A Framework for Efficient Web Services Discovery

A Thesis

Submitted in the fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Kailash Chander
(Registration No. 950903040)

Under the supervision of

Dr. R.K. Sharma

Professor, Department of Computer Science and Engineering,

Thapar University, Patiala-147004, INDIA

Thapar University, Patiala

Department of Computer Science and Engineering,

Thapar University, Patiala -147004, INDIA

Dec, 2016

CERTIFICATE

I hereby certify that the work which is being presented in this thesis entitled "**A Framework for Efficient Web Services Discovery**", in fulfillment of the requirements for the award of degree of Doctor of Philosophy submitted in Department of Computer Science and Engineering of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. R.K. Sharma and refers other research works which are duly listed in the reference section.

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

(Kailash Chander) Dated: 2/7/17

Regn. No. 950903040

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

(R.K. Sharma) 2.7.17

Professor

Department of Computer Science and Engineering

Thapar University

Patiala, 147004

Punjab, INDIA.

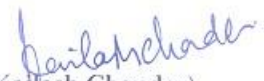
ACKNOWLEDGEMENT

I would like to express my sincere appreciation to my supervisor, **Prof. R.K. Sharma**, for being the pillar of support and encouragement throughout my research work. His experience, strength, tenderness and willfulness, has taught me the lessons of life, which are of immense help to me to take decisions in my every endeavor. I am thankful to him that he has full confidence in my ability and this work would have not been completed without his vision, knowledge and encouragement.

I also acknowledge the co-operation and encouragement extended to me by my brother **Dr. Amit Kumar Bhardwaj** who provided me support in different stages of my work. I am thankful to all the faculty and staff members of CSED for their support.

I am thankful to my wife **Seema** and my loving son **Suryansh** and daughter **Samridhi**, without their unbiased sacrifice and support, this work would have not been completed.

Although I have tried to express my gratitude to every person who contributed in this work directly or indirectly, there may still be someone hiding behind the veils of my forgetful part of memory. Last but not the least, I would like to thank all such souls.



(Kailash Chander)

Abstract

As number of web services is increasing day by day, it becomes essential to select the correct kind of web service for a user requirement. This is really a challenge to select a web service by understanding its capabilities provided in its description. The selection process of a web service depends upon the mechanism used for discovery and selection of the web service, which is a key component in service-oriented architecture. Some of the existing methodologies for web services discovery cater to web services with semantic annotations *i.e.* web services that have related semantic descriptions and some are merely based on syntactic based approaches. Selecting an appropriate service out of the discovered services poses a challenge to users. Ranking plays an important role in selecting a desired web service and there exists a number of ways for ranking the web services.

In this thesis, we present a framework for finding the most suitable web service according to user's requirements and improve the discovery process. A framework called Automatic Classification and Ranking of Web Services (*ACRWebS*) has been proposed and designed to deal with the problem of selecting the best web service based on the users need. Our approach uses *WordNet* based semantic similarity of web services and the user query. Path-length based similarity algorithm has been used in order to find a similarity score between user query and web service description data. It has been observed that the similarity score proposed by the framework meets the expectations of human interpretation. Performance of the *ACRWebS* has been evaluated and tested using statistical measures.

In the proposed framework, classification techniques have been combined with ranking techniques to find the most relevant service for a user (consumer). A dataset of the active web services has been created for conducting experiments. We have also taken another existing test dataset of a researcher as a benchmark for the comparison of experimental results.

List of Abbreviations

ACRWebS	Automatic Classification and Ranking of Web Services
AI	Artificial Intelligence
ANN	Artificial Neural Network
ANoVA	Analysis of Variance
API	Application Program Interface
AR	Association Rule
AWS	Amazon Web Services
B2B	Business to Business
BoW	Bag of Words
CoS	Cost of Service
DT	Decision Tree
DR	Decision Rule
GUI	Graphical User Interface
HA	Hierarchical Agglomerative
IR	Information Retrieval
k-NN	*k*-Nearest Neighbour
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
n	Number of Services
NB	Naïve Bayes
NLP	Natural Language Processing
OWL	Ontology Web Language
PLSA	Probabilistic Latent Semantic Indexing
QoS	Quality of Service
RF	Random Forest
RMI	Remote Method Invocation
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol

SVM	Support Vector Machine
TD	Term Document
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
UBR	UDDI Business Registry
UDDI	Universal Discovery Description and Interface
UI	User Interface
VSM	Vector Space Model
WSDL	Web Services Description Language
WSMO	Web Services Modeling Ontology
WSRF	Web Services Resource Framework
XML	Extensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Internet is playing a very important role in day-to-day business. Internet users not only want to leverage internet for information searching and publishing, they also want to connect with their business partners and customers. The emergence of distributed computing led to a new era of network computing which resulted into emergence of World Wide Web and Internet. But, as a known fact new technologies have their own new challenges, distributed computing also has a challenge in the form of heterogeneity problem. There are always differences in operating systems, architectures, application interfaces, security requirements and data standards. All the involved entities in the system should be able to interoperate despite having all these differences. To overcome the heterogeneity problem, many languages and technologies like Java RMI, DCOM and CORBA have been developed. These technologies hide heterogeneity problem with the help of standard and common interfaces for communication. Also, these technologies have their own issues and do not fully support the heterogeneous environment. Some of them are specific programming language dependent and some are operating system dependent. For example, DCOM components communicate only if they are deployed on windows OS and programmed in C#, C++ or C. Similarly, Remote Method Invocation (RMI) technology based applications communicate only if they are programmed in Java. Another problem with the distributed computing is that its communication protocol requires specific firewall ports to be opened for the communication and this poses the security risks.

