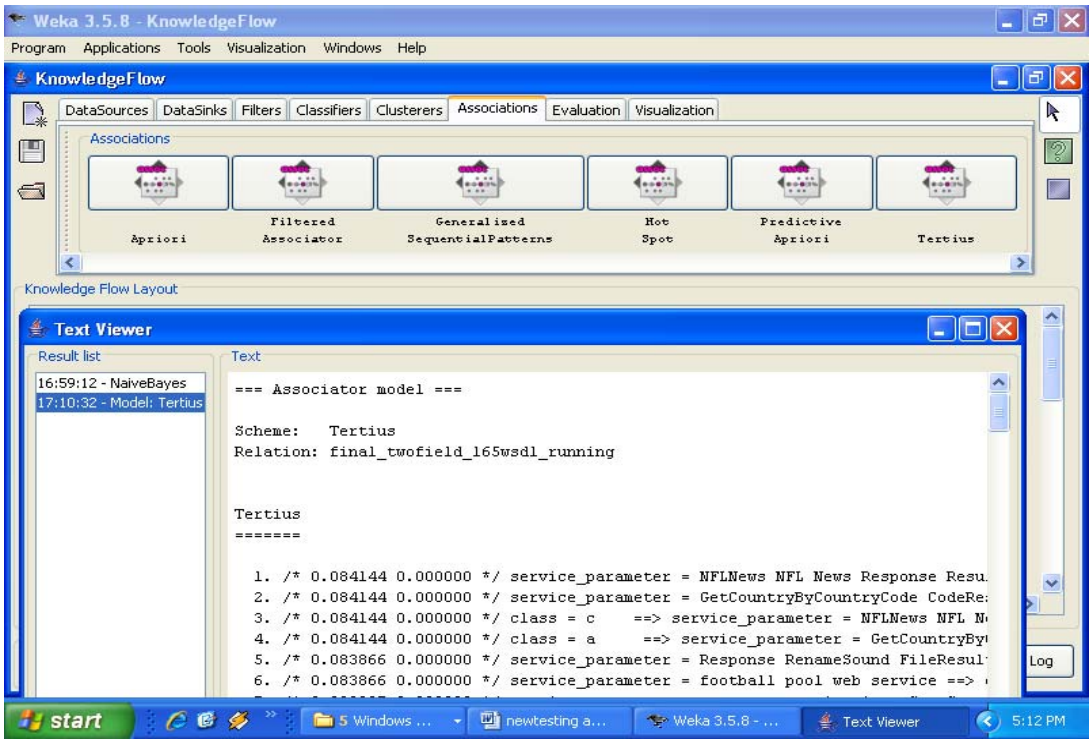

Figure 5.6: Association Rules generation with Tertius algorithm

Some discovered rules are presented below. The number on the right of each rule is a measure of how much evidence exists for that rule (i.e. confirmation value):

1. service_parameter = Stock Quote ServiceInfo ==> class = d {0.080467}
2. service_parameter = Quote Stock StockQuote ==> class = d{0.080467 }
3. service_parameter = City Country Codes Response ==> class = a{0.069187 }
4. service_parameter = Phone Number validate Verify ==> class = b{0.074537}

Association rules using predictiveapriori algorithm:

