

MINING MEANING OF WEB SERVICES

by

Rozina Nisa

2011-NUST-MS PhD-CSE (E)-27

MS-11(CSE)



Submitted to Department of Computer Engineering

in fulfilment of the requirements for the degree of

Masters of Science

in

Computer Software Engineering

Thesis Supervisor

Dr. Usman Qamar

College of Electrical and Mechanical Engineering

National University of Science and Technology

May 2013



In the name of Allah most beneficent most merciful

وَلَا يُحِيطُونَ بِشَيْءٍ مِّنْ عِلْمِهِ إِلَّا بِمَا شَاءَ

And they can't encompass any thing from His knowledge, but to extend He wills [2:255]

This page is intentionally left blank

ACKNOWLEDGEMENTS

"Those who do not thank people, they do not thank Allah." (Al-Tirmidhi 1878).

There is an old saying 'It takes a village to raise a child' and I am not an exception. It is with the grace of Almighty that I was led into the company of the following people, whose generosity, enthusiasm, and good shepherding sustained me in producing this work: Dr. *Usman Qamar*, Dr. *Shoab Ahmed Khan*, Dr. *Saad Rehman*, Dr. *Muhammad Abbass* and all those people who generously shared their knowledge and expertise with me. I am beholden to my family for their love and continuing encouragement. My friends always backed me up very strongly. Therefore, I thank every single one of them for their support.

ABSTRACT

Web services have been evolved as a versatile and cost effective solution for exchanging dissimilar data between distributed applications. They have become a fundamental part of service oriented architecture (SOA). The most challenging problem being faced by service oriented architecture is to figure out what a service does and how to use its capabilities without direct negotiation with service provider. Discovering and exploring web services registered with UDDI registry or Web Services-Inspection (WS-Inspection) documents requires exact search criteria such as service category, service name and service URL.

This study focuses on creating a smarter automated web service classification technique by applying Maximum Entropy machine learning algorithm to attributes of Web Service Description language (WSDL) documents. WSDL document allows web services clients to learn operations, communication protocols and correct message format of service. Manually analyzing WSDL documents is the best approach but most expensive. Therefore a text mining based approach is suggested for classifying web services into functional groups.

Keywords – *Service oriented architecture (SOA), UDDI registry, WSDL documents, Text Mining.*

Contents

ACKNOWLEDGEMENTS	i
ABSTRACT	iii
Chapter 1	1
INTRODUCTION	1
1.1 Overview	1
1.2 Background and Motivation	1
1.3 Objective and Contribution	2
1.4 Outline	3
Chapter 2	4
LITERATURE REVIEW	4
2.1 Service Oriented Architecture (SOA) an Overview	4
2.2 Type of Web Services	6
2.3 WSDL Document Structure	8
2.3.1 Services	9
2.3.2 Messages	9
2.3.3 PortTypes	10
2.3.4 Types	10
2.3.5 Bindings	11
2.3.6 Documentation	12
2.4 Related Work	12
2.4.1 Text Classification Algorithms – State of Art	12
2.4.2 Web Service Classification/Clustering Approaches	13
2.5 XML Parsing	16
2.5.1 XML Parsing APIs	16
2.5.2 WSDL Parsing APIs	17
2.6 Text Pre-processing	18
2.6.1 Tokenization	19
2.6.2 Stemming	19
2.6.3 Lemmatization	20
2.6.4 Stop Word Elimination	22

2.6.5	Function Word Elimination	22
Chapter 3	25
PROPOSED CLASSIFICATION APPROACH		25
3.1	Overview	25
3.2	Feature Extractor	27
3.3	Text Pre-processor.....	29
3.3.1	Tokenization	30
3.3.2	Word Splitter	30
3.3.3	Stop Word Removal	30
3.3.4	Function Word Removal	30
3.3.5	Lemmatization	30
3.4	Web Service Classifier.....	31
3.4.1	Vector Formation.....	31
3.4.2	Classifier.....	32
Chapter 4	34
CLASSIFICATION RESULTS AND COMPARISON.....		34
4.1	Overview	34
4.2	Data Preparation.....	34
4.3	Experimental Results.....	35
4.4	Comparison to Existing Classification and Clustering Techniques.....	37
4.6	Analysis of Results.....	39
Chapter 5	41
CONCLUSION AND FUTURE WORK		41
5.1	Overview	41
5.2	Conclusion.....	41
5.3	Future Work	42
APPENDICES		44
Appendix A.1:	Sample WSDL File of Water Service	44
Appendix A.2:	Cross Validation Results	48
Appendix A.3:	Percentage Split Results	52
BIBLIOGRAPHY.....		56

LIST OF FIGURES

Figure 2.1: Communication in Service Oriented Architecture (SOA)	5
Figure 2.2: Core components that make up SOA implementation	6
Figure 2.3: Annotating Semantics in WSDL Document	7
Figure 2.4: Block diagram illustrating relationship between WSDL sections.....	8
Figure 2.5: Structure of WSDL message.	9
Figure 2.6: Structure of WSDL portType.....	10
Figure 2.7: Structure of WSDL types	11
Figure 2.8: First phase rule set for Porter stemmer algorithm	20
Figure 2.9: Steps for lemmatization of noun and verb phrases.....	21
Figure 2.10: Demonstration of stop words elimination	22
Figure 3.1: Schematic block diagram illustrating architecture of our classification approach	25
Figure 3.2: Framework of web service classification	26
Figure 3.3: WSDL Schema extracted from WaterService web service.....	28
Figure 3.4: WSDL Messages extracted from WaterService web service	28
Figure 3.5: WSDL PortTypes extracted from WaterService web service	29
Figure 3.6: Information in WaterService after pre-processing phase.	31

LIST OF TABLES

Table 2.1: Summarization of WSDL features used by existing techniques	15
Table 2.2: List of function (closed words) words categorized into their word classes.....	23
Table 3.1: Confusion Matrix for Classification of Dataset Categories.....	32
Table 4.1: Dataset Categories and their web services.....	34
Table 4.2: Mean precision and recall for classification of dataset categories	35
Table 4.3: Relationship between Positive and Negative Results	36
Table 4.4: Statistics of Training and Test Data.....	36
Table 4.5: Performance comparison with existing techniques	37
Table 4.6: Performance evaluation related to five identified categories	38

LIST OF ABBREVIATIONS

<u>Abbreviation</u>	<u>Illustration</u>
UDDI	Universal Description, Discovery and Integration
SOA	Service Oriented Architecture
IR	Information Retrieval
WSDL	Web Service Description Language
WSMO	Web Service Modelling Ontology
SWSF	Semantic Web Service Framework
OWL	Ontology Web Language

Chapter 1

INTRODUCTION

Where is the wisdom we have lost in knowledge?

Where is the knowledge we have lost in information?

(‘Choruses from the Rock’ by T.S.Eliot - 1934)

1.1 Overview

Service oriented architecture (SOA) is glue that allows web applications to work in collaboration. It has become a driving force for Service-oriented Computing (SOC) paradigm. In heterogeneous environments SOC paradigm uses web services as the basic building block to support low cost, easy and rapid composition of distributed applications. Web services leverage existing distributed business processes from tightly coupled to loosely coupled service oriented architectures. Therefore business processes are no longer limited to web services within the enterprise’s boundary.

1.2 Background and Motivation

In service-oriented architecture, a service is defined as a software component that communicates using SOAP and XML-based standard messaging protocols [1]. A web service exposes its interfaces using Web Service Description Language [2]. In service-oriented architecture, there are three types of players namely *Service Consumers*, *Service Providers* and *Service Registries*. A central repository called Universal Description, Discovery and Integration (UDDI) is used by service providers to publish and register their web services. UDDI registries are used by web service consumers to locate their required web services and metadata associated with them.

Majority of service providers including Yahoo and Amazon used to publish their web services using their own websites instead of public registries or service brokers. Therefore, there is an increasing trend of searching web services through search engines [3]. The growing number of web services on web raises a challenging problem of locating desired web services. Several search engines [4, 5, and 6] provide keyword or query based web service search facility. They partially match search queries with terms in web service name, location and other attributes specified in WSDL file. These search queries do not capture ultimate semantic of web services. Users must be aware of correct and concise keywords to retrieve the most relevant set of web services. For example, if a user is searching for ZipCode, WSDL files containing keywords postal code and zip will not be returned in search result.

In recent few years, text mining has gained a lot of attention due to increasing need for managing vast amount of information in text documents. Semantic text analysis of web service description (WSDL) files to determine their functionality and capabilities is really useful for efficient retrieval of web services. Semantics are attached to web services by annotated them using special markup languages. Plebani, P. adopted language SAWSDL (Semantic Annotation for WSDL) for annotating WSDL documents with enrich semantics [7]. Beniamino Di Martino presented schema match making approach for semantic web service discovery using WSDL descriptions, WSDL-S, WSMO, Ontology Web Language (OWL) ontologies, SWSF and OWL-S [8]. But manually annotating huge set of available web services is not a feasible task.

However semantic information of web service can be extracted from its WSDL file. Web service semantics can be inferred by mining service description, messages, operations and schema details in WSDL file. Woogle, a web service search engine [9] is build on the basis of clustering information available in WSDL descriptions, operations and their parameters. Khalid Elgazzar [3] proposed a novel approach to mine features from WSDL documents and cluster them into functionality based groups. Marcello Bruno [10] proposed an approach that uses service textual description to perform automatic classification across specific domains.

1.3 Objective and Contribution

This thesis is aimed at improving discovery of web services by proposing a text mining based novel approach for web service classification. We intend to create a classification mechanism that focuses on pre-processing of web service features available in WSDL documents.

Therefore, our main contributions are as follows:

1. We present an approach that extracts contents (specific information from service name, service documentation, WSDL messages, WSDL ports and WSDL schema) from WSDL documents in order to classify web services into eight functionally similar categories. This thesis aims to provide smarter web service curation using text-mining techniques.
2. We compare our approach with existing classification and clustering techniques and prove that our approach gives high precision and recall values.

1.4 Outline

Chapter 2 discusses the related work. We present the design and methodology of our classification approach in chapter 3. Chapter 4 discusses results and detail comparison of our approach with existing techniques. Finally, chapter 5 concludes this thesis and proposes an outlook of possible extensions, modifications, and improvements as future work.

Chapter 2

LITERATURE REVIEW

In this chapter, we give a brief overview on the state-of-the-art of existing approaches of web service classification and clustering. We mainly considered the work that is based on mining information available in WSDL documents. We describe methodology of each approach and selected WSDL features.

2.1 Service Oriented Architecture (SOA) an Overview

Service oriented architecture (SOA) allows creation of applications by combining interoperable and loosely coupled services. It is a style of information system (IS) architecture. These loosely coupled services inter-operate with components in distributed environment and are independent of underlying programming language and platform. For example, a web service might be implemented in J2EE or .Net and application utilizing web service might be using different language and platform.

Following are the key characteristics of Service Oriented Architecture.

- SOA web services have well defined interfaces written in platform independent XML documents. Web services are described by a standard known as Web Service Description Language (WSDL).
- SOA web services communicate with messages defined using XML Schema. Communication among service providers, service consumers and services normally takes place in a heterogeneous environment with least knowledge about communicating neighbour. Therefore, messages between web services are considered to be the key business documents exchanged between enterprises.
- SOA web services are registered by service providers in service registries. Service consumers can look up desired services in registry and invoke them. The standard used for service registry is Universal Description, Definition, and Integration (UDDI).

- Quality of service (QoS) is associated with each SOA web service. Some important QoS elements include reliable messaging, security requirements and policies regarding service consumers.

Figure 2.1 illustrates communication of components in Service Oriented Architecture (SOA)

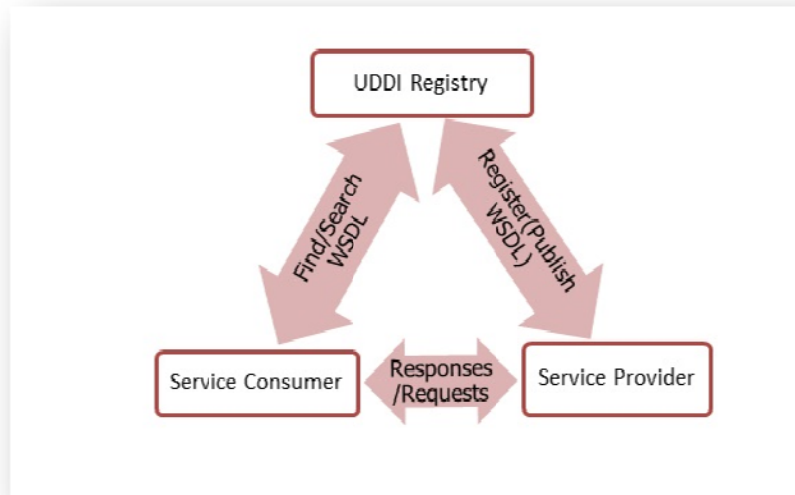


Figure 2.1: Communication in Service Oriented Architecture (SOA)

IT industry is based on infrastructure that is heterogeneous across system software, operating systems, applications and application infrastructure. It is not possible to build new infrastructure for business processes from scratch because enterprises has to quickly respond to business changes and leverage investments for newer business requirements. Service oriented architecture is a giant step to address new business requirements in a granular fashion.

The widespread availability of XML standard based web services has given a new momentum to Service oriented architecture. Figure 2.2 illustrates the advantages being offered by core elements of SOA.