Owing to these aforementioned problems, distributed technologies are federated and tightly coupled. This becomes an important issue for enterprise applications. If there is a change in business partner or technology supplier, then this will impact the application and whole application may be required to be reengineered. This reduces the integration flexibility for enterprise applications and business to business (B2B) interaction. This leads to an

impact on the ability of an organization to meet competition from the changing markets. Enterprise applications having the same communication technology can be easily integrated. Otherwise, additional development efforts is required to achieve the successful enterprise integration. To counter this issue of enterprise application integration, web services technology has been developed.

1.1 Web services

Web services is a standardized way of integrating web based applications with the help of Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Discovery Description and Interface (UDDI) open standards over an internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services and UDDI is used for listing which services are available. Web services allow different applications from different sources to communicate with each other without custom coding. This saves a lot of time and effort and hence reduces the cost of enterprise integration. Since all the communication is in XML format, web services do not depend on any operating system or programming language. Hence it is both language and platform independent.

Recently, some commercial web services have been in tremendous use. For example, in the case of Amazon Web Services (AWS), a consumer company can get access to AWS and can consume its provided functionality by paying money according to the usage of service. It can be seen that web services are gradually playing an important role in various applications, mainly the commercial applications. Developers can consume the web services and can also use the functionality provided by these services via their own Graphical User Interface (GUI). Another major advantage of web services is to reuse application components. We do not need to make these components repetitively. Web services solve the problem of interoperability by giving different applications a way to link the data.

Several services for business areas such as weather reports, currency conversion or even for language translation are available. In essence, web services provides freedom to

communicate between different platforms and applications. Web services offer a well-defined interface through which other programs may interact by sending messages based on internet protocols and web standards. The description of a web service interface is based on the WSDL file that describes the syntax of the input and output messages using XML standard. This file also provides other details needed for the invocation of the service. The communication is based on the SOAP, an XML-based framework that provides a message construct that can be exchanged over a variety of underlying protocols.

The basic process for implementing a web service is:

- A service provider composes a web service and uses WSDL file to describe it.
- The web service provider registers the service in a UDDI registry / Repository.
- A web service consumer or another service searches and requests the registered service by querying UDDI or Repositories.
- The requesting consumer or service binds the registered service using SOAP
- Communication takes place between user and service with messages and data exchanged in XML format.

1.1.1 Web services components

In the traditional client-server world, we have the servers offering some functionality that could be used or called by the clients. Some kind of lookup service acts as a broker between a client and a server. Web services consist of three components as described below.

- Web services Broker that acts as a lookup service between services provider and services requester.
- Web services Provider that publishes services to the services broker.
- Web services Requester that asks the services broker where to find suitable services provider.

Figure 1.1 illustrates the relationship between various components of web services.

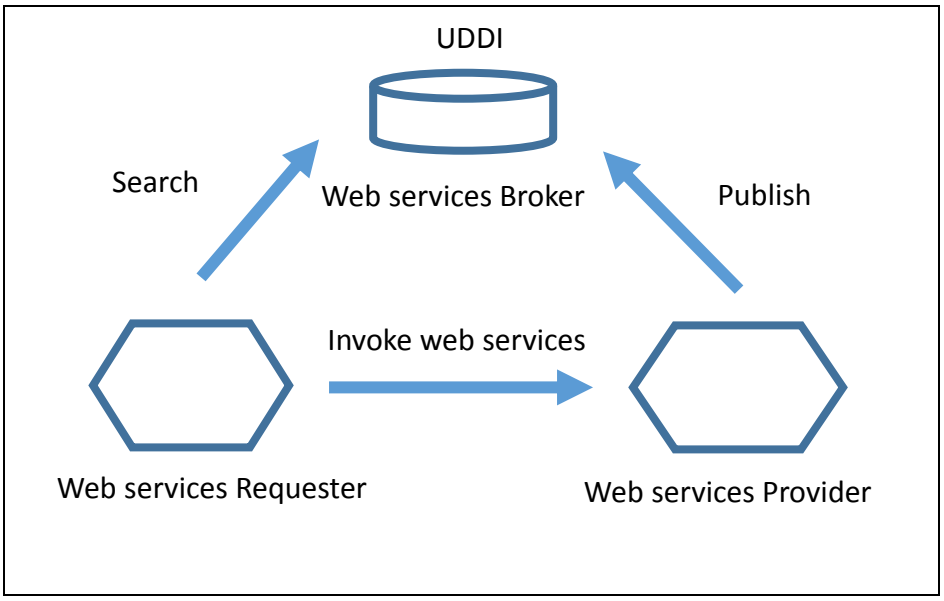


Figure 1.1: Web services components

The process of web services usage is described in the following steps.