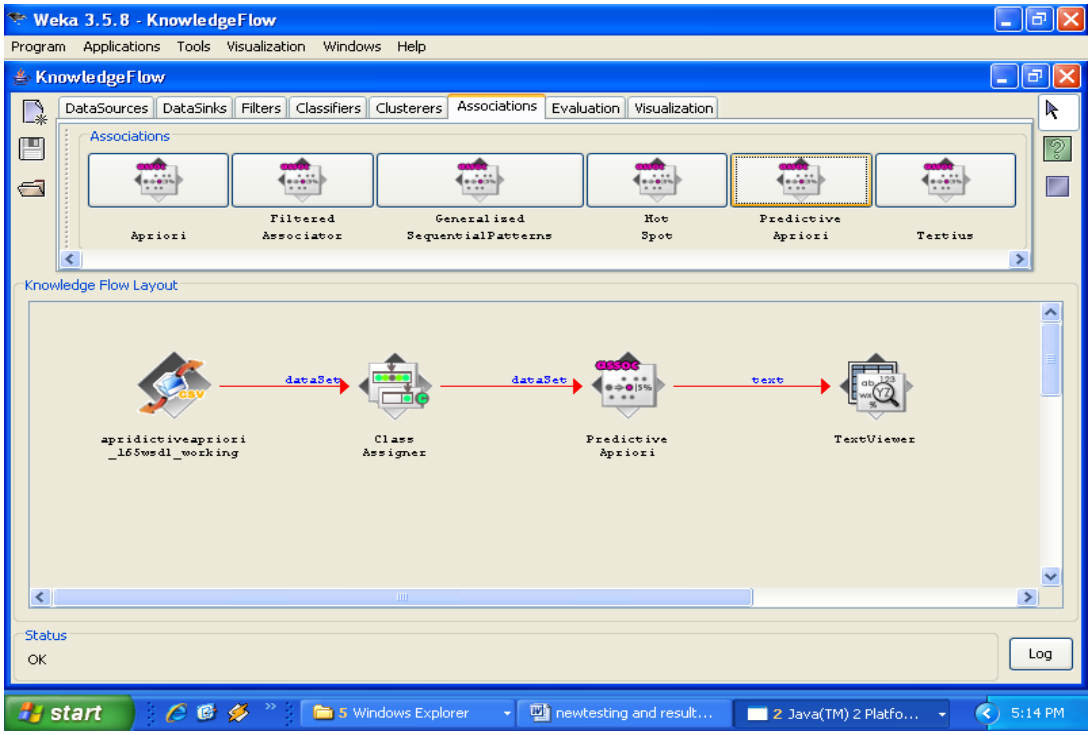

Figure 5.7: Set up for PredictiveApriori algorithm

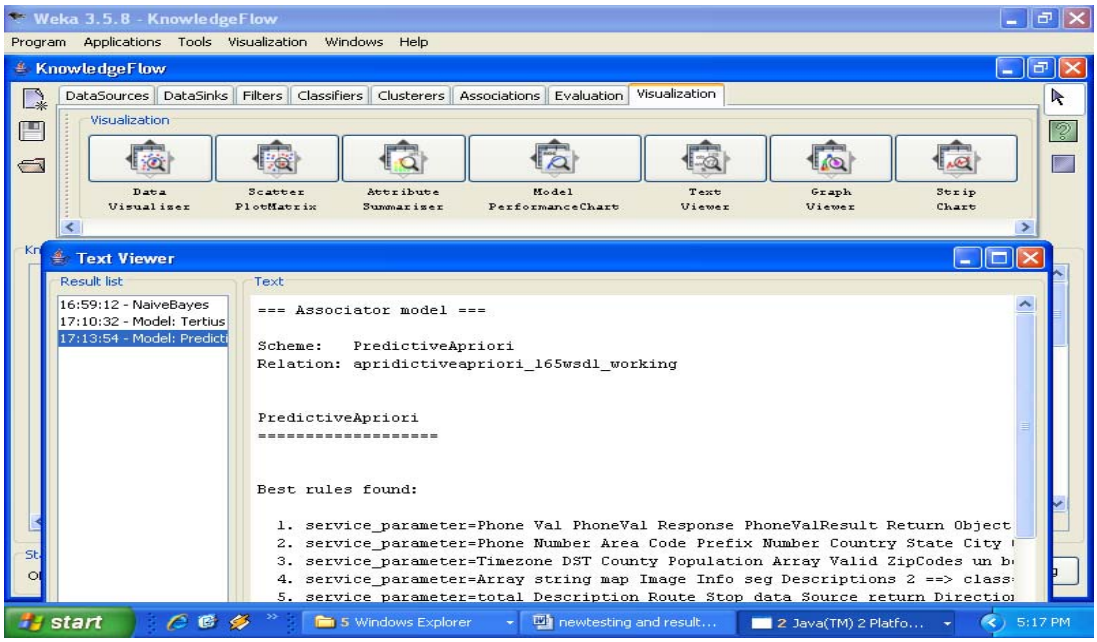

Figure 5.8: Association rules generated using PridictiveApriori algorithm

Some discovered rules are presented below. The number on the right of each rule is a measure of how much evidence exists for that rule (i.e. confirmation value):

Rules generated form predictive Apriori algorithms are:

1. Class=a ==> service_parameter=Phone Number Area Code Prefix Number Country State City County acc:(0.05141)
2. Class=b ==> service_parameter=Phone Val Response Return Object Status Description acc:(0.06949)

Association rules implementation in Rapid Miner:

Number of rules generated in Rapid Miner was many times more than the rules generated by WEKA toolkit. Many rules were redundant and the confirmative value was very low of the order around 0.005000.