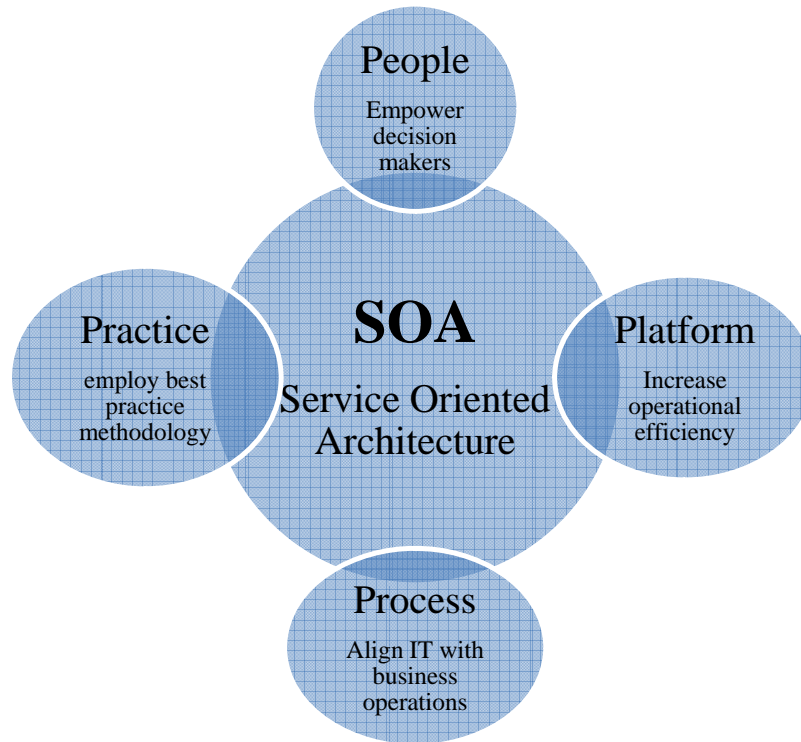


Figure 2.2: Core components that make up SOA implementation

Some of the main goals of Service Oriented Architecture are listed below.

- Individual business requirements can be implemented directly by web services. The main advantage offered by this approach is rearrangement of web services according to changing business workflows. Thus, web services improve overall alignment of business processes.
- Single web service can be used by multiple business applications thus avoiding duplication of resources, reducing maintenance costs and development time. This will in turn increase quality and consistency of data.
- SOA interfaces improves application integration by providing standard based application integration and avoid overhead of special information adapters.
- “A single source of truth” can be established by SOA that can be validated, protected and more reliable as compared to multiple distinct data sources.

2.2 Type of Web Services

Knowledge discovery by data mining has gained much importance in recent few years, especially in mining web service description files.

There are two types of web services

1. Semantic Web Services

Semantic web services describe themselves by using Semantic Web Services Ontology (SWSO). SWSO is specified using Semantic Web Service Language (SWSL) [11]. Semantic web services annotate web service description documents with richer semantic specification in order to provide accurate and flexible automated web service discovery. Figure 2.3 illustrates process of incorporating semantics in WSDL document.

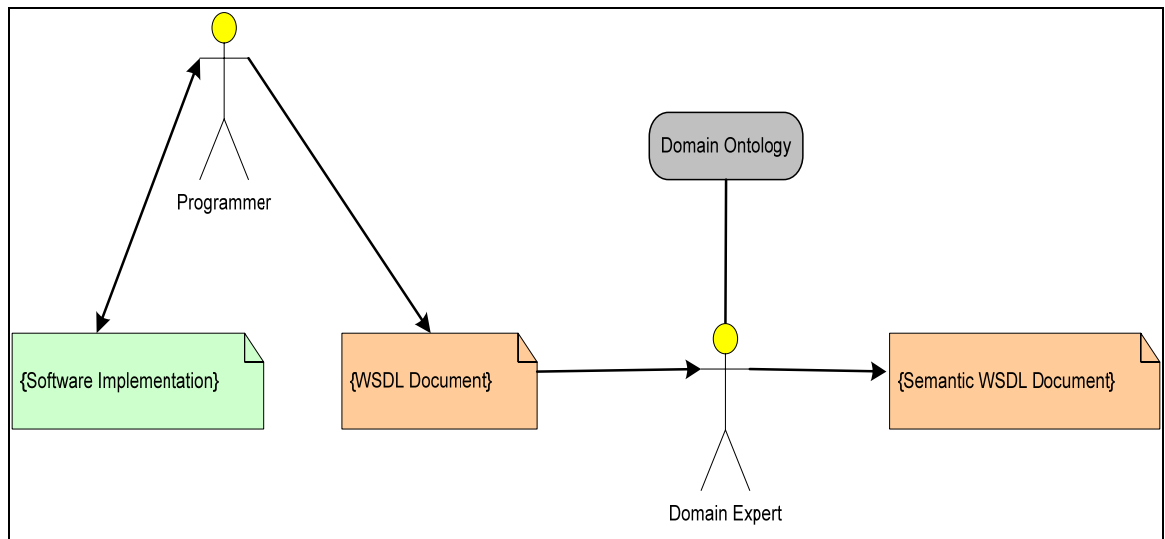


Figure 2.3: Annotating Semantics in WSDL Document

2. Non Semantic Web Services

Non-semantic web services describe themselves by Web service Description Language (WSDL). No semantic details are manually added to WSDL document. Semantic web services are extension of non-semantic web services.

Non-semantic web services have gained more popularity in software development industry because of their simplicity. Web service discovery techniques vary depending upon web service description language. High-level match making techniques [11] are used to discover semantic web services whereas text-mining techniques are used to retrieve non-semantic web services [3]. This thesis is targeting non-semantic web services classification approach.

2.3 WSDL Document Structure

Non-semantic web services are described by an XML based language called Web Service Description Language (WSDL) [2]. A web service provider publishes its web service by registering WSDL file and some related documents with UDDI registries. A WSDL document specifies

- Details of operations required to invoke and consume the web service.
- Physical location of WSDL.
- Binding information of several transports formats as well as protocols between web service and web service requester.

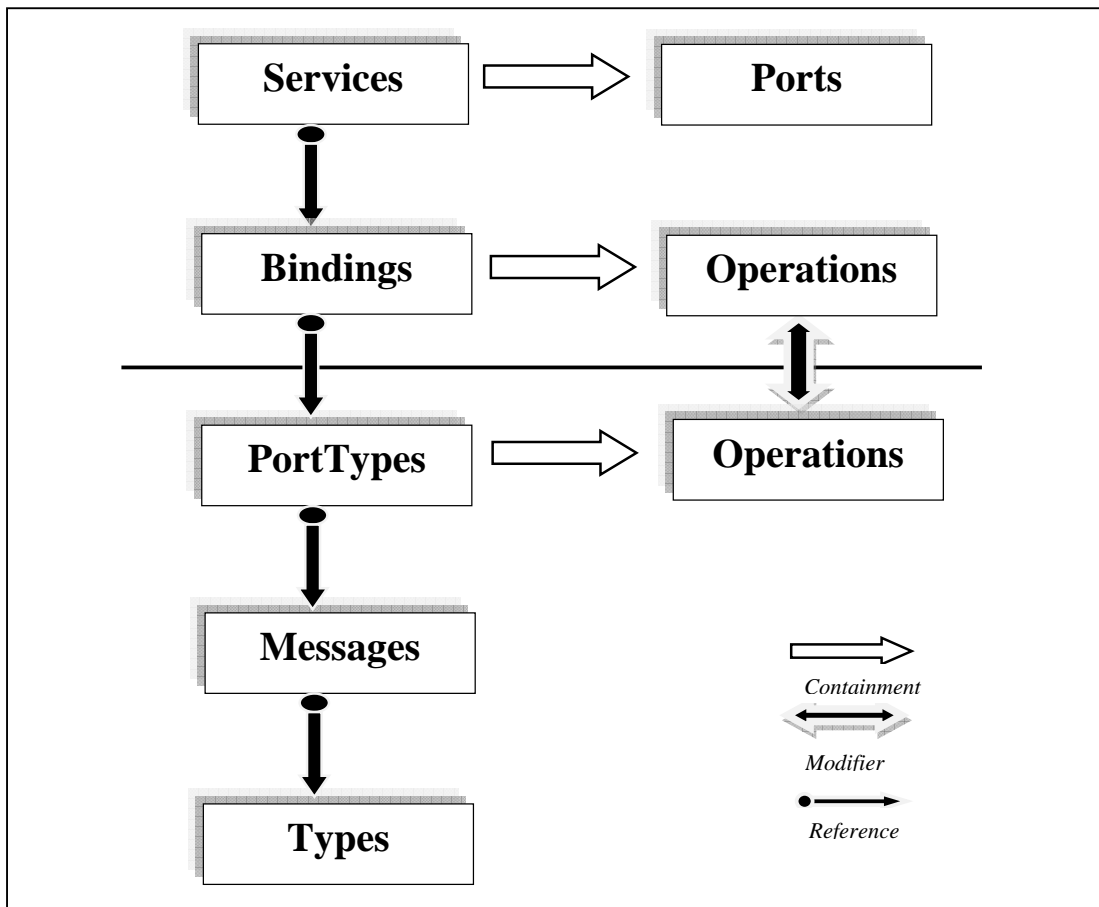


Figure 2.4: Block diagram illustrating relationship between WSDL sections.

According to Figure 2.4 definitions in Type section are used by Messages section, PortTypes use definitions in messages section. PortTypes section is referred by Binding section and in turn, it is referred by Services section. Operation elements are contained by PortTypes and Binding sections and port elements are contained by Services section.

The main advantage of WSDL is that it is platform, protocol and language independent. It does not require use of any specific language for web service implementation. It also supports message exchange through SOAP over www. We have considered *Web Service Description Language (WSDL) version 1.1* for feature extraction because WSDL version 2.0 is more complex and not supported by most of software development tools. A sample WSDL file is presented in Appendix A.1.

A detail description of WSDL features is given below.

2.3.1 Services

A service element describes the name of service and aggregates multiple service ports.

2.3.2 Messages

Messages specify data being communicated between consumers and providers of web service. There are two types of messages supported by web service, input and output messages. Web service parameters are described by input messages and data returned by web service is described by output messages.

Figure 2.5 illustrates structure of message element.

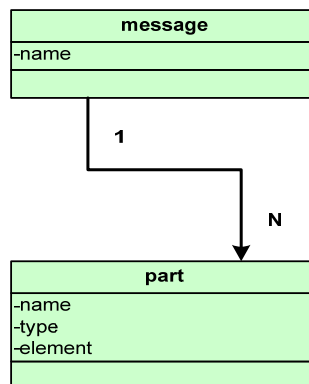


Figure 2.5: Structure of WSDL message.

Each message consists of zero or more part elements. Each part element refers to input/output parameters of web service operation.

2.3.3 PortTypes

A web service can have multiple ports. Ports refer to web service end points and bindings define the transport protocol for these ports. A port type element contains a set of operations supported by web service. Each operation contains input/output parameters. Figure 2.6 illustrates structure of portType element.

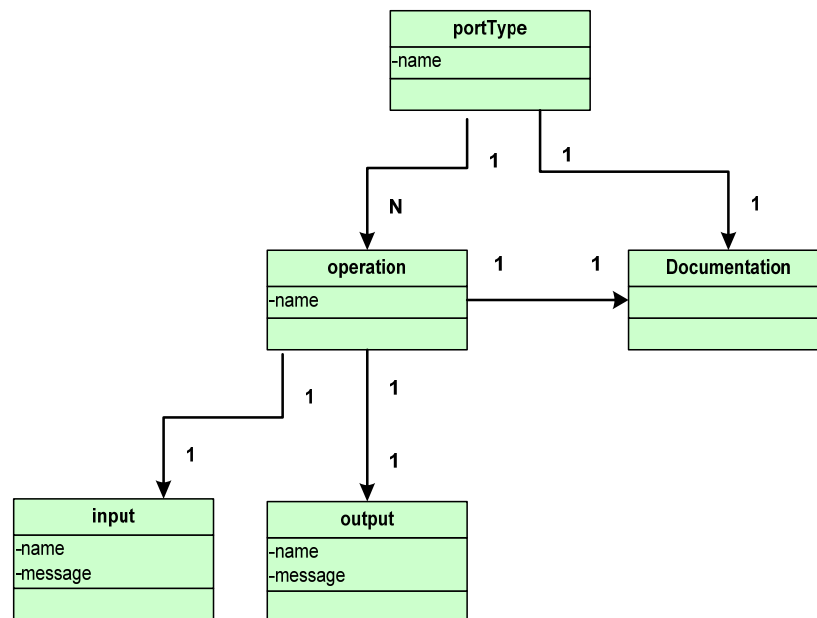


Figure 2.6: Structure of WSDL portType

2.3.4 Types

This element describes language and machine independent data containers for message exchange. In short, data types used by web service are defined in Types element. W3C XML Schema specification is used by WSDL to define data types. Types element is not used if web service uses simple build in types such as integers and strings. Figure 2.7 illustrates structure of types element.