A web services provider describes the web services in a WSDL document and publishes it on UDDI registry using publisher's Application Program Interface (API), which is based on SOAP. The UDDI project handles a global public registry called the UDDI Business Registry (UBR). This registry is available to all without any charges (http://www.uddi.org), and the general public have access to all information registered in the public registry. Organizations can have their private registry to support their requirements.

Since web services use XML on top of HTTP, so there cannot be any problem with firewalls. Web services protocol stack consists of XML, SOAP, WSDL and UDDI. Figure 1.2 shows various web services established standards on XML.

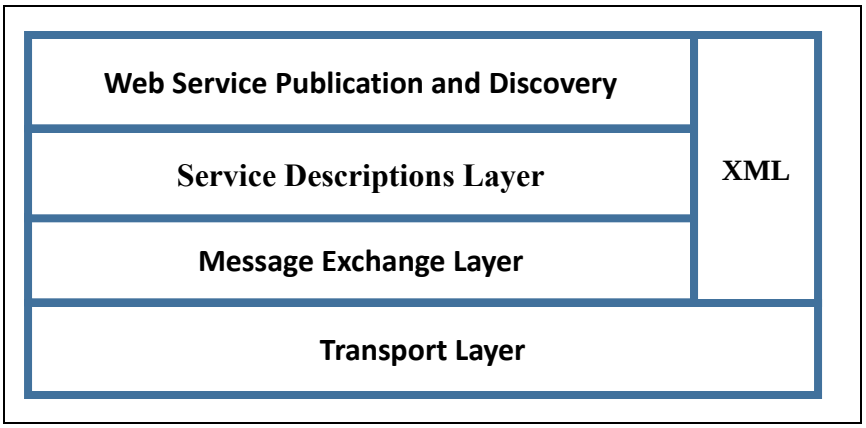


Figure 1.2: Web services standards

Transport layer

The transport layer moves messages across applications. The most commonly used transport protocol is the standard HTTP protocol, apart from other web protocols such as FTP and SMTP.

Message exchange layer

Message layer uses SOAP protocol that allows programs that run on disparate operating systems such as windows and linux to communicate using HTTP and XML. SOAP is based on XML, which is the foundation of all web services and is widely used in web services applications.

WSDL layer

This layer represents a way of specifying a public interface for a web service. It contains information on available functions, on data types for XML messaging, binding information about the transport protocols, and the location of the specific web services. WSDL provides a common format for describing and publishing web services information. A client application that requires information about a service such as the data it expects to receive, the results being delivered and the supported transport protocol, uses WSDL file to find this

information. When one creates a web service, it must be described and advertised to its potential customers. Typically, WSDL is used with SOAP and the WSDL specification includes a SOAP binding.

UDDI layer

This layer represents a way to publish and find web services from the services registry. UDDI is a general purpose registry and it can be used to register any type of web services. UDDI is used for listing which services are available and is a specification for distributed registries of web services. A UDDI web services registry itself is a web service which can be accessed via SOAP from any application that wishes to discover web services. UDDI specifies interfaces for applications to publish and discover web services (as WSDL documents). An UDDI entry contains more than just a WSDL interface and implementation. It also includes metadata such as quality of services parameters, payment mechanisms, security and keywords for resource discovery.

1.1.2 Advantages of web services

Web services offer several business and technological benefits as given below.

- **Interoperability:** This is the most important feature of web services. Web services are independent of a programming language and provide developers flexibility for application development. In addition, use of standard communication techniques makes web services platform independent.

- **Usability:** Web services allow the business logic of many different systems to be exposed over the web. This gives applications the freedom to choose the web services that they need. Instead of re-inventing the wheel for each client, one needs to include additional application-specific business logic on the client-side only. This provides a flexibility to develop services and/or client-side code using the languages and tools that one wants to work upon.

- **Reusability:** Code reuse has long been one of the most eagerly pursued issues of software development and this is one of the key features of web services. Writing code once and using it many times saves money and efforts. This also reduces the risk involved in software development.

- **Deployability:** Web services are easy to deploy as these uses the standard internet technologies.

Web services have some disadvantages in the form of overheads and lack of versatility, as mentioned below.

Overhead: Transmitting all the data in XML is not as efficient as using a proprietary binary code. Although, this overhead is usually acceptable for most applications, this is difficult to find a critical real-time application that uses web services.

Lack of versatility: Currently, web services are not very versatile, since these allow some very basic forms for service invocation. Most of the programming languages, however offer a lot of supporting services such as persistency, notifications, lifecycle management, transactions, *etc*. that can be used to increase the versatility of web services. Fortunately, there are a lot of emerging web services specifications such as web services resource framework (WSRF) that are helping to make web services more flexible.

1.1.3 Web services description language

WSDL describes web services starting with the messages that are exchanged between a services provider and a requester. The messages themselves are described abstractly and then bound to a concrete network protocol. Web services definitions can be mapped to any language, object model, or messaging system. Both the sender and receiver of a web services message must have access to the same services description. The sender needs the services description to know how to format the message correctly and the receiver needs the services description to understand how to receive the message correctly. As long as both the sender

and receiver have the same services description (*e.g.* WSDL file), the detailed implementations behind the web services need not be known to the user.