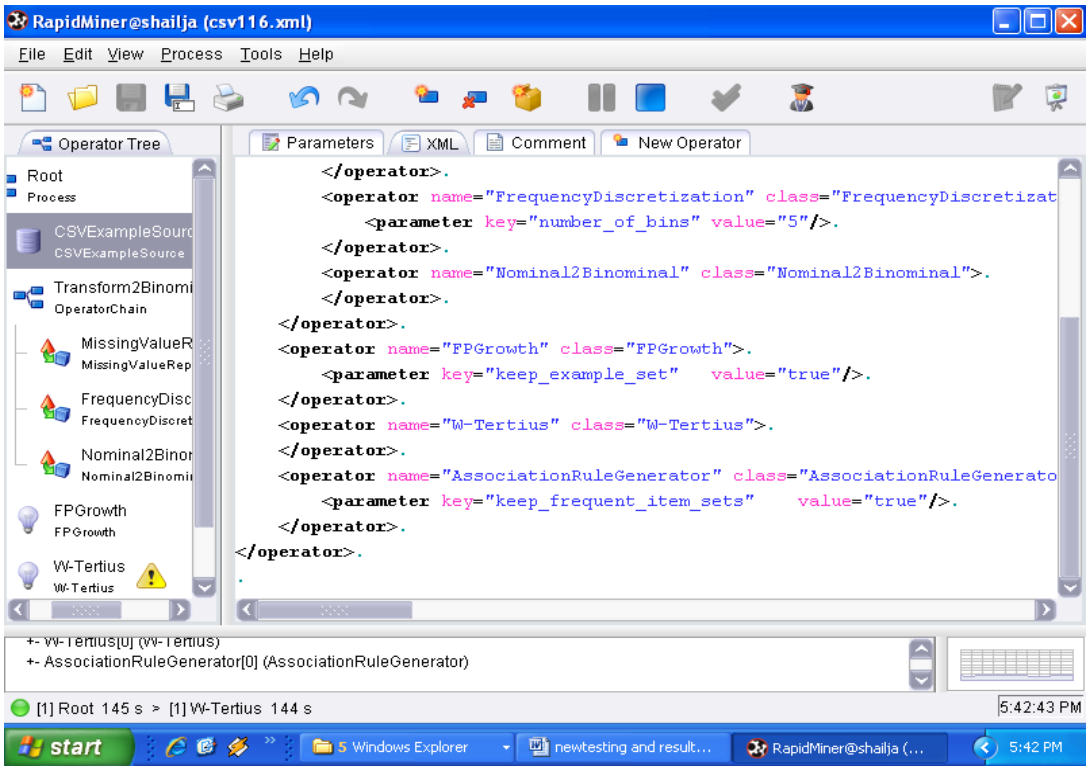

Figure 5.9: Set up in Rapid Miner for Tertius algorithm.