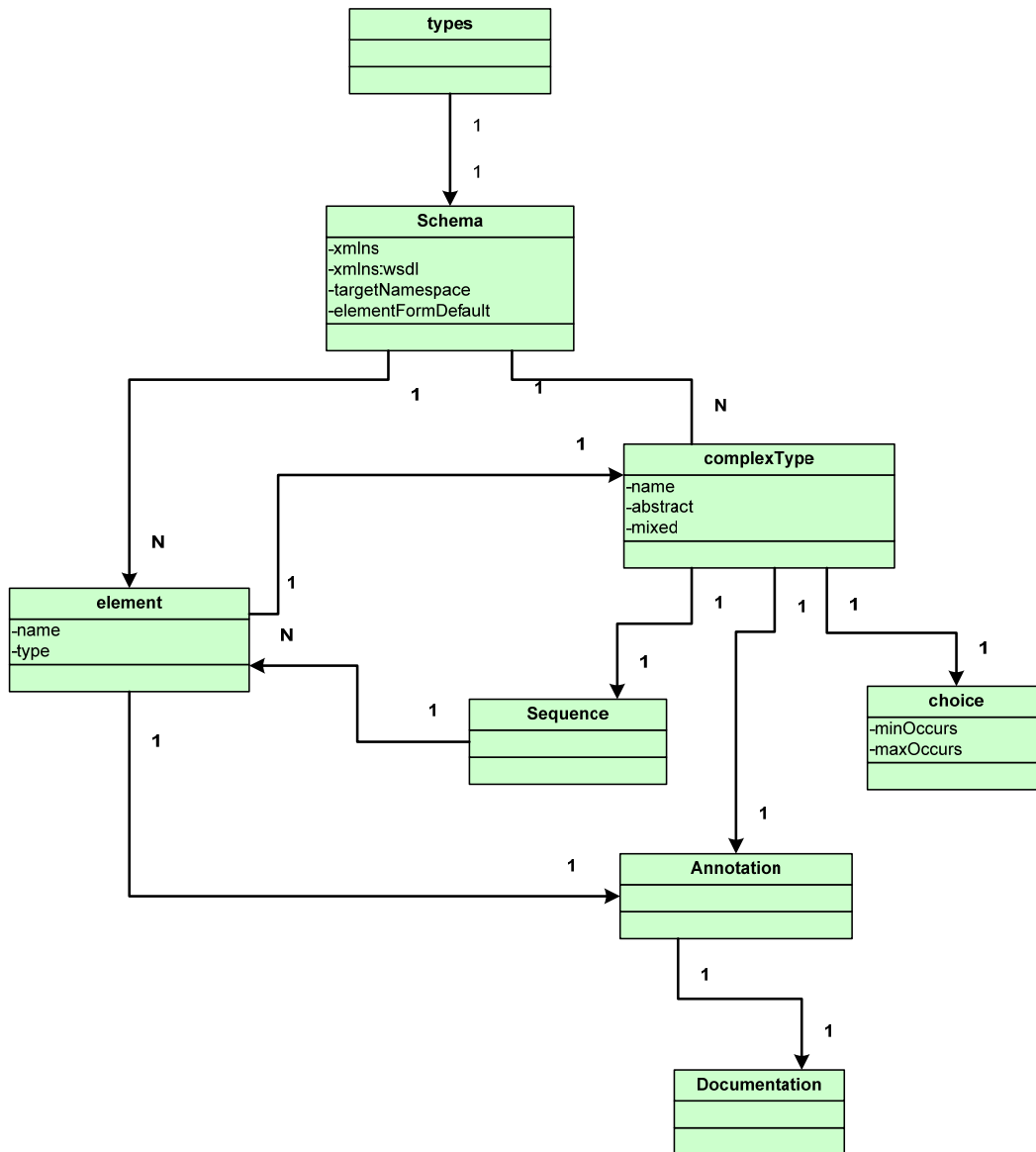


Figure 2.7: Structure of WSDL types

2.3.5 Bindings

Binding component provide details of data format and transport protocol for a portType operation. Binding through multiple transport protocols such as HTTP GET, HTTP POST, or SOAP are available. Multiple bindings can be specified for a single portType.

2.3.6 Documentation

Documentation element gives textual description of web service functionality. It is not specified in all WSDL documents.

2.4 Related Work

2.4.1 Text Classification Algorithms – State of Art

Automatic text classification is becoming an important research topic and yearning application because of huge set of digital text document that need to be process daily. In general, automatic text classification plays a vital role in text summarization, question answering and information extraction.

Intuitively classifying documents into a predefined set of categories is the task of text classification. More formally, text classification classifies each document D_i in a set of documents $\{D_1, D_2, D_3 \dots D_n\}$ to a category C_i in set of categories $\{C_1, C_2, C_3 \dots C_n\}$. In the research community the main approach for text classification is based on supervised machine learning techniques. These techniques require initial data set of classified documents to build a classifier.

Text classification is done using different machine learning techniques. Many of the researchers have used different classifiers for example Ant Miner, Support Vector Machines, Naïve Bayes, and K-nearest neighbour etc. Some of them have proposed new techniques by proposing hybrid classifiers for web service classification. Some algorithms have proven to perform better than others such as Maximum Entropy and Support Vector Machine.

Naïve Bayesian is often used in automatic text categorization because of its effectiveness and simplicity [29]. However, text is not modeled properly by Naive Bayesian which degrades its performance. Ping Bai [30] proposed an improved naïve Bayesian classification algorithm for classifying web text. In common Naïve Bayesian classifier all terms in text are equally important but Ping Bai suggested that terms in each title are more significant. This approach improves precision of text classification results.

Different text classification approaches have been developed based on principles of Support vector machine (SVM) algorithm [31]. Text classification by Support vector machine yields excellent precision but gives poor recall. However recall can be improved by adjusting threshold associated with SVM. Classification by SVM requires training by extensively labeled data, but labeling huge training dataset is resource and time consuming activity. Also high text dimensions results in sparse feature space. Algorithm performance is negatively affected by these factors.

Maximum entropy is extensively used for natural language processing tasks such as tagging of parts of speech, text segmentation and language modeling. Maximum entropy offers a lot of advantages as compared to other supervised machine learning algorithms such as Naïve Bayesian. In [26], Hui Wang performed extensive experiments to compare performance of Maximum Entropy with Naïve Bayes and Support Vector Machine (SVM). Maximum Entropy consistently outperforms SVM and Naïve Bayesian. This is because Maximum Entropy is robust and does not make inherent conditional independence assumptions between terms. It gives better performance than SVM in classification of sparse data.

2.4.2 Web Service Classification/Clustering Approaches

Different web service classification and clustering techniques has been proposed based on mining textual details in WSDL documents. In [32], Suman Saha proposed a two-step process of web service classification. In the first step, Tensor Space Model is used for capturing internal structure and text content of WSDL documents. In the second step tensor space model of each individual component is combined using Rough Set approach. This approach obtains two step improvements by using Tensor space model and Rough set based ensemble classifier. However, only three features from WSDL files are extracted for classification namely service name, operation names and input/output names and their descriptions. They do not consider WSDL Schema and WSDL Messages which reveal important information about functionality of web service.

Ioannis Katakis in [33] performed automated classification of web services using semantic annotations and textual descriptions of OWL-S instead of using WSDL

documents. This approach improves overall accuracy of classification by using extended feature vector and a collection of classifiers.

According to Fangfang Liu [34], very small text fragments in WSDL documents are not suitable for applying traditional information retrieval techniques. He suggested an approach for classification of web service operations based on calculating semantic distance between terms of two web services. Web services are categorized into similar groups based on semantic distances between WSDL terms, thus improving accuracy of similarity matrix.

A novel clustering approach is suggested by Jiangang Ma [35] to remove irrelevant services based on semantic analysis of user query. This approach utilizes Probabilistic Latent Semantic Analysis to capture semantic of user query and service descriptions so service comparison is performed at concept level.

Woogle [9] is a web service search engine proposed by Xin et al. This engine is capable of performing similarity search between web services by finding similar and composable operations. However, this approach has not used *WSDL Types*, which plays an important role in uniquely identifying a web service.

Marcello Bruno [10] proposed automated web service classification technique to map services to specific domains. This approach also identifies key concepts in WSDL document and builds a network of relationships between different web service annotations. In this approach web service classification is performed by Support Vector Machine algorithm using web service documentation and user queries as classification features.

The process of text mining for text categorization (text classification, topic spotting) has gained prominent status in the field of information retrieval in past few years. Liu and Wong [12] proposed a web service clustering approach by text mining web service description features such as WSDL contents, web service context, host name and service name. Clustering/classification process is not significantly affected by service host name and service context features. This is because service providers publish their web services mostly on their own website and one site might contain multiple web services. Therefore, mining host name and surrounding index web pages does not offer much advantage in determining meaning of web service.

Khalid Elgazzar [3] modified Liu and Wong [12] approach by selecting different set of web service features. These features include WSDL contents, types, messages, ports and web service name. This technique gives high precision and recall values. Aparna Konduri [14] proposed a hierarchical clustering approach that uses WordNet to compute web service semantic similarity. WSDL features used by this approach are WSDL operation names and WSDL parameter names. This approach does not yield high precision value because selected features do not convey proper meaning of web service.

Web service search engines must ensure that every searched web service must have high precision (usable) and recall (can be discovered). Jing Zhang [15] performed classification of web services based on structure of Web Service Description Language (WSDL) file. This approach does not yield good results because of plain statistical method used for classification. Table 2.1 summarizes WSDL features used by each web service classification/clustering approach.

Table 2.1: Summarization of WSDL features used by existing techniques

Technique		Features	
1.	An approach to support web service classification and annotation [10]	1.	Web Service Documentation (WSDL and other provided documents)
		2.	User Queries
2.	Clustering of web services based on semantic similarity [14]	1.	Operation names
		2.	Operation parameter names (Input/Output)
3.	Web service classification [15]	1.	Service name
		2.	Service Documentation
		3.	Operation names and description
		4.	Operation parameters (Input/Output)
4.	Web service clustering using text mining techniques [12]	1.	WSDL contents
		2.	WSDL context
		3.	Service host

		4. Service name
5.	Clustering WSDL documents to bootstrap the discovery of web services [3]	<ol style="list-style-type: none"> 1. WSDL contents 2. WSDL types 3. WSDL messages 4. WSDL ports 5. Service name
6.	Classification of web services using Tensor [32]	<ol style="list-style-type: none"> 1. Service name and description 2. WSDL operation name and description 3. Operation parameter names and description (Input/output) 4. UDDI descriptions

2.5 XML Parsing

XML (Extensible Mark-up Language) is a language for converting documents in machine-readable form. Most of the information on internet is shared in the form of XML documents. Parsing and uploading XML data is a common activity between heterogeneous and distributed applications.

Web service description files are also written using Extensible Mark-up Language so we need to parse them before being passed to text pre-processing phase. For this we perform analysis of various XML parsers specifically WSDL parsers.

2.5.1 XML Parsing APIs

There are different APIs for parsing XML files. Some of these are listed below.

1. Simple API for XML (SAX)

SAX is an event-driven and lexical interface in which contents of document are read serially and callbacks are sent to handler object methods of user's design. Although SAX is an efficient parser API but random information from XML is difficult to extract. This is because it burdens application author by keeping track of part of XML document being processed.

2. Pull Parsing (XPP)

Xml Pull Parser is used when all xml elements need to be processed efficiently and quickly. XPP is the simplest way of parsing xml because it is just one grade up from XML tokenization. Tokenization and buried pull interface is part of every SAX parser. XPP exposes this buried layer to allow faster XML parsing.

3. Document Object Model (DOM)

Document Object Model is an application-programming interface that represent entire document as a tree of node objects. Most implementations of DOM are memory intensive as entire XML document need to be loaded into memory for tree construction before objects are being accessed. Nevertheless, most of the available XML parsers make use of DOM because of its platform and language neutral interface.

4. Data Binding

Data binding is a type of XML parser in which XML elements are converted into hierarchy of custom typed classes. This contrasts from Document Object Model, which creates generic objects from XML elements. Data binding identifies errors at compile time rather than run time thus simplifying implementation.

2.5.2 WSDL Parsing APIs

Classification of web services require parsing of WSDL files. As constituents of WSDL are already well defined so it feasible to extract desired web service information using existing WSDL parsing APIs. Different APIs have been developed for efficient parsing of WSDL files using XML parsers described in *Section 2.5.1*. A brief overview of some of these APIs is given below.

1. Apache Woden

Apache Woden [27] is DOM-based XML parser that implements a Java class library for reading, processing and writing WSDL documents. Currently it supports WSDL version 2.0 but later it will be providing support for past, present and future versions of WSDL documents. Thus, Apache Woden delivers a “framework” that can

accommodate future and past WSDL specifications by saving the effort to rewrite entire processor.

2. Membrane SOA Model

Membrane SOA Model [21] is an object model and Java API for parsing XML Schema and WSDL files. It also provides support for analysis and comparison of schema and WSDL documents. WSDL version 1.1 is supported by membrane SOA Model. It is a lightweight java API that provide following rich set of features.

- WSDL document creation
- Schema document creation
- SOAP request creation
- SOAP Request Template creation
- WSDL document parsing
- WSDL document manipulation
- Comparison of two WSDL documents
- Comparison of two WSDL Schema documents

3. Java APIs for WSDL (JWSDL)

JWSDL is a Java API for reading, writing, modifying, creating and re-organizing WSDL documents. It is not designed to perform syntactic validation of WSDL documents. However, it can be used for validating WSDL file semantically. It is dependent upon *org.w3c.dom* interfaces. JWSDL is designed to represent incomplete and incorrect WSDL documents in WSDL editors and tools.