WSDL is an XML based standard for describing web services. This standard provides the information to services requester about the location of web services and also about the process of invoking the web services. As shown in Figure 1.3, a typical WSDL document has root element as "<definitions>". The sub-elements under the "<definitions>" element are: <types>, <message>, <portType>, <binding>, and <service>. Figure 1.4 contains an example WSDL file.

The "<types>" element contains all the data type definitions of the described service for sending and receiving messages. The XML schema namespace (http://www.w3.org/2001/XMLSchema) is used for the data type definitions.

```
<definitions>

<types>
      data type definitions........
</types>

<message>
      definition of the data being communicated....
</message>

<portType>
      set of operations......
</portType>

<binding>
      protocol and data format specification....
</binding>

</definitions>
```

Figure 1.3: WSDL file structure

```
<?xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote.wsdl"
xmlns:tns="http://example.com/stockquote.wsdl"
xmlns:xsd1="http://example.com/stockquote.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
<schema targetNamespace="http://example.com/stockquote.xsd"
xmlns="http://www.w3.org/2000/10/XMLSchema">
<element name="TradePriceRequest">
<complexType>
<all>
<element name="tickerSymbol" type="string"/>
</all>
</complexType>
</element>
<element name="TradePrice">
<complexType>
<all>
<element name="price" type="float"/>
</all>
</complexType>
</element>
</schema>
</types>
<message name="GetLastTradePriceInput">
<part name="body" element="xsd1:TradePriceRequest"/>
</message>
```

Continued on next page

```
<message name="GetLastTradePriceOutput">
<part name="body" element="xsd1:
TradePrice"/>
</message>
<portType name="StockQuotePortType">
<operation name="GetLastTradePrice">
<input message="tns:GetLastTradePriceInput"/>
<output message="tns:GetLastTradePriceOutput"/>
</operation>
</portType>
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="GetLastTradePrice">
<soap:operation soapAction="http://example.com/GetLastTradePrice"/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
<service name="StockQuoteService">
<documentation>My first service</documentation>
<port name="StockQuotePort" binding="tns:StockQuoteBinding">
<soap:address location="http://example.com/stockquote"/>
</port>
</service>
</definitions>
```

Figure 1.4: WSDL file: an example

These elements of a WSDL file are explained as below.

Definitions: This part includes the data type definitions and message definitions. Message definitions make use of the data type definitions.

Message: The "<message>" element explains how the data types defined in the "<types>" element are bound with the request and response messages. The "<message>" element can appear a number of times depending on how many functions the web service provides.

Operations: These describe the actions for the messages supported by the web services. The information within the "<portType>" element is a set of operations. "<Operation>" sub-elements are used to identify the functions provided by the service and their operation types.

There are four kinds of operations defined in the WSDL specifications.

i) **One-way operation**: The endpoint receives a message.

ii) **Request-response operation**: The endpoint receives a message and sends a reply.

iii) **Solicit-response operation:** The endpoint sends a message and receives a reply.

iv) Notification operation: The endpoint sends a messag**e.**

These four operations are represented by three sub-elements within the "<operation>" element, *i.e.,* the "<input>" element, the "<output>" element, and the "<fault>" element.

- A one-way operation only specifies the "<input>" element.
- A request-response operation specifies the "<input>" element first, followed by the "<output>" element and some optional "<fault>" elements.
- A solicit-response operation specifies the "<output>" element first, and then the "<input>" element followed by some optional "<fault>" elements.
- Notification operation specifies one "<output>" element.

Port Types: Port types are a set of operations provided by a web service.

Service Bindings: This element depicts the bindings of service portType to a port. A port is illustrated by a combination of network address and a portType.

The relationship among various elements of a WSDL file is shown in Figure 1.5.