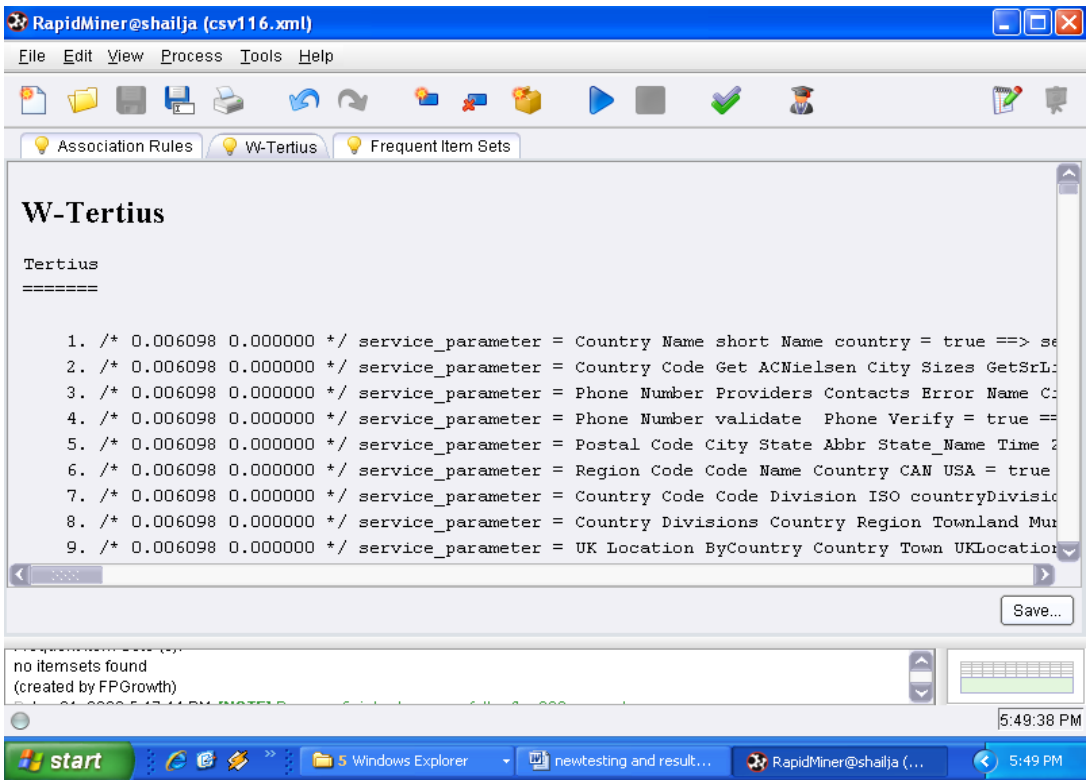

Figure 5.10: Rules generation in Rapid Miner.

5.3 Classification based upon Naïve Bayes classifier on the same data set:

Process of Naïve bayes classification.

1. .csv or .arff file is loaded to the WEKA knowledge flow through arffLoader or csvLoader.
2. The classiassigner object is used for specifying which column we want to use as class.
3. Naive bayes algorithm is applied to the data set
4. Results are viewed in the Text viewer object.

Output for applying the Naïve bayes algorithm to the dataset for Web Services is shown in the Figure below.