2.6 Text Pre-processing

In text pre-processing, original textual information is converted into ready structure for data mining. Real word data is noisy and inconsistent that may lead mining process to inaccurate result. Therefore pre-processing is required to extract accurate and consistent features from data. In other words, pre-processing is the process of slotting in textual documents to an information retrieval system (IR). An efficient pre-processor provides high IR performance (high precision and recall). It presents documents effectively in terms of space and time. This is the most complex and critical phase that converts each document as a set of index terms. The main purpose of pre-processing is to categorize textual documents into relevant groups.

Quality of generated index terms determines efficiency and effectiveness of data mining process [17].

Some text pre-processing methods are described below.

2.6.1 Tokenization

Tokenization is the foremost stage of Morphological Analyses. It is the process of splitting a stream of text into words, symbols and phrases called tokens. This dataset of tokens is given as input to next pre-processing steps. Although text is stored in machine-readable format, meaningless characters like hyphen, comma, brackets etc. need to be eliminated by tokenization [17].

2.6.2 Stemming

Stemming is the process of reducing words to their stem/root. The hypothesis behind stemming is that words originating from same stem or root often convey similar meanings. For example words:

- Printer
- Printing
- Prints and
- Printed are reduced to stem Print.

Stemming improves IR by matching similar words and reduces indexing size by combining words with same root. Porter stemmer algorithm [18] is most commonly used for stemming. This algorithm consists of five sequential phases and various conventions are used to apply rules during each phase for example select a rule from group of rules that can be applied to longest suffix. Rules being followed in first phase are illustrated in Figure 2.8.

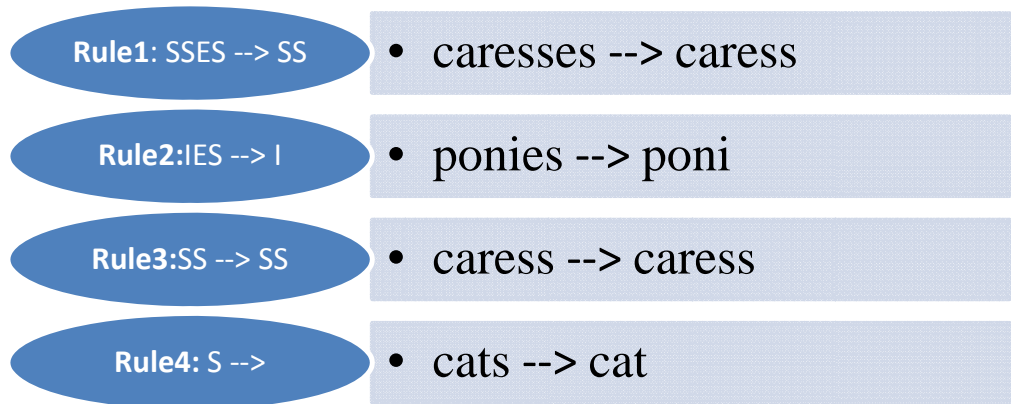


Figure 2.8: First phase rule set for Porter stemmer algorithm

2.6.3 Lemmatization

Lemmatization is the process of reducing varied or derivational forms of a word to a common root form. It seems that both stemming and lemmatization perform similar task but there is significant difference.

A crude heuristic process is used by stemming to cut off ends of words with the aim of achieving its goal correctly but often it eliminates derivational affixes. However, lemmatization reduces words to a valid dictionary form by using vocabulary and performing morphological analysis of words. This valid dictionary form is known as *lemma*. For example, consider a token *saw*, stemming would reduce it to word *s*, whereas lemmatization will return either *saw* or *see* depending on whether the word is used as noun or as verb. In short, different inflectional forms of word (lemma) are collapsed by lemmatization whereas stemming [19] collapse only derivationally related words. Figure 2.9 illustrates steps for lemmatization of noun and verb phrases used by Badam-Osor Khaltar [28] for information retrieval.

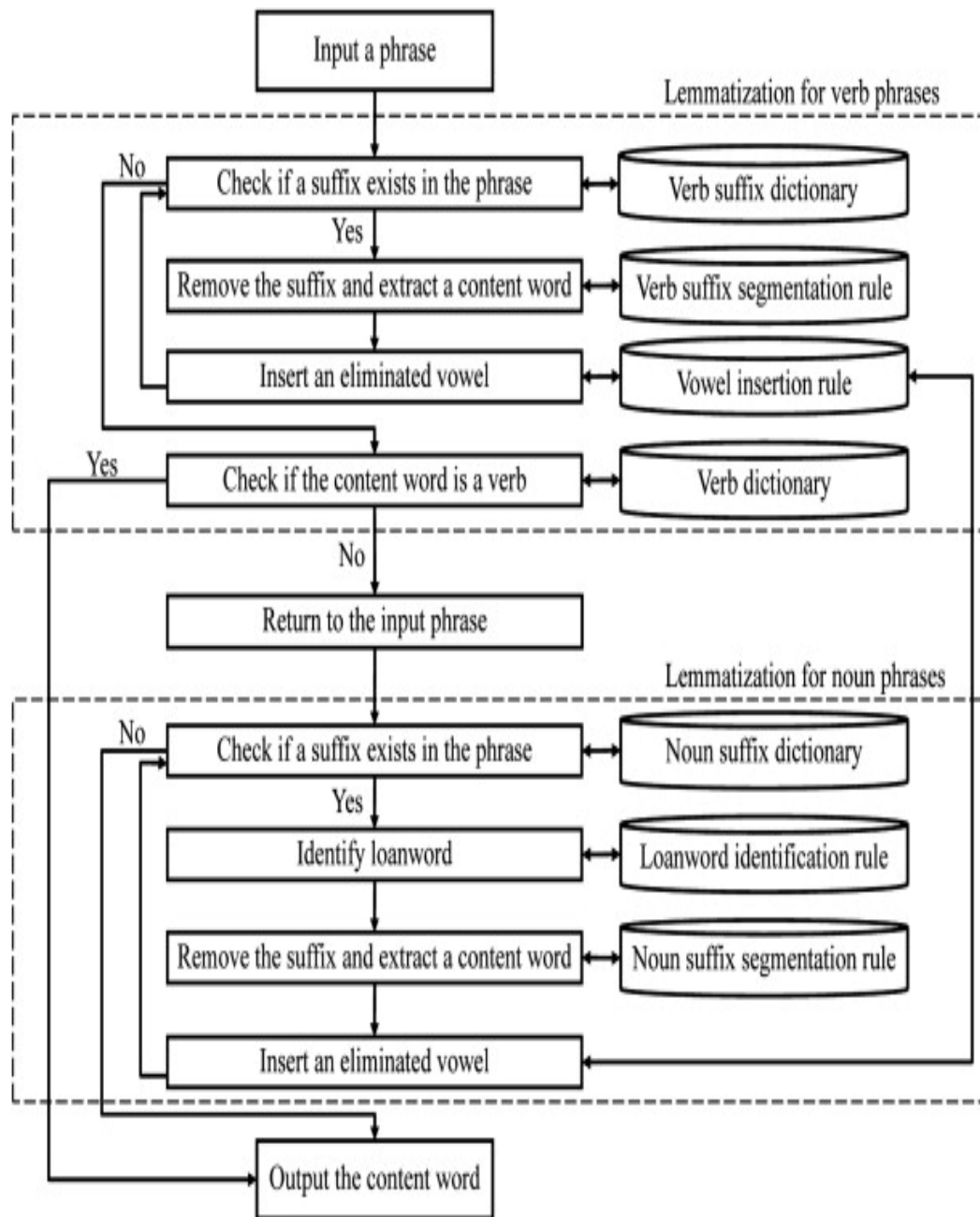


Figure 2.9: Steps for lemmatization of noun and verb phrases.

Most of web service classification and clustering approaches make use of stemming rather than lemmatization. Our suggested approach has used lemmatization rather than stemming and has achieved high precision and recall values.

2.6.4 Stop Word Elimination

In English language, there are a number of words that do not contribute to the meaning of sentence and can be easily eliminated. These words are known as *Stop Words*. Stop words are useless in IR and text mining. Examples of stop words are about, a, an, by etc. There are about 400-500 hundred stop words in English language [16].

Search engines eliminate stop words from a search query because stop words force much less traffic than keywords. In short, high density of stop words reduces importance of contents for search engines.

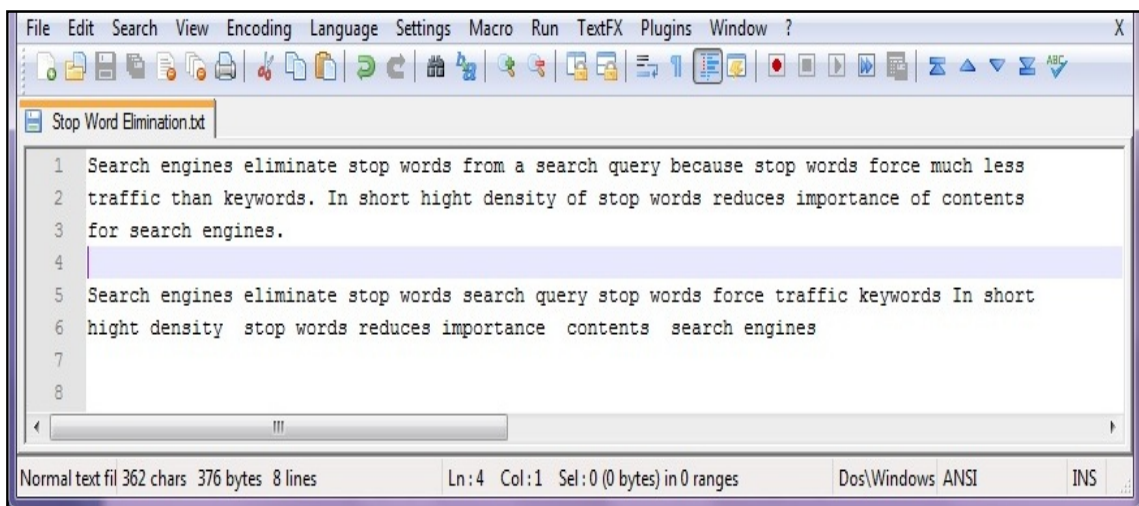


Figure 2.10: Demonstration of stop words elimination

In Figure 2.10, second paragraph shows elimination of stop words from first paragraph. Second paragraph is shorter than first one: 32 words versus 23. This indicates that almost 30 – 40 percent of words are stop words and this text does not help search engines in text mining. This is why precision is improved by removing stop words from textual information because text mining is more focused and to the point [9].

2.6.5 Function Word Elimination

Function words are also known as synsemantic, grammatical or structure-class words. These words express grammatical and structural relationships with other words in a

sentence. They do not have substantial lexical meaning but act as glue to hold sentences together. Therefore, they can be easily removed from textual information without affecting meaning of sentence or document. Words that are not function words are called lexical words, content words or auto semantic words [20]. Function words are independent of each other and Poisson distribution model can be used for distinguishing content words from function words [12]. Function words can also be eliminated by creating your own function list from publically available function lists. Table 2.2 illustrates function list compiled by flesl.net.

Table 2.2: List of function (closed words) words categorized into their word classes.

Prepositions		Pronouns		Determiners	
about	in	I	it	the	a/an
across	inside	you	we	some	any
against	into	he	they	this	each
along	near	me	us	that	no
around	of	her	them	every	half
at	off	him	our/ours	all	twice
behind	on	my	their/theirs	both	two
beside	onto	mine	itself	one	second
besides	over	her	ourselves	first	another
by	through	hers	themselves	other	last
despite	to	his	something	next	few
down	toward	myself	nothing	many	little
during	with	himself	someone	much	less
for	within	herself		more	least
from	without	anything		most	own
		everything		several	
		anyone		no	
		everyone			
		ones			
		such			

Conjunctions		Modal Verbs		Adverbs	
and	or	can	might	here	therefore
but	so	may	would	there	however
after	before	will	should	today	besides
when	since	shall	must	tomorrow	moreover
as	while	could		now	though
because	although			then	otherwise
if	though	Primary Verbs		always	else
what	who	be		never	instead
where	whose	do		sometimes	anyway
which		have		usually	incidentally
how				often	meanwhile
than					

Thus, there is need to develop a precise and refined model for web service classification, which has a strong pre-processing base, and a concise set of WSDL features. Because selecting a large set of features or a very limited set greatly impacts accuracy of classification.

Chapter 3

PROPOSED CLASSIFICATION APPROACH

The main aim of this research is to provide a compact web service classification approach based on pre-processed and mined information from WSDL files. Therefore, in this chapter, we present a detailed description of our approach and explain that how we carried out pre-processing of features extracted from WSDL file. Then finally, how data is classified into categories.

3.1 Overview

We build our system using Java Platform SE 7 in Eclipse. Our proposed system mainly consists of three phases.