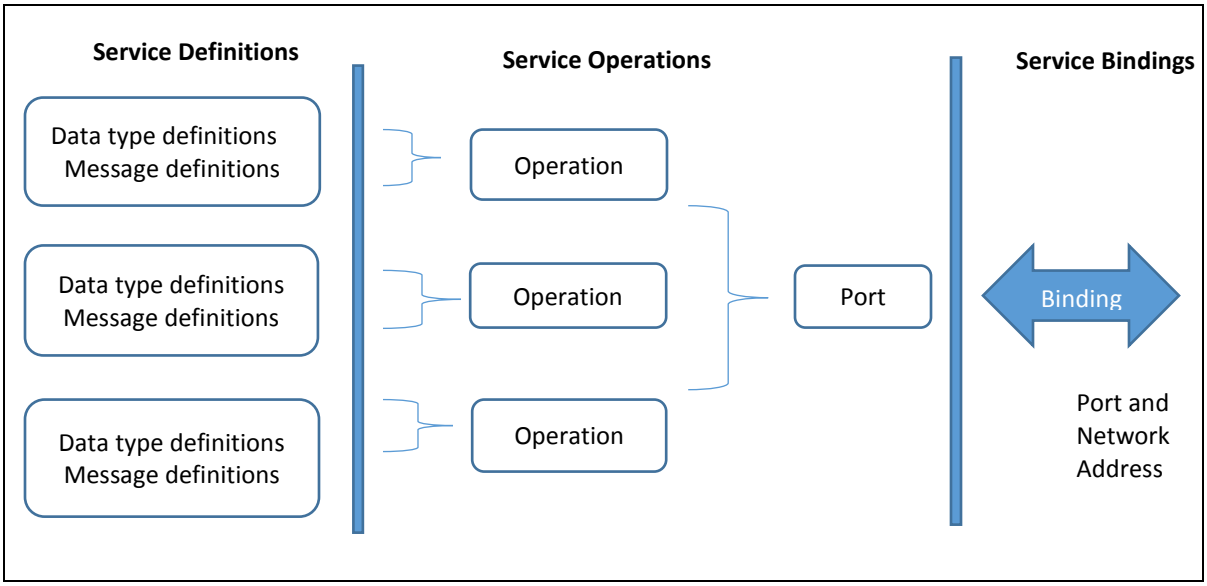


Figure 1.5: Relationship between elements of a WSDL file

1.1.4 Web services discovery: current trends

Currently, web services are available in repositories such as UDDIs and web portals such as Xmethods (http://www.xmethods.net), webservicex (http://www.remotemethods.com) and webservicelist (http://www.webservicelist.com). There are three main approaches for the discovery of potential web services:

i) Using centralized repositories such as UDDIs

ii) Using web crawling techniques

iii) Searching specific web portals

Though there are not many public UDDIs available, yet organizations are using the option of the discovery of web services through private UDDI Business Registries (UBRs) and

search engines which are maintained by the individual organizations. There exist syntactic based standards such as WSDL, SOAP, and UDDI to support the discovery of web services. Seekda (http://seekda.com) is currently the most comprehensive global search engine for searching public web services.

1.1.5 Various roles in web services architecture

Web services life cycle encompasses a well-defined hierarchy of roles, as explained below.
Service provider: From an architectural perspective, this is the platform that hosts web services. From the business perspective, this is the owner of the web services.

Service requester: This is the user of the web services interested in using the functionality offered by services.

Service registry: This is a logically centralized directory of services. The registry provides a central place where developers can publish their new services or find existing ones.

1.2 Problem areas

With the adoption of Service Oriented Architecture (SOA) and the increase of internet based business, there has been a drastic increase in web services usage. This becomes difficult for the users to search web services that are exactly required by them. Let us illustrate how web services discovery is an area of concern. We will begin with different types of use cases for the discovery of services.

Figure 1.6 shows a simple use case which provides a base for understanding the requirements of a user. This use case shows that all users who want to find a web service search a centralized repository. The result of use case is a service that satisfies user's need. Following cases are pertinent in this situation.

Use Case 1: Browse web services descriptions in UDDI

In this use case, user browses through the UDDI repository manually and looks into all categories. As the repository does not have user-friendly format of standard, a user generally have a problem in selecting the right kind of web services.

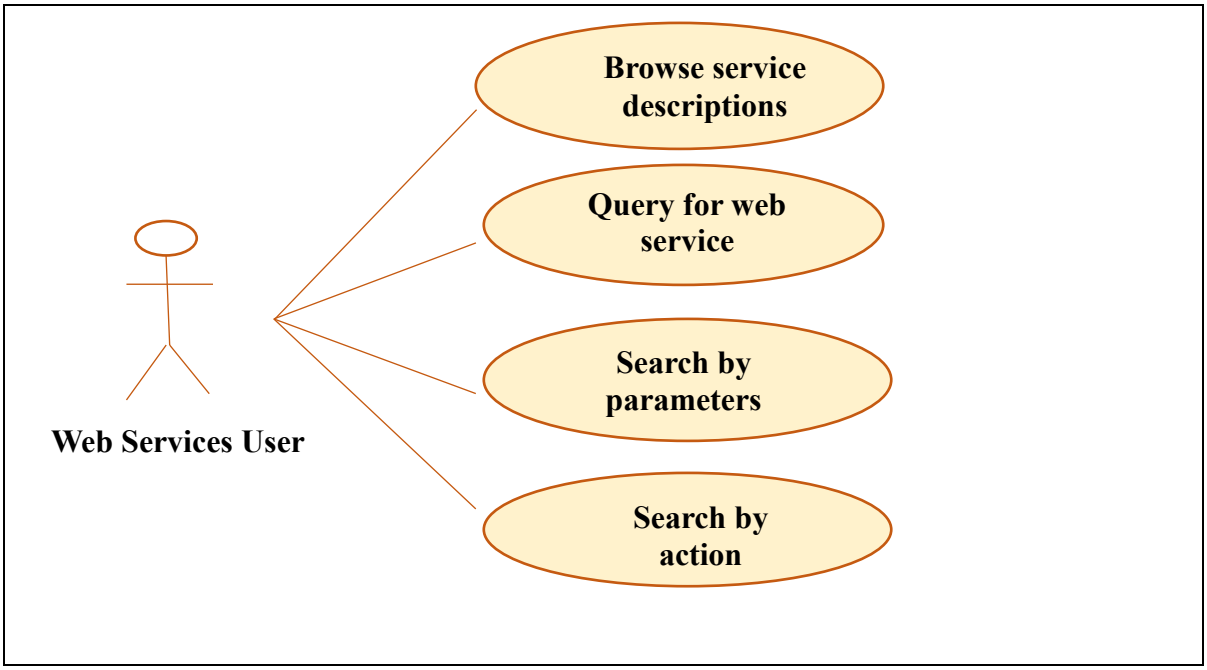


Figure 1.6: Use case of web services discovery

Use Case 2: Query for searching service

In this case, a user queries to the services repository with the help of a query language. This language describes user requirements for search and receives a response from the system. Query language should reflect search criteria.