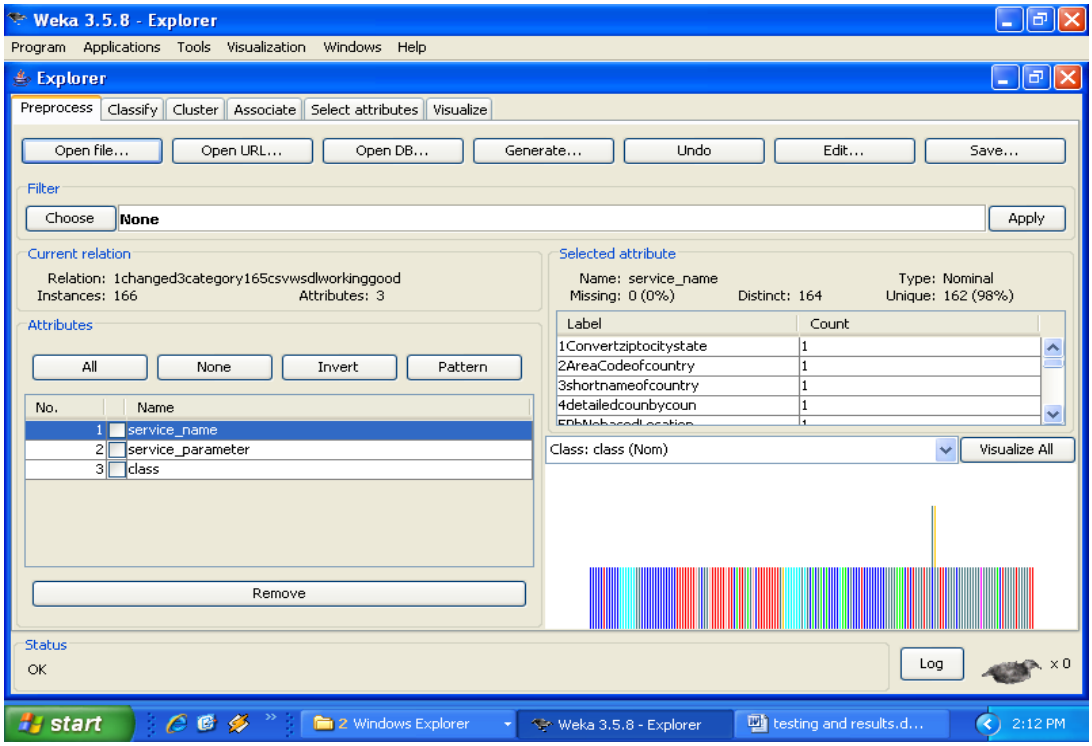

Figure 5.11: Data loaded in WEKA for Classification

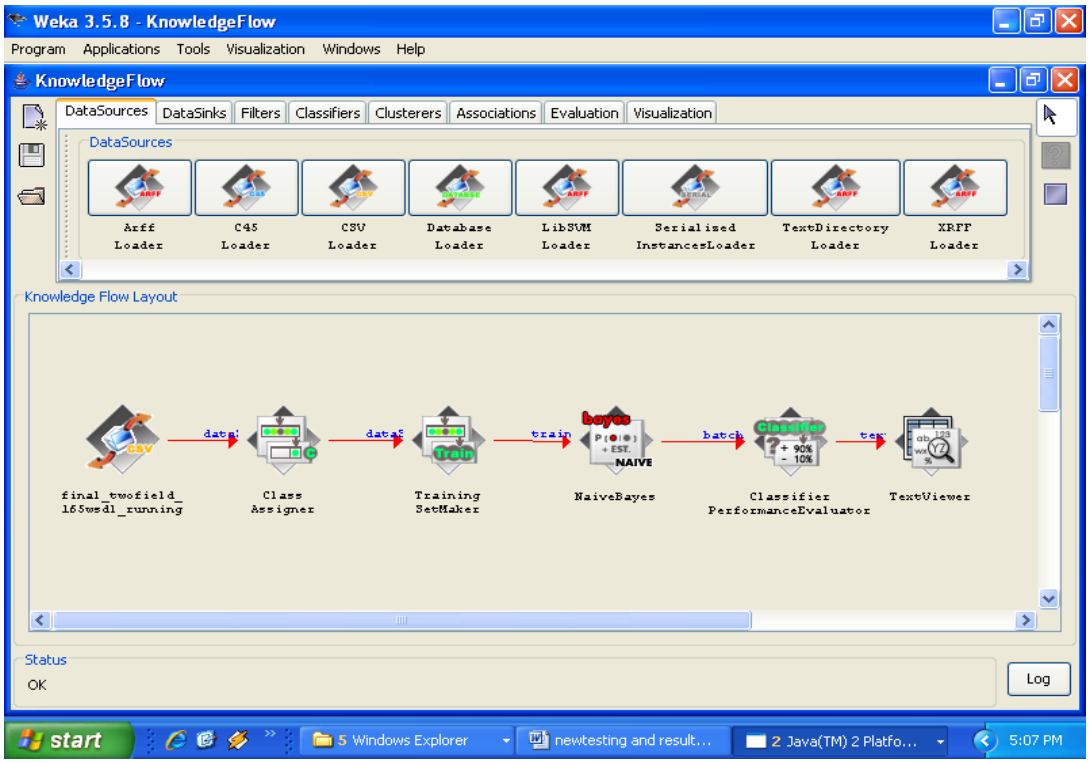

Figure 5.12: Set up for Naïve Bayes Classification

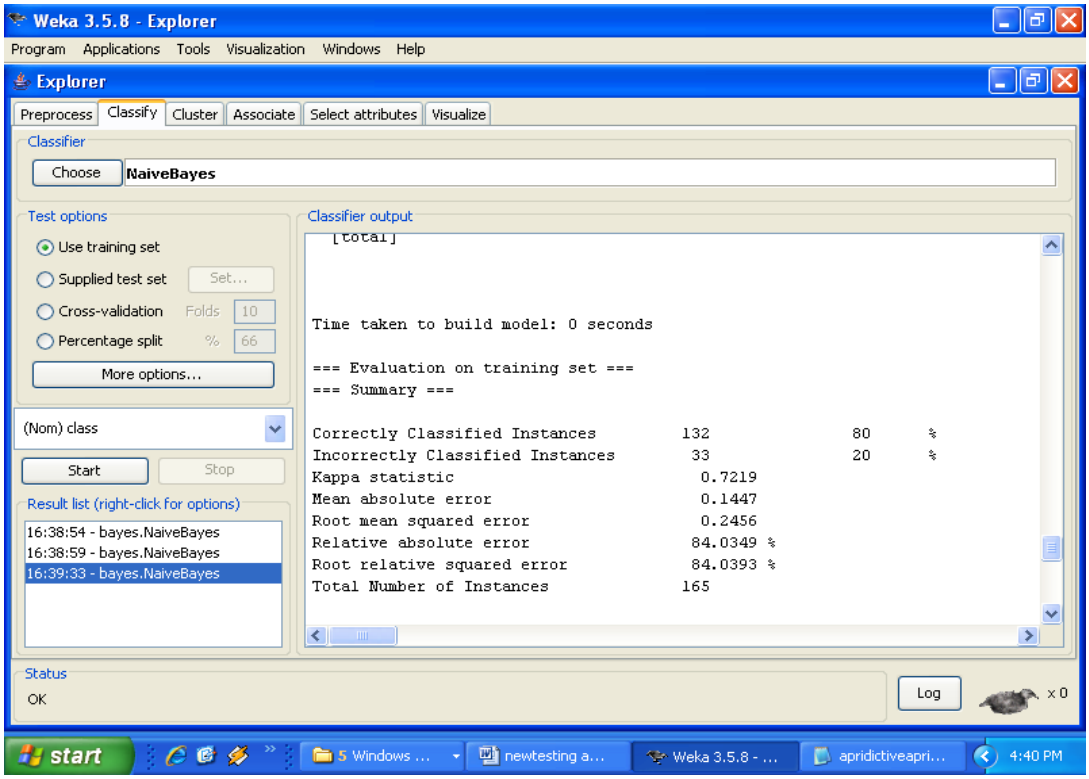

Figure 5.13: Classification Results for Naïve Bayes.