1. WSDL Feature Extractor
2. Text Pre-processor
3. Web Service Classifier

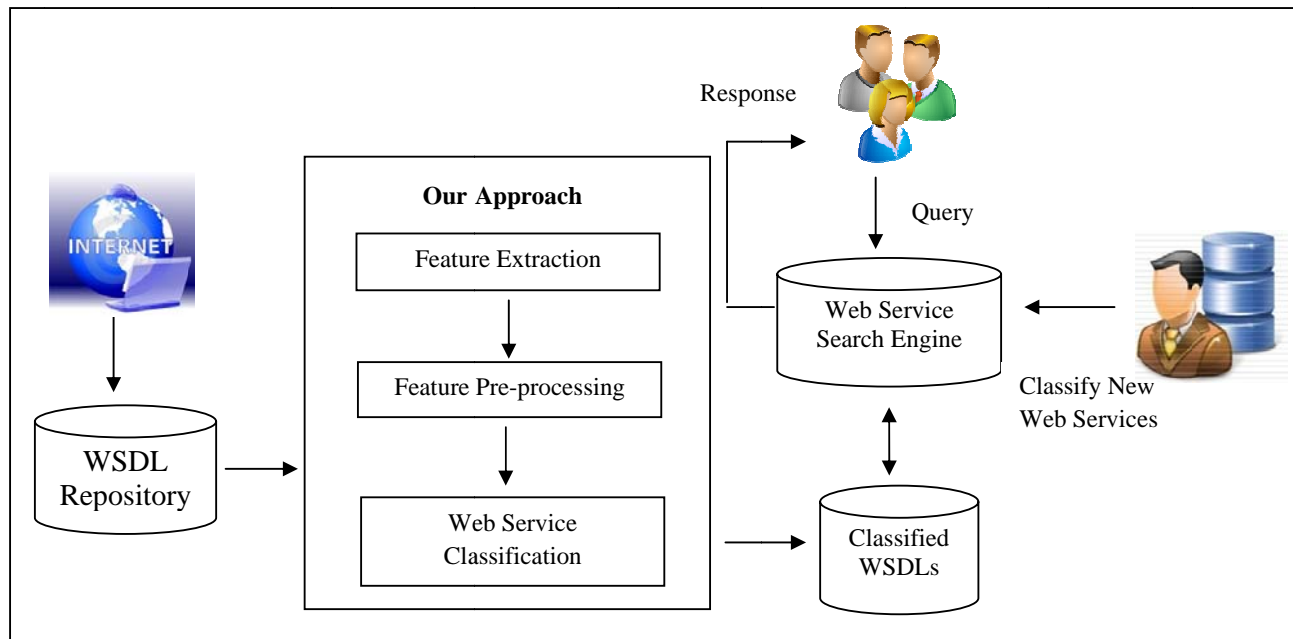


Figure 3.1: Schematic block diagram illustrating architecture of our classification approach

Figure 3.1 illustrates architectural model of our proposed approach to aid web service search engines in classifying WSDL documents. WSDL documents are downloaded from internet and offline pre-processing is performed to classify them into functionally similar groups. A WSDL document may be assigned to multiple categories (Ranking Classification) but proposed approach in this thesis assign single category to each WSDL document (Hard Classification). Search engines can use this approach to build up their web service repository by determining domain or group of newly arrived web services.

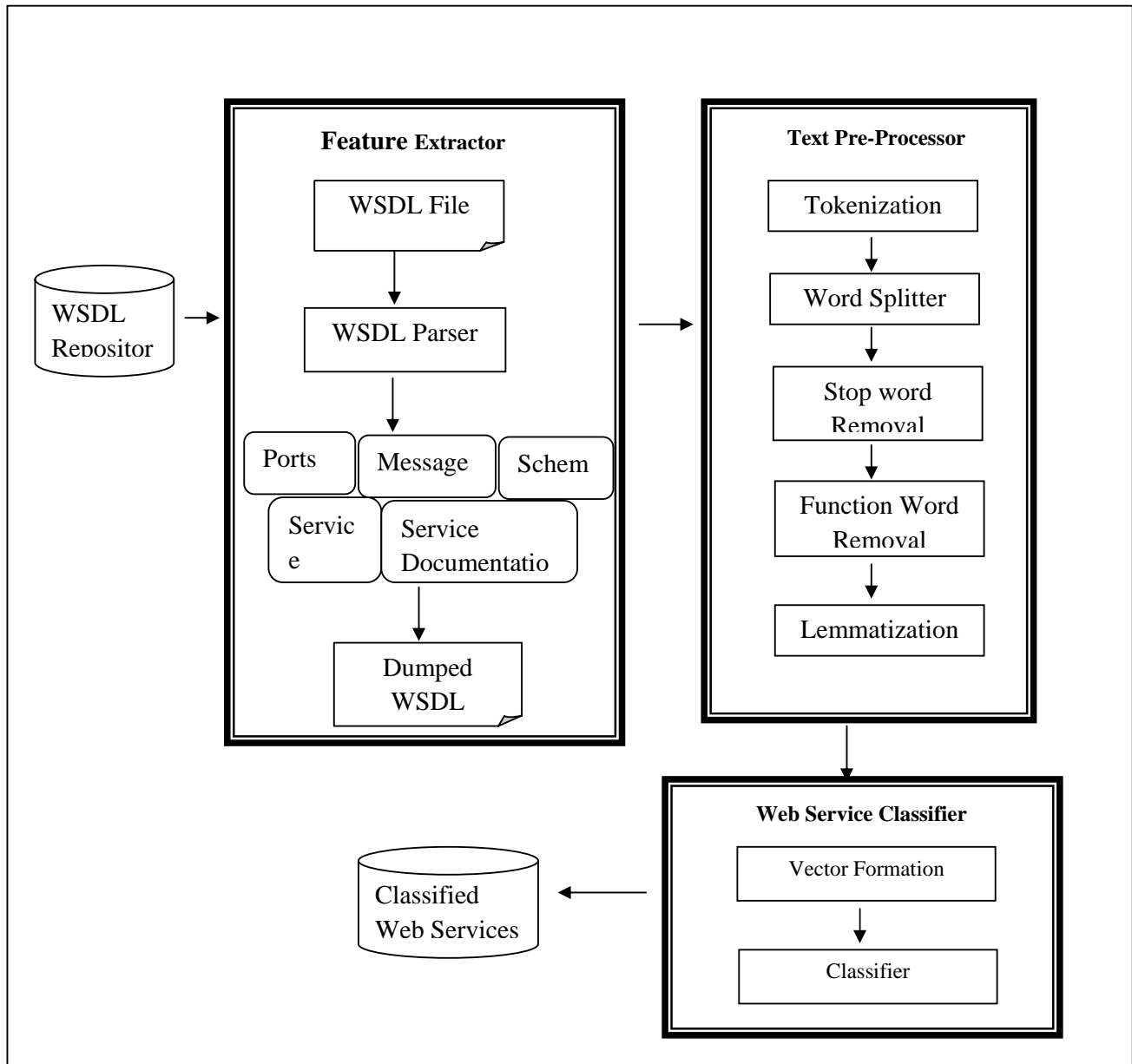


Figure 3.2: Framework of web service classification

Figure 3.2 illustrates steps during each phase of classification approach. Following sections give a detail understanding of each phase.

3.2 Feature Extractor

We started by reading WSDL documents from WSDL repository. WSDL documents will be processed for extracting relevant features. In this phase, contents of WSDL document are parsed to extract

1. Service name
2. Service documentation
3. WSDL schema
4. WSDL messages
5. WSDL port types.

Specific and selected information is extracted from each element of WSDL documents.

Complex Types are the most informative element in WSDL documents [9]. Each element of WSDL schema consists of name and type attribute. However element type attribute is not being used as a source of information in our classification approach. This differ our approach from Khalid Elgazzar's [3] where element types are extracted to find similarity between pair of web services. Following information from WSDL schema is extracted as part of WSDL content.

1. Name attribute of complex type.
2. Documentation content of element in sequence of complex type.
3. Documentation content of complex types.
4. Documentation content of elements in schema.

Figure 3.3 shows a small portion of WSDL schema extracted from *WaterService* web service (provided in Appendix A.1).

```

InputParameters Input parameters for the tool
Show Parameters [Show parameters in output alignment, as in genewise.]
Show Pretty ASCII [Show pretty ASCII alignment viewing, as in genewise.] First
Sequence [The first DNA sequence to be aligned can be entered directly into the form.
The sequence must be in a recognized format eg. GCG, FASTA, EMBL, GenBank.
Partially formatted sequences are not accepted. Adding a return to the end of the
sequence may help certain applications understand the input. Note that directly using
data from word processors may yield unpredictable results as hidden/control
characters may be present. There is a limit of 1MB for the sequence entry.] Second
Sequence [The second DNA sequence to be aligned can be entered directly into the
form. The sequence must be in a recognized format eg. GCG, FASTA, EMBL,
GenBank. Partially formatted sequences are not accepted. Adding a return to the end
of the sequence may help certain applications understand the input. Note that directly
using data from word processors may yield unpredictable results as hidden/control
characters may be present. There is a limit of 1MB for the sequence entry.]
ArrayOfString wsResultTypes List of renderers available to output the result of the
job wsResultType Details about a renderer used to output the result of the job A short
description of the renderer A suggested file suffix to be used when saving the data
formatted by the renderer .....

```

Figure 3.3: WSDL Schema extracted from WaterService web service

From each message element, following information is extracted.

1. Name attribute of part.
2. Element attribute of part.

Figure 3.4 shows WSDL messages extracted from *WaterService* web service.

```

runResponse tns:runResponse runRequest tns:run
getStatusResponse tns:getStatusResponse
getStatusRequest tns:getStatus getResultTypesResponse
tns:getResultTypesResponse getResultTypesRequest
tns:getResultTypes getResultResponse
tns:getResultResponse getResultRequest tns:getResult
getParameterDetailsResponse
tns:getParameterDetailsResponse
getParameterDetailsRequest tns:getParameterDetails
getResponse tns:get Response get Request tns:get

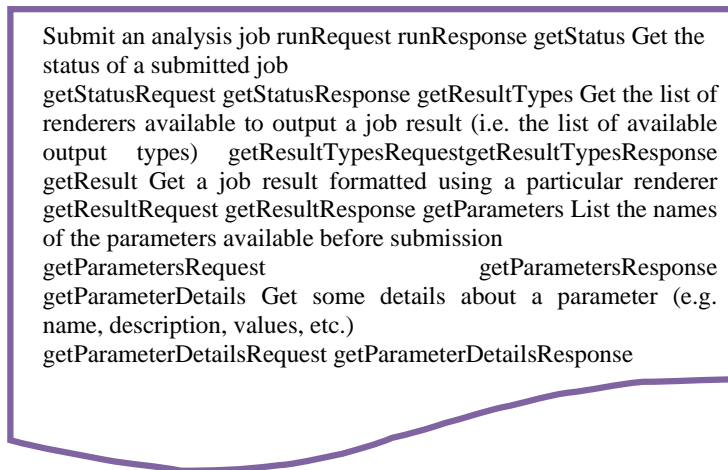
```

Figure 3.4: WSDL Messages extracted from WaterService web service

Information extracted from port types include:

1. Name attribute of portType.
2. Documentation content of portType.
3. Name attribute of each operation in portType.
4. Documentation content of each operation in portType.
5. Name attribute of input/output parameters in each operation.

Figure 3.5 shows WSDL port types extracted from *WaterService* web service.



```
Submit an analysis job runRequest runResponse getStatus Get the
status of a submitted job
getStatusRequest getStatusResponse getResultTypes Get the list of
renderers available to output a job result (i.e. the list of available
output types) getResultTypesRequest getResultTypesResponse
getResult Get a job result formatted using a particular renderer
getResultRequest getResultResponse getParameters List the names
of the parameters available before submission
getParameterRequest getParametersResponse
getParameterDetails Get some details about a parameter (e.g.
name, description, values, etc.)
getParameterDetailsRequest getParameterDetailsResponse
```

Figure 3.5: WSDL PortTypes extracted from WaterService web service

This information including *Service Name* and *Service Documentation* is dumped into a text file. These dumped WSDL contents are used as a base for web service classification. Null values of any attribute are not included.

WSDL documents are parsed using *Membrane SOA Model* [21]. It is a Java API for reading and analyzing XML Schema and WSDL documents.

3.3 Text Pre-processor

The goal of pre-processing phase is to enhance the quality of information available for classification. Information in text file might be inconsistent and may lead mining process to inaccurate results. During this phase different pre-processing steps are applied to extract accurate and consistent information. The end product of this phase is a text file which is

efficient in terms of both time and space and maintains good information retrieval (IR) performance.

We use following steps for pre-processing on dumped WSDL contents.

3.3.1 Tokenization

Tokenization caters for consistency in textual information. Java string tokenization is used to convert block of characters into dataset of words. All the former pre-processing steps require dataset of words.

3.3.2 Word Splitter

Word splitter performs splitting of concatenated words based on their case. For example, a web service has operation named *ValidateAddressResponse*. This name is meaningless unless it is split into words *validate*, *Address* and *Response*. Word splitter is introduced as a new pre-processing step which is not used by existing web service clustering and classification techniques [3, 9, 10, 12, and 14].

3.3.3 Stop Word Removal

The next step in pre-processing is filtering stop words from textual information. Stop list contains prepositions and articles which are insignificant and can be easily removed from document. SMART stop word list [22] is used to eliminate stop words.

3.3.4 Function Word Removal

Stop words list typically eliminates function words but there are a few function words which are not stop words. This step eliminates rest of function words by performing comparison of SMART stop word list [22] and publically available function word list [23].

3.3.5 Lemmatization

The last step of pre-processing is lemmatization. Lemmatization is performed using *Stanford CoreNLP*. Stanford CoreNLP is an integrated framework that provides a suite of tools for natural language processing [24].