Use Case 3: Parameter based web services search

In this case, a user queries to the services based on web services parameters such as input, output values of the functions. These parameters can be domain objects.

Use Case 4: Action based search

This type of search will depend on the actions provided by the web services.

It has been observed that current discovery mechanisms are limited in their search capabilities as they are mostly based on the keywords based matching. The web services consumer searches web services in registry and submits requirements using keywords. The keyword based techniques used in common search engines are not efficient as many irrelevant services may be included in the description of the queried keywords. The key issue is that keywords are a poor way to capture the semantics of services request or services advertisement. Thus a different mechanism is needed, one that entails locating web services based on their offered capabilities.

The keyword based method of services discovery has a low recall. This is due to the reason that the result of the services with same concepts as that defined by user query are not selected and returned to the user. For example, a service named *car* may not be returned from the query *automobile* submitted by a user, though these are same at the conceptual level. The user requirement may depend on various resources offered by the services such as interfaces, functionality, security and commercial point. Despite the success of the UDDI specifications and its rapid uptake by the industry, the capabilities of services discovery are rather limited. The lack of machine understandable semantics in the technical specifications and classification schemes used for retrieving services, prevent service registries from supporting complete automation for effective services discovery.

Some other challenges for web services discovery are:

- Quantum of services available on the internet is high.
- WSDL format has not been standardized.
- There is limited support for searching functionally similar web services.
- Development of a common ontology for myriad of web services is difficult and cumbersome task.
- Manual annotation of web services is a challenging task.

In the current scenario, search engine and other semantic based search engines require technical details of the required services for the discovery purpose. Services users need sufficient domain knowledge about the required services to provide the functional input to

the discovery engine. Several existing tools and mechanisms allow this discovery, however, those approaches often skip different aspects such as services quality, reuse, evolution and user's comments making the discovery result feebly relevant to requester's need and prevent the requesters from using up-to-date and available web services efficiently.

Often, the service consumers do not have detailed domain knowledge and it is difficult to provide functional and technical details for the services discovery. So, most of the time these search engines do not provide desired results as the information provided by users is not up to the mark and sufficient.

The procedures of composition, discovery and invocation of web services are centered on human involvement. The lack of semantic information for web services capability is a hindrance for automated discovery of web services. Also, manually searching for web services has become time-consuming and more painful for the consumer as the number of services has increased many folds with time.

It is desirable to provide a web services framework to be able to provide recommendations of appropriate web services, which are of interest to users. However, the task of recommending web services is very challenging and is one of the major problem areas of web services.

1.3 Current solutions

In the previous section, we presented a number of challenges associated with web services discovery. Researchers have developed solutions based on various techniques such as artificial intelligence and ontology web language to address these challenges and problems. There also exist methods based on the semantic similarity, which have been tried in web services discovery problem.

Semantic web initiative tries to solve problems related to knowledge representation by suggesting standards, tools and languages for information annotation. Thus, the combination of semantic web techniques with web services seems the best way to give appropriate

semantic notion to web services in order to achieve the desirable level of automation for web services discovery. A few languages and frameworks have been proposed in this direction.

1.3.1 Quality of services

It has been observed that most of the current efforts address functional aspects of web services. However, the non-functional properties also known as Quality of Service (QoS), are equally important in a business environment and play a deciding role in selecting a specific resource. These QoS could be performance, throughput, reliability, availability, and trust.

1.3.2 Semantic based approaches

Semantic web services discovery aims to discover the best matched web service. Some of the current approaches use semantics to support this initiative. Semantic similarity measures play an important role in the extraction of semantic relations. Semantic similarity measures are widely used in Natural Language Processing (NLP) and Information Retrieval (IR) systems. It mostly depends on the measurement of the similarity degree between services request and services advertisement.

Many research areas such as linguistics, artificial intelligence, cognitive science and knowledge engineering have been taking advantage of the semantic similarity between keywords. Calculation of semantic similarity between web services plays an important role for web services discovery. Particularly, semantic services discovery aims to locate the best selected services after match-making. This depends on the measurement of similarity between a consumer's requirements and the profile description of published services. This is usually achieved by using ontologies for web services semantics. In nutshell, measurement of semantic similarity between web services can be reduced to computing semantic distances between ontologies. The semantic ontology languages for services, such as the Ontology Web Language for services (OWL-S) and Web Services Modeling Ontology (WSMO), are

required to represent services capabilities semantically. The various information sources for the semantic ontology are:

- Corpus of documents.
- Lexical resources such as semantic networks, dictionaries and thesauri.
- Internet.

Semantic networks are considered better choices for estimating semantic similarity than other lexical resources. *WordNet* is one of the largely used semantic networks for approximating the word semantic similarities.