5.4 Analysis and Results

The results from our implementations show that although there was some degree of dependency between category and contents of WSDL but this degree of dependency was far away for any pattern making. The number of rules that we got from our implementation were not sufficient for becoming the basis of a classifier. We can not make a classifier based upon these rules as the resulted confirmative value for these rules from both the algorithms are found to be very low. However, on the basis of the data set we observe that the cause of these results is the variety of argument naming styles used by different developers of web services. These figures show that confirmative values of every association rule was found to be less than .10, which is very low to become the basis of classification.

Then in the second implementation, we applied Naïve Bayes classifier to our data set. When compared to previous results with association rules, the new results appear to be significantly better with a high accuracy. Thus we can be assured that the results are significantly improved using the primitive classification algorithms than the classification based on association rules.

Chapter 6

Conclusion and Future Scope

One of the factors hindering the adoption of Web Services as broadly shared and reused technology across application, enterprise, and community boundaries as one may expect is that manually assigning a proper category for a Web Services description is very difficult. Recent research in text mining and machine learning have shown significant improvement in automatic classification and labeling of documents. We proposed automatic classification system based upon the WSDL descriptions that combines text mining and machine learning techniques for classifying Web Services. Our implementation tried to find out the association relation between category of Web Services and WSDL descriptions and devise rules for automatic classification system for the Web Services. The results from our first implementation (applying association rules on Web Services) have shown that although there were some degrees of dependency between the category of a service and terms extracted from description document of Web Services i.e. WSDL file, but still the discovered rules were far away from becoming the bases of a classification system, mainly due to the variety of argument names that Web Services can employ. On the basis of our implementation it was observed that importance of an argument to a particular category increases proportionally to the number of times an argument appears in the services of this particular category, but this importance decreased if an argument is common in the whole collection.

We conducted a second implementation, with the same data-set, to measure the degree of dependency between the comments and method signature of a Web Services and the category this service belongs to. We applied Naïve Bayes classifier on the same data set and when compared to previous results with association rules, the new results appear to be significantly better with a high accuracy. The achieved accuracy of classification model may vary depending on the classifier selected. The performance of the classification system is related to many factors, such as data set, feature selection methods, classification methods and more. Although we have analyzed relation with a well known set of Web Services, the reported results may vary with different data-sets. It was also found that the accuracy of classification system improves with the increasing set of services in each category. To sum up, we have shown that classification algorithms like

naïve bayes are better methods for automatically classifying Web Services than the techniques based upon association rules.

In future we are planning to add semantic support to the Web Services classification system so that intelligent publishing as well as search of Web Services can be implemented. Further we are also working to devise a tool which extracts only method signatures from WSDL document automatically to improve the classification accuracy.

References

[1] S. Robak and B. Franczyk, “Modeling Web Services Variability with Feature Diagrams”, Proceedings of Web-Services, and Database Systems, pages 120-128, 2002.

[2] Dashofy, E. M. van der Hoek, A. Taylor, “A highly-extensible, XML-based architecture description language Software Architecture”, 2001, Proceedings of Working IEEE/IFIP Conference on 2001, Amsterdam, Netherlands.

[3] Curbera, F. Duftler, M. Khalaf, R. Nagy, W. Mukhi, N. Weerawarana, “Unraveling the Web Services, Web: an introduction to SOAP, WSDL, and UDDI”, Internet Computing, IEEE, mar/apr 2002, Volume: 6, issue:2.

[4] World Wide Web Consortium (W3C), “Web Services Description Language (WSDL) 1.1”, http://www.w3.org/TR/wsdl/, 2001.

[5] UDDI.org. UDDI Technical White Paper, http://www.uddi.org/pubs/Iru-UDDI-Technical-White-Paper.pdf, 2000.

[6] Cios, K.J. Pedrycz, W. Swiniarsk, “Data Mining Methods for Knowledge Discovery Neural Networks”, IEEE transactions Nov 98, volume: 9, issue: 6.

[7] R .Agrawal, T. Imielinski, A. Swami, “Mining Association Rules between Sets of Items in Large Databases", Proc. SIGMOD Conference, 1993.