We have used lemmatization instead of stemming for converting words to their base forms. Web service classification results show that lemmatization significantly

improves precision and recall. Figure 4.6 shows a small portion of *WaterService* after pre-processing.

```
Water Service block aligner align sequence assumption sequence share number
colinear block conservation separated potentially large vary length sequence
Dispatcher Service run submit analysis job run Request run response Status get
status submit job get Status Request Status Response result type get list renderer
available output job result list available output type get result type Request
result type response result get job result format use particular renderer get result
request result response parameter list name parameter available submission get
Parameter Request Parameter Response Parameter Detail get detail parameter
description value get Parameter Detail Request Parameter Detail Response run
response parameter tn run response run request parameter tn run Status
Response parameter show Pretty.
```

Figure 3.6: Information in WaterService after pre-processing phase.

3.4 Web Service Classifier

During this phase web services are classified into functionally similar categories. Automatic classification of web services not only help in service publication (for classification of new services) but also help in service retrieval (to focus user queries to a limited set of related web service categories)

Web service classifier is implemented using *MALLET* (MAchine Learning for Language Toolkit) [25]. It is an open source Java library for natural language processing, document clustering and classification. The classification of web services into domain specific groups is performed using algorithm *Maximum Entropy*. But before applying Maximum Entropy, MALLET maps information to be classified into feature vectors. So, this phase consists of two steps.

3.4.1 Vector Formation

Prior to performing classification with MALLET, pre-processed textual information of web services is converted into a list of feature vectors. A single list may contain complete data which is split at classification time into testing and training portions or user can create two separate lists for testing and training data. In our approach we are using single list of feature vectors.

One feature vector is generated per document, each word in document represent a dimension in vector and vector value at each position is sum of word occurrences in the document.

3.4.2 Classifier

After creation of feature vectors list, MALLET is used to perform web service classification. MALLET supports different classification algorithms including Naive Bayesian, Maximum Entropy and Decision Tree.

Maximum Entropy gives less classification error as compared to Naive Bayesian [26]. We perform classification of dataset using Maximum Entropy and Naïve Bayesian and statistics indicate that Maximum Entropy gives more accurate results with less standard error. So we use Maximum Entropy as our supervised classification algorithm. Table 3.1 shows classification results of categories during trial 3 of Percentage Split results. Classification results for all trials of Percentage Split are included in Appendix C.

Table 3.1: Confusion Matrix for Classification of Dataset Categories

Category			Predicted Results							Total	
			0	1	2	3	4	5	6		7
Actual Results	0	Address And Location	9	0	0	0	0	0	0	0	9
	1	Currency, Stock, Finance	2	5	0	0	0	0	0	0	7
	2	Data Retrieval	0	0	29	0	1	3	0	0	33
	3	Fax And Messaging	1	0	0	3	0	2	0	0	6
	4	Graphics And Multimedia	0	0	0	0	5	0	0	1	6
	5	Microarrays	0	0	3	0	0	10	0	0	13
	6	Sequence Analysis	0	0	0	0	0	2	20	0	22
	7	Value Manipulation And Unit Converter	0	0	0	0	0	0	0	7	7

In this thesis, we have proposed a modified approach for web service classification. This chapter clearly specifies which properties of WSDL documents contribute to form WSDL content feature and how these contents are pre-processed to remove unwanted textual information. During the last phase, classifier is trained and new web services are classified into functionally similar groups.

Chapter 4

CLASSIFICATION RESULTS AND COMPARISON

4.1 Overview

Now when we have already gone through feature extraction, pre-processing and finally classification steps, this chapter will focus on evaluating the effectiveness of our classification approach with reference to its comparison with existing techniques.

4.2 Data Preparation

In our approach, WSDL documents are used as main source of data repository because web service is completely described by information carried by these XML files. However, retrieving adequate and suitable WSDL files from Internet to satisfy needs of repository was a hard task.

Although a number of UDDI registries exist but often the set of services retrieved is useless. Most of the obtained services have dummy documentation and are trivial. We collect real-world web services from Internet. These web services are downloaded from web service publisher websites such as <http://www.xmethods.net>, <http://www.webservicex.net>, www.biocatalogue.org and <http://www.webservicelist.com>. A set of 350 web services is composed and manually classified into eight categories to serve as a comparison point for our classification approach. The data is organized in directories, one directory for each category. Table 4.1 gives names of categories and web services included in each category.

Table 4.1: Dataset Categories and their web services

Sr. No.	Category	No. of Web Services
1.	Address and Location	44
2.	Currency and Stock	41
3.	Fax and Messaging	20

4.	Graphics and Multimedia	21
5.	Sequence Analysis	88
6.	Value Manipulation and Unit Converter	20
7.	Microarray	25
8.	Data Retrieval	93

Web services are collected from different sources instead of single source to keep the concept generic.

4.3 Experimental Results

After being done with all text mining process, classification is performed using Maximum Entropy with Gaussian Prior Variance equals to 1.0.

Maximum entropy needs to be trained with pre-classified dataset of web services. In our case, feature vectors of 350 web services are formed and then classification is performed using Maximum entropy. Ten folds cross validation is used to determine accuracy of algorithm. Table 4.2 shows mean precision and recall values for each category in dataset.

Table 4.2: Mean precision and recall for classification of dataset categories

No.	Category	Precision	Recall
1	Address And Location	81.02%	83.13%
2	Currency, Stock, Finance	85.36%	81.52%
3	Data Retrieval	91.54%	89%
4	Fax And Messaging	95.91%	74.53%
5	Graphics and Multimedia	70.5%	63.85%
6	Microarrays	80%	84.55%
7	Sequence Analysis	99.98%	100%
8	Value Manipulation and Unit Converter	95.488%	93.0%

Precision and recall has been calculated using following equations.

$$\textit{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$\textit{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

The relationship of terms True Positive, True Negative, False Positive and False Negative is illustrated in Table 4.3.

Table 4.3: Relationship between Positive and Negative Results

	Predicted Category (Observation)	
Actual Category (Expectation)	True Positive (Correct result)	False Negative (Missing result)
	False Positive (Unexpected result)	True Negative (Correct absence of result)

A snapshot of cross validation results is shown in Appendix B. Table 4.4 shows mean accuracy, standard error and standard deviation of test and training data.

Table 4.4: Statistics of Training and Test Data

Data Set	Mean Accuracy (%)	Standard Error (%)	Standard Deviation (%)
Training Set	100	0.00	0.00
Testing Set	88.60	1.679	5.30

4.4 Comparison to Existing Classification and Clustering Techniques

A hefty list of existing web service classification and clustering techniques was presented in chapter 2. All of them focus on different features in WSDL document as depicted in Table 2.1 and almost similar steps for pre-processing information. The main driving force of this research work was to reduce feature set used for classification and to provide a more compact and accurate pre-processing approach. Secondly Maximum Entropy is being used for text classification [26] but it has not been applied yet for classification and clustering of web services. Utilizing Maximum Entropy for web service classification has significantly improved results.

The main difference of our approach with existing approaches is that it reduces feature set by using only one feature (WSDL contents) for web service classification.

Table 4.5 shows comparison of our approach with techniques [10, 14, and 15].

Table 4.5: Performance comparison with existing techniques

Approach	Accuracy	No of classified/clustered categories	Tested web services
An Approach to support Web Service Classification and Annotation [10]	Propose nearest three classes out of 11 with likelihood of 83%	11	205
Clustering of web services based on semantic similarity [14]	70%	5	8
Web Service Classification [15]	30%-40%	8	500
Our Approach	88.6%	8	350

In order to compare our approach with [3, 12], we created a dataset of web services belonging to five categories *Address Validation*, *Credit Card Check*, *Currency Exchange*, *Email verification* and *Weather* [3]. This is because these approaches lack implementation details and do not specify mean accuracy of result. However, they have specified precision and recall

for above mentioned web service categories. Most of these web services are obsolete and their WSDL documents are not available so only 38 valid WSDL files were downloaded. Table 4.6 illustrates performance comparison for Liu and Wong’s [12] and Khalid Elgazzar’s [3] approach versus our approach relative to five manually identified categories in dataset.

Table 4.6: Performance evaluation related to five identified categories

Category	Wei Liu, Wilson Wong Approach [12]		Khalid Elgazzar, Ahmed E. Hassan, Patrick Martin Approach [3]		Our Approach	
	Precision	Recall	Precision	Recall	Precision	Recall
Currency Exchange	84.2%	88.9%	90%	94.7%	92.8%	89%
Weather	70%	87.5%	94.1%	100%	93.98%	100%
Address Validation	60%	93.7%	83.3%	93.7%	67%	80%
Email Verification	58.3%	87.5%	80%	100%	87.5%	88%
Credit Card Services	60%	90%	90%	90%	75%	50%

We have noted that low precision and recall of our approach for “*Credit Card Services*” and “*Address Validation*” categories in Table 4.6 is due to unavailability of enough training data. Our approach has high precision and recall in case of “*Currency Exchange*”, “*Weather*” and “*Email Verification*” categories. This indicates that our approach performs well by only selecting one feature (WSDL contents) for web service classification. The impact of using modified pre-processing approach on WSDL contents has improved classification reliability by enhancing recall and precision.

4.6 Analysis of Results

Looking at the performance of our approach and existing web service clustering and classification approaches, we have noted that our approach yields high precision and recall for all identified categories.

When compared with approach [12], we have reduced the feature set from four features to one feature. Liu and Wong's used feature WSDL content but they do not clearly specify whether WSDL contents refers to documentation content in each tag or names of attributes. However our approach clearly specified which attributes and documentation contents contribute to WSDL content feature. Secondly different types of web services are published by providers on same web site so using service context and service host name features might mislead meaning of web service. So these two features does are not selected in our approach and results are improved significantly.

Khalid Elgazzar's [3] improved Liu and Wong's approach [12] by using a different set of features by replacing service context and service host features with WSDL types, messages and ports. WSDL contents include all contents of WSDL file after tag removal. Compared with our approach we do not include all the information in WSDL file as part of WSDL contents. We extract service name, service documentation, WSDL types, messages, and ports and combine them to form WSDL contents instead of using them separately. Khalid Elgazzar's used type attribute of WSDL types instead of using name attribute but in our approach we have used name attribute. Type attribute might be misleading because two web services belonging to same category might have different input types for same type of operations. For example, operations of a zipcode web service might take zipcode as a number while another zipcode web service might takes a string as parameter. Also Khalid Elgazzar's applied pre-processing techniques to only WSDL content feature; however a clear examination of WSDL ports, messages and types reveal that pre-processing is also required to mine accurate information from these features as well.

When compared with existing techniques, our approach suggested two new steps in pre-processing WSDL contents. Firstly we replaced stemming with *lemmatization* and proved that there is improvement in precision and recall. Secondly we introduce *Word Splitter* which split the compound words on the basis of their case. It is a common observation that most of the web service developers use compound words for operation names, messages and complex types. So Word Splitter plays an important role in improving classification results of our

approach. Aparna Konduri [14] performed classification of web services based on semantic similarity between web service operations and operation names. However, this approach yields low accuracy because of using limited set of WSDL information.

Our approach uses Maximum Entropy for web service classification. Maximum Entropy has been used for text categorization but so far it has not been used for classifying web services. This makes our approach comparable with Macello Brunu [10] approach. He suggested a classification approach using SVM but uses only a limited set of information from WSDL document resulting in low accuracy of classification. This approach does not always locate correct category of web service but often indicate a limited group of categories to which web service might belong.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Overview

Web Services are very common nowadays because of extensive use of Service Oriented Architecture (SOA) to support business processes. Application developers utilize existing web services to compose new custom web services. Therefore finding the desired set of web services is becoming a challenging and emergent research problem.

Thus, we dedicated ourselves to provide a web service classification approach that helps user to get web services of their desired domain and also allows search engines to enhance their repositories by classifying new set of web services.

5.2 Conclusion

In this research work, we have described the whole process of WSDL parsing, pre-processing and classification. The variation of this approach with existing techniques lies in selecting different WSDL feature, varied pre-processing methodology and Maximum Entropy classifier for web service classification.

Our approach starts with reading and parsing WSDL documents, specific information from WSDL definition, WSDL documentation, WSDL messages, WSDL portTypes and WSDL schema is extracted to form WSDL content feature. This feature is passed through various pre-processing steps including tokenization, word splitter, stop word removal, function word removal and at the end lemmatizer. Information from lemmatizer is fed into classifier which create feature vector of information and pass all feature vectors of WSDL documents to Maximum Entropy classifier. Maximum Entropy splits vectors into training and testing datasets and performs classification of web services into pre defined categories.

Classification of web service into functionally similar categories can effectively reduce the headache of search engines. Our proposed approach text mine different types of information from WSDL file and use this information for effective web service classification. In order to realize the approach, different challenging tasks such as dataset collection, measuring effectiveness of different WSDL parsers and lemmatizers and then selecting the appropriate one are resolved.

Effective web service classification is an important issue for non-semantic web services. In this thesis, we propose a machine learning approach that improves classification of non-semantic web services by performing mining of WSDL contents. This approach has reduced the feature set used by existing techniques for web service classification and clustering. Our classification results proves that instead of using different WSDL features separately and assigning them different weight, careful selection of suitable attributes from WSDL document and using them collectively as a single feature can still improves classification accuracy.

All the existing techniques utilize stemming for pre-processing WSDL contents, but we have used Lemmatization and Word Splitter for pre-processing. These two steps enhances accuracy of our suggested classification model. This is evident by comparison with existing techniques in Chapter 4.