Table 1.1 shows comparative analysis of syntax and semantic based techniques. These two are the most prominent techniques in web services discovery. It has been seen that semantic based approaches are more suitable for the web services discovery, though these have some associated challenges.

Table 1.1: Comparison of syntactic and semantic web services discovery methods

Approach	Merits	De-merits
Syntax based approaches	• Simple and mostly used technique • Keyword-based search is more familiar to the user • Standards already established *e.g.*, UDDI	• Mostly requires human Interaction • Unable to select similar service • Low precision of selection • Not suited for automatic processing
Semantic based approaches	• Reduce the manual discovery process and allow dynamic discovery of web services. • Reliable and effective Methodology • Effectiveness and relevancy of discovered web services can be enhanced by extending semantics through standards such as OWL, WSDL-S ans WSMO.	• End users should be familiar with the semantics, descriptions and implementation detail of web services. This makes the usage cumbersome. • The discovery space is often limited to some web services that follow a particular description standard. • Requires ontology mapping to support interoperability between ontologies. This is more a complex methodology.

1.3.3 Schema matching based approaches

There have been a number of solutions based on the schema or interface based matching of web services. These approaches are based on XML based descriptions of both web services request and WSDL descriptions. Generally, it takes two schemas as input and produces a mapping between the elements that correspond to each other semantically.

1.4 Classification methods in web services discovery

Classification is an extensively used mechanism for enabling web services discovery. Researchers are using various machine learning techniques on web services data to make discovery more efficient and effective. This data has been extracted from WSDL files and semantically annotated web services. Recently, machine learning models have been playing a significant role in the classification of web services discovery. A web services feature can be transformed into a format which can further be used in machine learning algorithms. Web services can be classified based on the WSDL file information such as the service name, input/output parameters, and operations. Classification of web services reduces the complexity of web services discovery to a great extent, as it reduces the search space and restricts the search to a particular domain.

There are many proposed approaches using machine learning methods for the web services classification area. But, still automated web services classification is a promising area of research and there is ample scope for improvement of efficiency using these approaches (Crasso *et al.*, 2008).

For each of the web services, its WSDL file is processed to extract the information into a collection of tokens. These tokens are further processed into a feature vector using the NLP techniques. In the next step these feature vectors are mapped to a category using classification algorithms.

The automatic classification of web services into predefined categories has been extensively used by researchers. The most popular classification models include Naïve Bayes (NB), Decision Tree (DT), Decision Rules (DRs), Association Rules (ARs), Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and Random Forest(RF). It has been found that most of the classification approaches used by researchers are based on supervised algorithm. A supervised algorithm uses the training data, where each web service is labeled with a category. These algorithms analyze the training data and produce a model, which can be used for classification of new services data. Machine learning approaches such

as NB, *k*-Nearest Neighbor (*k*-NN), SVM, ANN, Latent Semantic Analysis (LSA) and Genetic Algorithms (GA) *etc.* are few of the supervised based learning techniques.

Unsupervised learning techniques are used to classify web services based on their functional and non-functional similarities to enhance the web services discovery process. In unsupervised methods, three categories, namely, non-hierarchical clustering, hierarchical clustering and agglomerative have been widely used for the discovery of web services. An algorithm is trained to map a feature vector to a class label. Following sections provide an overview of various machine learning algorithms.

1.4.1 Naïve Bayes

Naïve Bayes classifiers are proven efficient algorithms for data mining applications. This classification method is named after Thomas Bayes (1702-1761), who proposed the Bayes theorem. Naive Bayes classifiers are among the most successful algorithms for classification of text and data. A Naïve Bayes classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. The main advantage of Naïve Bayes is that it requires only a small amount of training necessary for classification. Because independent variables (means and variances of the variables) are assumed, only the variances of the variable for each class need to be determined and not the entire covariance matrix.

This algorithm is based on Bayes rule of conditional probability. As per this rule, for a given hypothesis *H* and evidence *E* that bears on the hypothesis, the conditional probability of *H* given *E* is given by:

$$P(H|E) = \frac{(E|H)P(H)}{P(E)} \qquad (1.1)$$

Where $P(H)$ denotes the priori probability, the probability of the hypothesis before presenting an evidence. $P(E|H)$ denotes the conditional probability that H is true given evidence E. $P(E)$ denotes the probability of the evidence associated with the hypothesis.

1.4.2 Decision tree

A decision tree is one of the important knowledge structures that can result from data mining activities and is used for the classification problems. This algorithm generates a decision tree for classification of both nominal and numerical data. It is used to categorize the type of examples which may come in negative or positive forms. It is extensively used in data mining and machine learning.

The most widely-used version of the decision tree algorithm is Ross Quinlan's C4.5 (Quinlan, 1986) an extended, public-domain version of his earlier ID3 method. C4.5 uses the fact that each attribute of the data can be used to make a decision which splits the data into smaller subsets.

1.4.3 *k*-NN

k-NN is a very simple and efficient example based classification algorithm. This algorithm searches the *k* nearest neighbors among the pre-classified training documents and ranks those *k* neighbors based on their similarity scores. The categories of the *k* nearest neighbors are used to predict the category of the test document.