[8] Marti A. Hearst, “Untangling text data mining”, In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, pages 3–10, Morristown, NJ, USA, 1999.

[9] Fabrizio Sebastiani, “Machine learning in automated text categorization”, ACM Computing Surveys, 34(1):1–47, 2002.

[10] Bernd Klein, “Text Categorization or Classification”, http://www.bklein.de/textclassificati-on .php.

[11] Thomas, S. and S. Chakravarthy, “Incremental Mining of Constrained Associations”, Proc. of the 7th Intl. Conf. of High Performance Computing (HiPC), 2000.

[12]. Shenoy, P., “Turbo-charging Vertical Mining of Large Databases”, ACM SIGMOD Int'l Conference on Management of Data, 2000, Dallas.

[13]. Thuraisingham, B., “A Primer for Understanding and Applying Data Mining”, IEEE, 2000. Vol. 2, No.1: p. 28-31.

[14] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, “Web Services – Concepts, Architectures and Applications”, Springer Verlag, Berlin Heidelberg, 2004.

[15] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma, “METEOR-S Web Services annotation framework”, Proc. of the 13th international conference on WWW. ACM Press, 2004.

[16] Nicole Oldham, Christopher Thomas, Amit P. Sheth, and Kunal Verma, "METEOR-S Web Services annotation framework with machine learning classification", Semantic Web Services and Web Process Composition, Volume 3387 of LNCS, pages 137–146, San Diego, CA, USA, 2004, Springer.

[17] Andreas Heß, Eddie Johnston, and Nicholas Kushmerick, "ASSAM: A tool for semiautomatically annotating semantic Web Services", CSCWD 2008, 12th International Conference on Web Technologies, pages 470–475.

[18] Suman Saha, C. A. Murthy and Sankar K. Pal, "Classification of Web Services Using Tensor Space Model and Rough Ensemble Classifier", 2008.

[19] Atif Alamri, Mohamad Eid, Abdulmotaleb EI Saddik, "Classification of the state-of-the-art dynamic Web Services composition techniques", International Journal of Web and Grid Services 2006, Volume 2, No. 2, Pages 148-166.

[20] Microsoft UDDI Business Registry Node, http://uddi.microsoft.com/inquire, 2004.

[21] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten, "WEKA A Machine Learning Workbench for Data Mining", Pages 1305-1314, Berlin: Springer, 2005.

[22] Marco Crasso, Alejandro Zunino and Marcelo Campo, "AWSC: An approach to Web Services classification based on machine learning techniques", CONICET, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No 37 (2008), Pages 25-36.

[23] http://www.javapassion.com/webservices/WSDLBasics_speakernoted.pdf

[24] Marcello Bruno, Gerardo Canfora, Massimiliano Di Penta, and Rita Scognamiglio, "An Approach to support Web Services Classification and Annotation", In Proc. the IEEE International Conference on Web Technologies, 29 March-1 April 2005 Pages 138 – 143.

[25] C. J. Fall, K. Benzineb, "Literature survey: Issues to be considered in the automatic classification of patents", Volume 1.0, CLAIMS, WIPO, Geneva, Switzerland, Oct 2002.

[26] Robert E. Schapire, Yoram Singer, Amit Singhal, "Boosting and Rocchio Applied to Text Filtering", SIGIR'98, Melbourne, Austrailia.

[27] Miguel E. Ruiz, Padmini Srinivasan, "Hierarchical Text Categorization using Neural Networks", Kluwer Academic Publishers, Pages 87-108, Volume 5, Issue 1, 2002.

[28] Susan Dumais, "Using SVMs for Text Categorization", Decision Theory and Adaptive Systems Group, Microsoft Research.at http://research.microsoft.com/users/sdumais/ieee98-tc.doc.

[29] Chidanand Apte and Fred Damerau, "Automated Learning of Decision Text Categorization", ACM Computing Surveys, Mar 2002.

[30] Rapid-I GmbH Stockumer Str. 475 44227 Dortmund,Germanyhttp://www.rapidminer.com/