This approach is validated on dataset of 350 web services manually categorized into 8 categories namely Address and Location, Currency and Stock, Fax and Messaging, Graphics and Multimedia, Sequence Analysis, Value Manipulation and Unit Converter, Microarray and Data Retrieval yielding accuracy up to 88%. Experiments show that quality of retrieved information is improved as compared with existing approaches.

Our classification approach can be used by web service search engines for classification of new services as well as to focus user queries to a refined set of web service categories.

5.3 Future Work

As future work, we plan to use same pre-processing approach for classification of semantic web services and classification of web pages. Another improvement is the use of Support Vector Machine (SVM) algorithm for web service classification with same pre-processing approach. This allows comparison of Maximum Entropy and Support Vector Machine algorithms for classification of web services.

At present, there is no universal dataset available for classification of web services. Our technique and all the existing techniques downloaded their own set of web services for verification of their results. However, a better and more accurate comparison can be performed if a standard set of web services is available. More work can be performed in future on our approach when such a standard dataset is available.

Last but not least, additional pre-processing capabilities such as converting abbreviation to their augmentations, replacing misspelled words with the correct ones etc. can be implemented to provide more accurate mining of WSDL contents.

APPENDICES

Appendix A.1: Sample WSDL File of Water Service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  name="WaterService"
  targetNamespace="http://soap.jdispatcher.ebi.ac.uk"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://soap.jdispatcher.ebi.ac.uk"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:documentation>DNA Block Aligner (DBA) aligns two sequences under
  the assumption that the sequences share a number of colinear blocks of
  conservation separated by potentially large and varied lengths of DNA in
  the two sequences.
  </wsdl:documentation>
  <wsdl:types>
  <xsd:schema xmlns="http://soap.jdispatcher.ebi.ac.uk"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="unqualified" elementFormDefault="unqualified"
  targetNamespace="http://soap.jdispatcher.ebi.ac.uk">
  <xsd:complexType name="InputParameters">
  <xsd:annotation>
  <xsd:documentation xml:lang="en"> Input parameters for the
  tool</xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
  <xsd:element minOccurs="0"
  maxOccurs="1" name="para"
  nillable="true" type="xsd:boolean">
  <xsd:annotation>
  <xsd:documentation xml:lang="en">Show Parameters [Show parameters in output
  alignment, as in genewise.]</xsd:documentation>
  </xsd:annotation>
  </xsd:element>
  <xsd:element minOccurs="0"
  maxOccurs="1" name="pretty"
  nillable="true" type="xsd:boolean">
  <xsd:annotation>
  <xsd:documentation xml:lang="en">Show Pretty ASCII [Show pretty ASCII
  alignment viewing, as in genewise.]</xsd:documentation>
  </xsd:annotation>
  </xsd:element>
  <xsd:element minOccurs="0"
  maxOccurs="1" name="asequence"
  nillable="true" type="xsd:string">
  <xsd:annotation>
  <xsd:documentation xml:lang="en">First Sequence [The first DNA sequence to
  be aligned can be entered directly into the form. The sequence must be in a
  recognised format eg. GCG, FASTA, EMBL, GenBank. Partially formatted
  sequences are not accepted. Adding a return to the end of the sequence may
  help certain applications understand the input. Note that directly using
  data from word processors may yield unpredictable results as hidden/control
  characters may be present. There is a limit of 1MB for the sequence entry.]
  </xsd:documentation>
```

```

</xsd:annotation>
</xsd:element>
<xsd:element minOccurs="0"
maxOccurs="1" name="bsequence"
nillable="true" type="xsd:string">
<xsd:annotation>
<xsd:documentation xml:lang="en">Second Sequence [The second DNA sequence
to be aligned can be entered directly into the form. The sequence must be
in a recognised format eg.GCG, FASTA, EMBL, GenBank. Partially formatted
sequences are not accepted. Adding a return to the end of the sequence may
help certain applications understand the input.Note that directly using
data from word processors may yield unpredictable results as hidden/control
characters may be present. There is a limit of 1MB for the sequence entry.]
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ArrayOfString">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="string"
nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="wsResultTypes">
<xsd:annotation>
<xsd:documentation xml:lang="en">List of renderers available to output the
result of the job</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="type"
type="wsResultType"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="wsResultType">
<xsd:annotation>
<xsd:documentation xml:lang="en">Details about a renderer used to output
the result of the job</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element maxOccurs="1" minOccurs="0" nillable="true" name="description"
type="xsd:string">
<xsd:annotation>
<xsd:documentation xml:lang="en">A short description of the
renderer</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="1" nillable="false" name="fileSuffix"
type="xsd:string">
<xsd:annotation>
<xsd:documentation xml:lang="en">A suggested file suffix to be used when
saving the data formatted by the renderer</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="1" nillable="false" name="identifier"
type="xsd:string">
<xsd:annotation>
<xsd:documentation xml:lang="en">The renderer identifier to be used when
invoking the getResult() method</xsd:documentation>

```



```

</xsd:annotation>
</xsd:element>
<wsdl:message name="runResponse">
<wsdl:part name="parameters" element="tns:runResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="runRequest">
<wsdl:part name="parameters" element="tns:run"></wsdl:part>
</wsdl:message>
<wsdl:message name="getStatusResponse">
<wsdl:part name="parameters" element="tns:getStatusResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getStatusRequest">
<wsdl:part name="parameters" element="tns:getStatus"></wsdl:part>
</wsdl:message>
<wsdl:message name="getResultTypesResponse">
<wsdl:part name="parameters"
element="tns:getResultTypesResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getResultTypesRequest">
<wsdl:part name="parameters" element="tns:getResultTypes"></wsdl:part>
</wsdl:message>
<wsdl:message name="getResultResponse">
<wsdl:part name="parameters" element="tns:getResultResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getResultRequest">
<wsdl:part name="parameters" element="tns:getResult"></wsdl:part>
</wsdl:message>
<wsdl:message name="getParameterDetailsResponse">
<wsdl:part element="tns:getParameterDetailsResponse"
name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:message name="getParameterDetailsRequest">
<wsdl:part element="tns:getParameterDetails" name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:message name="getParametersResponse">
<wsdl:part element="tns:getParametersResponse"
name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:message name="getParametersRequest">
<wsdl:part element="tns:getParameters" name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:portType name="JDispatcherService">
<wsdl:operation name="run">
<wsdl:documentation>Submit an analysis job</wsdl:documentation>
<wsdl:input name="runRequest" message="tns:runRequest"></wsdl:input>
<wsdl:output name="runResponse" message="tns:runResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getStatus">
<wsdl:documentation>Get the status of a submitted job</wsdl:documentation>
<wsdl:input name="getStatusRequest"
message="tns:getStatusRequest"></wsdl:input>
<wsdl:output name="getStatusResponse"
message="tns:getStatusResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getResultTypes">
<wsdl:documentation>Get the list of renderers available to output a job
result (i.e. the list of available output types)</wsdl:documentation>
<wsdl:input name="getResultTypesRequest"
message="tns:getResultTypesRequest"></wsdl:input>

```

```

<wsdl:output name="getResultTypesResponse"
message="tns:getResultTypesResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getResult">
<wsdl:documentation>Get a job result formatted using a particular
renderer</wsdl:documentation>
<wsdl:input name="getResultRequest"
message="tns:getResultRequest"></wsdl:input>
<wsdl:output name="getResultResponse"
message="tns:getResultResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getParameters">
<wsdl:documentation>List the names of the parameters available before
submission</wsdl:documentation>
<wsdl:input message="tns:getParametersRequest"
name="getParametersRequest"></wsdl:input>
<wsdl:output message="tns:getParametersResponse"
name="getParametersResponse"></wsdl:output>
</wsdl:operation>
<wsdl:operation name="getParameterDetails">
<wsdl:documentation>Get some details about a parameter (e.g. name,
description, values, etc.)</wsdl:documentation>
<wsdl:input message="tns:getParameterDetailsRequest"
name="getParameterDetailsRequest"></wsdl:input>
<wsdl:output message="tns:getParameterDetailsResponse"
name="getParameterDetailsResponse"></wsdl:output>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="JDispatcherServiceHttpBinding"
type="tns:JDispatcherService">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="run">
<wsdl:documentation>Submit an analysis job</wsdl:documentation>
<soap:operation soapAction="urn:Run" />
<wsdl:input name="runRequest">
<soap:body use="literal" />
</wsdl:input>
<wsdl:output name="runResponse">
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:service name="JDispatcherService">
<wsdl:port name="JDispatcherServiceHttpPort"
binding="tns:JDispatcherServiceHttpBinding">
<soap:address location="http://www.ebi.ac.uk/Tools/services/soap/wise2dba"
/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Appendix A.2: Cross Validation Results

----- Trial 0 -----

Trial 0 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
Trial 0 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
1.0
Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.9411764705882353

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	4	4
1	Currency,Stock,Finance	.	1	.	.	1	.	.	.	2
2	Data Retrieval	.	.	5	.	.	1	.	.	6
3	Fax And Messaging	.	.	.	4	4
4	Graphics and Multimedia	3	.	.	.	3
5	Microarrays	2	.	.	2
6	Sequence Analysis	9	.	9
7	Value Manipulation and Unit Convertor	4	4

Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=
0.9411764705882353

----- Trial 1 -----

Trial 1 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
Trial 1 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
1.0
Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8529411764705882

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	1	1	2
1	Currency,Stock,Finance	.	3	3
2	Data Retrieval	.	.	11	.	1	1	.	.	13
3	Fax And Messaging	.	.	1	2	3
4	Graphics and Multimedia	2	.	.	1	3
5	Microarrays	2	.	.	2
6	Sequence Analysis	1	7	.	8
7	Value Manipulation and Unit Convertor	0

Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=
0.8529411764705882

----- Trial 2 -----

Trial 2 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 307 instances
Trial 2 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
1.0
Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8571428571428571

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	4	4
1	Currency,Stock,Finance	.	4	1	5

2	Data Retrieval	.	.	9	.	.	3	.	.	12
3	Fax And Messaging	.	.	.	3	3
4	Graphics and Multimedia	1	.	.	.	1
5	Microarrays	.	.	1	.	.	2	.	.	3
6	Sequence Analysis	6	.	6
7	Value Manipulation and Unit Convertor	1	1

Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8571428571428571

----- **Trial 3** -----

Trial 3 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
 Trial 3 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
 Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.7647058823529411

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	1	.	.	.	1	1	.	.	3
1	Currency,Stock,Finance	1	5	1	.	1	.	.	.	8
2	Data Retrieval	.	1	5	.	.	1	.	.	7
3	Fax And Messaging	.	.	.	1	1
4	Graphics and Multimedia	2	.	.	.	2
5	Microarrays	.	.	1	1
6	Sequence Analysis	1	10	.	11
7	Value Manipulation and Unit Convertor	1	1

Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.7647058823529411

----- **Trial 4** -----

Trial 4 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
 Trial 4 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
 Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.8529411764705882

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	6	6
1	Currency,Stock,Finance	.	1	1
2	Data Retrieval	1	.	11	.	.	1	.	.	13
3	Fax And Messaging	.	.	.	1	1
4	Graphics and Multimedia	1	.	.	1
5	Microarrays	3	.	.	3
6	Sequence Analysis	3	.	3
7	Value Manipulation and Unit Convertor	1	1	4	6

Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8529411764705882

----- **Trial 5** -----

Trial 5 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
 Trial 5 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished

Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix Confusion Matrix, row=true, column=predicted accuracy=0.9411764705882353

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	2	1	3
1	Currency,Stock,Finance	.	4	4
2	Data Retrieval	.	.	10	10
3	Fax And Messaging	.	.	1	3	4
4	Graphics and Multimedia	0
5	Microarrays	3	.	.	3
6	Sequence Analysis	10	.	10
7	Value Manipulation and Unit Convertor	0

Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.9411764705882353

----- Trial 6 -----

Trial 6 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances

Trial 6 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished

Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix Confusion Matrix, row=true, column=predicted accuracy=0.9117647058823529

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	6	.	.	.	1	.	.	.	7
1	Currency,Stock,Finance	1	3	4
2	Data Retrieval	.	.	8	8
3	Fax And Messaging	.	.	.	1	1
4	Graphics and Multimedia	3	.	.	.	3
5	Microarrays	.	.	1	.	.	2	.	.	3
6	Sequence Analysis	7	.	7
7	Value Manipulation and Unit Convertor	1	1

Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.9117647058823529

----- Trial 7 -----

Trial 7 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 307 instances

Trial 7 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished

Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix Confusion Matrix, row=true, column=predicted accuracy=0.8857142857142857

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	5	5
1	Currency,Stock,Finance	.	3	3
2	Data Retrieval	.	.	8	8
3	Fax And Messaging	.	.	.	2	2
4	Graphics and Multimedia	.	1	2	.	.	1	.	.	4
5	Microarrays	2	.	.	2
6	Sequence Analysis	10	.	10
7	Value Manipulation and Unit Convertor	1	1

Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8857142857142857

----- Trial 8 -----

Trial 8 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
 Trial 8 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.9411764705882353

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	5	1	.	.	6
1	Currency,Stock,Finance	1	3	4
2	Data Retrieval	.	.	7	7
3	Fax And Messaging	.	.	.	1	1
4	Graphics and Multimedia	3	.	.	.	3
5	Microarrays	2	.	.	2
6	Sequence Analysis	7	.	7
7	Value Manipulation and Unit Convertor	4	4

Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.9411764705882353

----- Trial 9 -----

Trial 9 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 308 instances
 Trial 9 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.9117647058823529

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	4	4
1	Currency,Stock,Finance	1	6	7
2	Data Retrieval	.	.	8	.	.	1	.	.	9
3	Fax And Messaging	0
4	Graphics and Multimedia	1	.	.	.	1
5	Microarrays	.	.	1	.	.	3	.	.	4
6	Sequence Analysis	7	.	7
7	Value Manipulation and Unit Convertor	2	2

Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.9117647058823529

MaxEntTrainer,gaussianPriorVariance=1.0

Summary. train accuracy mean = 1.0 stddev = 0.0 stderr = 0.0

Summary. test accuracy mean = 0.8860504201680672 stddev = 0.05309594557478448 stderr = 0.016790412253665708

Appendix A.3: Percentage Split Results

----- Trial 0 -----

Trial 0 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
Trial 0 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8932038834951457

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	11	1	12
1	Currency,Stock,Finance	.	13	13
2	Data Retrieval	.	2	23	.	1	2	.	.	28
3	Fax And Messaging	1	.	.	5	1	.	.	.	7
4	Graphics and Multimedia	1	.	.	.	4	.	.	.	5
5	Microarrays	.	.	1	.	.	9	.	.	10
6	Sequence Analysis	24	.	24
7	Value Manipulation and Unit Convertor	1	3	4

Trial 0 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8932038834951457

----- Trial 1 -----

Trial 1 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
Trial 1 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8737864077669902

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	9	9
1	Currency,Stock,Finance	2	5	7
2	Data Retrieval	.	.	29	.	1	3	.	.	33
3	Fax And Messaging	1	.	.	3	.	2	.	.	6
4	Graphics and Multimedia	5	.	.	1	6
5	Microarrays	.	.	3	.	.	10	.	.	13
6	Sequence Analysis	2	20	.	22
7	Value Manipulation and Unit Convertor	7	7

Trial 1 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8737864077669902

----- Trial 2 -----

Trial 2 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
Trial 2 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0

Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.8737864077669902

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	13	2	15
1	Currency,Stock,Finance	2	10	12
2	Data Retrieval	.	.	21	.	1	2	.	.	24
3	Fax And Messaging	.	.	.	6	6
4	Graphics and Multimedia	2	1	1	.	3	1	.	.	8
5	Microarrays	7	.	.	7
6	Sequence Analysis	26	.	26
7	Value Manipulation and Unit Convertor	.	1	4	5

Trial 2 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=
 0.8737864077669902

----- **Trial 3** -----

Trial 3 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
 Trial 3 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
 1.0

Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.8543689320388349

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	7	1	1	.	1	1	.	.	11
1	Currency,Stock,Finance	.	8	1	.	1	.	.	.	10
2	Data Retrieval	.	.	27	.	.	3	.	.	30
3	Fax And Messaging	.	.	1	5	1	.	.	.	7
4	Graphics and Multimedia	.	.	2	.	4	1	.	.	7
5	Microarrays	.	.	1	.	.	7	.	.	8
6	Sequence Analysis	28	.	28
7	Value Manipulation and Unit Convertor	2	2

Trial 3 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=
 0.8543689320388349

----- **Trial 4** -----

Trial 4 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
 Trial 4 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
 1.0

Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.8446601941747572

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	7	4	.	1	1	1	.	.	14
1	Currency,Stock,Finance	2	11	.	.	1	.	.	.	14
2	Data Retrieval	.	.	29	29
3	Fax And Messaging	.	.	1	7	8
4	Graphics and Multimedia	.	.	2	.	1	1	.	.	4
5	Microarrays	.	.	1	.	.	4	.	.	5
6	Sequence Analysis	25	.	25
7	Value Manipulation and Unit Convertor	1	3	4

Trial 4 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=
 0.8446601941747572

----- Trial 5 -----

Trial 5 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
 Trial 5 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
 Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.912621359223301

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	12	2	14
1	Currency,Stock,Finance	1	10	.	.	1	.	.	.	12
2	Data Retrieval	.	1	25	.	.	1	.	.	27
3	Fax And Messaging	.	.	.	3	.	1	.	.	4
4	Graphics and Multimedia	5	.	.	1	6
5	Microarrays	.	.	1	.	.	5	.	.	6
6	Sequence Analysis	25	.	25
7	Value Manipulation and Unit Convertor	9	9

Trial 5 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.912621359223301

----- Trial 6 -----

Trial 6 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
 Trial 6 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
 Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.8932038834951457

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	8	1	.	.	1	.	.	.	10
1	Currency,Stock,Finance	1	7	1	.	1	.	.	.	10
2	Data Retrieval	.	.	29	.	.	3	.	.	32
3	Fax And Messaging	.	.	.	3	2	.	.	.	5
4	Graphics and Multimedia	.	1	.	.	5	.	.	.	6
5	Microarrays	9	.	.	9
6	Sequence Analysis	26	.	26
7	Value Manipulation and Unit Convertor	5	5

Trial 6 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8932038834951457

----- Trial 7 -----

Trial 7 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
 Trial 7 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
 Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=1.0
 Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
 Confusion Matrix, row=true, column=predicted accuracy=0.970873786407767

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	13	13
1	Currency,Stock,Finance	.	14	14
2	Data Retrieval	.	.	23	.	.	1	.	.	24
3	Fax And Messaging	1	.	.	4	5
4	Graphics and Multimedia	8	.	.	1	9
5	Microarrays	8	.	.	8

```

6           Sequence Analysis . . . . . 24 . |24
7 Value Manipulation and Unit Convertor . . . . . 6 |6

```

Trial 7 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.970873786407767

----- Trial 8 -----

```

Trial 8 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
Trial 8 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
1.0
Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8155339805825242

```

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	12	1	.	.	1	1	.	.	15
1	Currency,Stock,Finance	2	11	1	14
2	Data Retrieval	.	1	23	.	2	2	.	.	28
3	Fax And Messaging	.	.	.	4	4
4	Graphics and Multimedia	2	.	2	1	4	.	.	.	9
5	Microarrays	.	.	3	.	.	3	.	.	6
6	Sequence Analysis	19	.	19
7	Value Manipulation and Unit Convertor	8	8

Trial 8 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8155339805825242

----- Trial 9 -----

```

Trial 9 Training MaxEntTrainer,gaussianPriorVariance=1.0 with 239 instances
Trial 9 Training MaxEntTrainer,gaussianPriorVariance=1.0 finished
Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 training data accuracy=
1.0
Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 Test Data Confusion Matrix
Confusion Matrix, row=true, column=predicted accuracy=0.8446601941747572

```

	label	0	1	2	3	4	5	6	7	total
0	AddressAndLocation	11	.	.	.	1	.	.	.	12
1	Currency,Stock,Finance	2	7	.	.	1	.	.	.	10
2	Data Retrieval	2	.	22	1	.	2	.	.	27
3	Fax And Messaging	1	.	.	1	2
4	Graphics and Multimedia	2	.	.	.	5	1	.	1	9
5	Microarrays	.	.	2	.	.	7	.	.	9
6	Sequence Analysis	23	.	23
7	Value Manipulation and Unit Convertor	11	11

Trial 9 Trainer MaxEntTrainer,gaussianPriorVariance=1.0 test data accuracy=0.8446601941747572

```

MaxEntTrainer,gaussianPriorVariance=1.0
Summary. train accuracy mean = 1.0 stddev = 0.0 stderr = 0.0
Summary. test accuracy mean = 0.8776699029126214 stddev = 0.04123642831377731
stderr = 0.01304010360417928

```

BIBLIOGRAPHY

1. Erin Cavanaugh - Product Marketing Manager Altova. "Web services: Benefits, challenges, and a unique, visual development solution".
2. "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", June 26, 2007, <http://www.w3.org/TR/wsdl20>.
3. Khalid Elgazzar, Ahmed E. Hassan and Patrick Martin, "Clustering WSDL Documents to Bootstrap the Discovery of Web Services," icws, pp.147-154, 2010 IEEE International Conference on Web Services, 2010.
4. Binding point. <http://www.bindingpoint.com/>.
5. Grand central. <http://www.grandcentral.com/directory/>.
6. Salcentral. <http://www.salcentral.com/>.
7. Plebani, P., Pernici, B., "URBE: Web Service Retrieval Based on Similarity Evaluation," Knowledge and Data Engineering, IEEE Transactions on , vol.21, no.11, pp.1629,1642, Nov. 2009
8. Beniamino Di Martino, "Semantic web services discovery based on structural ontology matching", International Journal of Web and Grid Services, Volume 5, No.1, 2009.
9. X. Dong, A. Halevy, J. Madhavan and E. Nemes, J. Zhang, "Similarity Search for Web Services", VLDB Conference. Toronto,2004, Canada.
10. M. Bruno, G. Canfora, M. D. Penta, and R. Scognamiglio, "An approach to support web service classification and annotation," in Proc. of the IEEE Int. Conf. on e-Technology, e-Commerce and e-Service (EEE'05), 2005, pp. 138-143. <http://www.w3.org/Submission/SWSF-SWSL/>.

11. Peng Liu, Jingyu Zhang and Xueli Yu “Clustering-Based Semantic Web Service Matchmaking with Automated Knowledge Acquisition” Web Information Systems and Mining , Lecture Notes in Computer Science Volume 5854, 2009, pp 261-270.
12. Wei Liu, Wilson Wong, “Web service clustering using text mining techniques,” International Journal of Agent-Oriented Software Engineering, Vol. 3, No. 1, pp. 6-26, 2009.
13. Yu Wanjun; Song Xiaoguang, "Research on text categorization based on machine learning," Advanced Management Science (ICAMS), 2010 IEEE International Conference on , vol.2, no., pp.253,255, 9-11 July 2010.
14. Konduri, Aparna, “Clustering of Web Services Based on Semantic Similarity” University of Akron, Computer Science, 2008, <http://etd.ohiolink.edu/send-pdf.cgi/Konduri%20Aparna.pdf?akron1199657471>.
15. Jing Zhang Dan Pan, “Web Service classification”, Department of Computer Science UC, Irvine UC, Irvine, pand@ics.uci.edu.
16. Teresa Gonçalves and Paulo Quaresma, “Evaluating Preprocessing Techniques in Text Categorization”, International Journal of Computer Science and Application Issue 2010.
17. Anil Kumar and S.Chandrasekhar, “Text Data Pre-processing and Dimensionality Reduction Techniques for Document Clustering”, International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 5, July – 2012.
18. Porter, M.F. (1980), an algorithm for suffix stripping.
<http://tartarus.org/~martin/PorterStemmer/def.txt>
19. “Stemming and lemmatization”,
<http://nlp.stanford.edu/IRbook/html/htmledition/stemming-and-lemmatization-1.html>
20. “Function Word”, http://en.wikipedia.org/wiki/Function_word

21. “Membrane SOA Model – Java API for WSDL and XML Schema”
<http://www.membrane-soa.org/soa-model/>.
22. Mita K Dalal and Mukesh A Zaveri, “Automatic Text Classification: A Technical Review”, International Journal of Computer Applications 28(2):37-40, August 2011, published by Foundation of Computer Science, New York, USA.
23. Function Word List,
http://www.flesl.net/Vocabulary/Singleword_Lists/function_word_list.php
24. The Standard Natural Language Processing Group ,
<http://nlp.stanford.edu/software/corenlp.shtml>
25. Machine Learning for Language Toolkit, <http://mallet.cs.umass.edu/index.php>.
26. Hui Wang; Lin Wang, Lixia Yi, "Maximum Entropy framework used in text classification," Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference, vol.2, no., pp.828,833, 29-31 Oct. 2010.
27. <http://ws.apache.org/woden/>. Last published 04/05/2013.
28. Badam-Osor Khaltar, Atsushi Fujii, “A lemmatization method for Mongolian and its application to indexing for information retrieval”, Information Processing & Management Volume 45, Issue 4, July 2009, Pages 438–451.
29. He Youquan; Xie Jianfang; Xu Cheng, "An improved Naive Bayesian algorithm for Web page text classification," Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on , vol.3, no., pp.1765,1768, 26-28 July 2011.
30. Ping Bai; Junqing Li, "The Improved Naive Bayesian WEB Text Classification Algorithm," Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on, vol., no., pp.1, 4, 18-20 Jan. 2009.
31. Zhijie Liu; Xueqiang Lv; Kun Liu; Shuicai Shi, "Study on SVM Compared with the other Text Classification Methods," Education Technology and Computer Science

- (ETCS), 2010 Second International Workshop on , vol.1, no., pp.219,222, 6-7 March 2010.
32. Suman Saha , C. A. Murthy , Sankar K. Pal, Classification of web services using tensor space model and rough ensemble classifier, Proceedings of the 17th international conference on Foundations of intelligent systems, May 20-23, 2008, Toronto, Canada.
 33. Ioannis Katakis, Georgios Meditskos, Grigorios Tsoumakas, Nick Bassiliades, and Ioannis P. Vlahavas. On the combination of textual and semantic descriptions for automated semantic web service classification. In AIAI, volume 296 of IFIP, pages 95–104. Springer, 2009.
 34. Fangfang Liu, Yuliang Shi, Jie Yu, Tianhong Wang, Jingzhe Wu, "Measuring Similarity of Web Services Based on WSDL," icws, pp.155-162, 2010 IEEE International Conference on Web Services, 2010.
 35. Jiangang Ma , Yanchun Zhang , Jing He, Efficiently finding web services using a clustering semantic approach, Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008), p.1-8, April 22-22, 2008, Beijing, China .