Appendix

1. WSDL for Free FAX service

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="FreeFaxService"
    targetNamespace="http://www.OneOutBox.com/wsdl/FreeFaxService.wsdl"
    xmlns:tns="http://www.OneOutBox.com/wsdl/FreeFaxService.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  <documentation>Further details at www.OneOutBox.com</documentation>
- <message name="freeFaxRequest">
- <part name="Sender" type="xsd:string">
  <documentation>Your email address, so that delivery notifices can be forwarded
    to your attention. Not used for other purposes.</documentation>
    </part>
- <part name="ToNum" type="xsd:string">
  <documentation>The international dialing code of the recipient FAX machine.
    e.g. USA dial 1+areacode+number. Non-numerics within the string are
    ignored.</documentation>
    </part>
- <part name="Name" type="xsd:string">
  <documentation>Delivery information at the destination, such as name and
    mailstop. May include spaces (or _) and RETURN (or /)</documentation>
    </part>
- <part name="Text" type="xsd:string">
  <documentation>The contents of the FAX to be delivered. Will be formatted
    roughly 80 characters wide on the page. No limit.</documentation>
    </part>
    </message>
- <message name="freeFaxResponse">
- <part name="return" type="xsd:string">
  <documentation>The return code is a tracking number. Not used within the Free
    FAX service, but available for tracking purposes in the commercial
    ProFAX.</documentation>
    </part>
    </message>
- <portType name="FreeFaxPortType">
- <operation name="SendFreeFAX">
  <input message="tns:freeFaxRequest" />
  <output message="tns:freeFaxResponse" />
    </operation>
    </portType>
- <binding name="FreeFaxBinding" type="tns:FreeFaxPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="SendFreeFAX">
  <soap:operation soapAction="urn:OneOutBox#SendFreeFAX" />
- <input>
  <soap:body use="encoded" namespace="urn:OneOutBox"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

```
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:OneOutBox"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    </operation>
    </binding>
- <service name="FreeFaxService">
  <documentation>Provides a Web Services interface to free worldwide FAX
    transmission service, provided by 1outbox
    (www.1outbox.com).</documentation>
- <port name="FreeFaxPort" binding="tns:FreeFaxBinding">
  <soap:address location="http://www.OneOutBox.com:80/cgi-
    bin/soap/outbox.cgi" />
    </port>
    </service>
    </definitions>
```

2. WSDL for send sms web service

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:sms"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:sms">
- <types>
- <xsd:schema targetNamespace="urn:sms">
  <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
  <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </xsd:schema>
    </types>
- <message name="sendSMSToManyRequest">
  <part name="uid" type="xsd:string" />
  <part name="pwd" type="xsd:string" />
  <part name="phone" type="xsd:string" />
  <part name="msg" type="xsd:string" />
    </message>
- <message name="sendSMSToManyResponse">
  <part name="status" type="xsd:string" />
    </message>
- <portType name="SendSMSPortType">
- <operation name="sendSMSToMany">
  <documentation>Sends the same SMS to multiple phone numbers. Give your 10
    digit phone number for user ID. Separate each phone number with a
    semicolon(';').</documentation>
  <input message="tns:sendSMSToManyRequest" />
  <output message="tns:sendSMSToManyResponse" />
    </operation>
    </portType>
- <binding name="SendSMSBinding" type="tns:SendSMSPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="sendSMSToMany">
  <soap:operation soapAction="urn:sms#SendSMSToMany" style="rpc" />
- <input>
  <soap:body use="encoded" namespace="urn:SendSMSToMany"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:SendSMSToMany"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    </operation>
    </binding>
- <service name="SendSMS">
- <port name="SendSMSPort" binding="tns:SendSMSBinding">
  <soap:address location="http://www.aswinanand.com/sendsms.php" />
    </port>
    </service>
      </definitions>
```

Example of .arff File with 6 WSDL documents:

@relation web_services

@attribute service_name string
@attribute service_parameter string
@attribute class{a,b,c,d}

@data

sendsmsworld,"webservice From Email Address Country Code Mobile Number Message send SMS Response Result true send SMS SoapOut parameter Http ",b
FreeFaxService,"Free Fax Service Request Response world wide transmission Fax Binding encoded",b
servicexml,"message Phone Secret result secret GetSecret key phone prompt PhonePort PhonePhoneBinding PhoneService",b
areacodeworldwebservice,"AreaCodeWorldResponse AreaCodeWorldResponse CREDITS AVAILABLE Area Code World HttpGetOut body area code",a
globalweather,"Get Weather City Name Get Weather Response Get Weather Result Get Cities By Country",a
weathernet,"Country Name Get Weather Soap Out Get Cities By CountryResponse GetCities By CountryResult Get Weather SoapIn",a
Convertziptocitystate,"Longitude Latitude Country Firm City StateAbbrev ZipCode BarCode FIPS CSMA TimeZone AreaCode StreetName Distance",a

List of Publications

- Shailja Kaushik and Shalini Batra, "*Automatic Classification of WSDL documents*" published in International Conference on Intelligent Systems and Networks (IISN-2009) held at Institute of Science and Technology, Klawad, Yamuna Nagar from February 14-16, 2009, Pages 211-214.