1.4.4 Artificial neural network

An ANN is an information processing paradigm that is inspired by the working of biological nervous system, such as how brain processes the information. ANNs, such as people, learn by examples. This can be configured for explicit applications, such as data classification and pattern recognition using a learning process.

1.4.5 Support vector machine

SVMs are powerful learning method presented by Vapnik *et al.* (1995). They are well suited for text classification. Since their appearance in early nineties, SVMs have been widely used for machine learning and data mining tasks and are quite popular for classification and regression due to their promising empirical performance. Apart from text and data mining, SVMs have been extensively used in the areas such as bioinformatics, fraud detection, construction of insurance tariffs and direct marketing *etc.* SVMs are based on the structural risk minimization principle from computational learning theory. They can be used to learn polynomial classifiers, radial basic function (RBF) networks, and three-layer sigmoid neural nets. One remarkable property of SVMs is that their ability to learn is independent of the dimensionality of the feature space. They allow not only the best classification performance (*e.g.*, accuracy) of the training data, but also leave much room for correct classification of the future data.

1.4.6 Random forest

Random forest is a notion of the general technique of random decision forests (Ho, 1995). This is an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and providing the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct decision trees problem of over-fitting to their training set.

1.5 *WordNet* and semantic similarity based on *WordNet*

WordNet is a large lexical database of English words. *WordNet* is a research project of Princeton University (Miller, 1990). In *WordNet*, nouns, verbs, adverbs and adjectives are organized by a variety of semantic relations into Synonym Sets (SynSets), which represents one concept.

WordNet has been used to measure semantic similarity among words since it has the inherent advantage of being structured in the similar way of simulating human recognition behaviors. There exist various methods for calculating the semantic similarity using *WordNet*. But they have their associated challenges in terms of cost and processing requirements. Lesk method (Banerjee and Pedersen, 2002) proposed that relatedness of two words is proportional to the extent of overlaps of their dictionary definitions. But, its computational cost is relatively high. We have chosen the path (node count) based approach to measure the similarity between SynSets. This approach is simple and involves less complexity and computational cost as compared to other approaches.

WordNet has been designed to establish the connection between four types of Parts of Speech (POS) - noun, verb, adjective and adverb. The smallest unit in a *WordNet* is SynSets, which represents a specific meaning of a word. It includes the word, its explanation and its synonyms. The specific meaning of one word under one type of POS is called a sense. Each sense of a word is in a different SynSet. It has been observed that most of the research uses nouns for semantic similarity. The definition of SynSets varies from the very specific one to the very general. Each SynSet has a gloss that defines the concept it represents. SynSets are connected to one another through explicit semantic relations.

Four commonly used semantic similarities are hyponym/hypernym (is-a), part meronym/part holonym (part-of), member meronym/member holonym (member-of) and substance meronym/substance holonym (substance-of). Examples of these relationships are: Mango is a fruit (is-a) and Mouse is part of computer (part-of). It is also observed that Hyponym (is-a) is the most common relationship and this accounts for about 80% of the noun relations (Meng *et al.,* 2013). Another example is *a board* and *a plank* grouped in the SynSets {board, plank}. But *a board* can also indicate a group of people and to disambiguate these homonymic significances *a board* will also belong to the SynSets {board, committee}.

1.5.1 Advantages of *WordNet* based approach

This is the most intuitive way of measuring the similarity between two SynSets. It calculates the semantic similarity between a pair of SynSets depending on the number of links existing between them in the *WordNet* taxonomy. The other advantages provided by *WordNet* based approaches over other approaches are:

i) This is the simplest approach and involves less complexity. It is easier to handle and more efficient to calculate the similarity degree compared to the other methods.

ii) It allows developers to enhance web services with semantic information without semantic annotation against ontology.

iii) *WordNet* is not domain specific and eliminates the semantic annotation cost of services.

1.6 Performance analysis

In order to evaluate performance of the framework, following statistical measures has been used.

Precision

The performance of proposed model is measured using precision and recall of the services returned by the model. Precision is the measurement to select the most precise web services. Higher the value of precision better is the similarity of the web services selected.

The Precision is defined as

$$\frac{A}{A+B} \times 100 \qquad (1.2)$$

Where, A is a number of relevant services selected and B is a number of irrelevant services selected. In other words, Precision is a measure of the accuracy provided that a specific class has been predicted.

It is also defined by:

$$tp/(tp + fp) \quad (1.3)$$

where *tp* and *fp* are the numbers of true positive and false positive predictions for the considered class.

Recall

The recall is the measurement to select as many web services as possible that are related to a user query. In other words, recall is a measure of the ability of a prediction model to select instances of a certain class from a data set. It is also called sensitivity, and corresponds to the true positive rate.

The recall of a query *q* is measured as follows

$$\frac{A}{A + C} \times 100 \quad (1.4)$$

Where *A* is the number of relevant web services selected and *C* is a number of relevant web services which are not selected. It is also defined by the formula:

$$\frac{tp}{tp + fn} \times 100 \quad (1.5)$$

Where *tp* and *fn* are the numbers of true positive and false negative predictions for the considered class. *tp* + *fn* is the total number of test examples of the considered class.

Accuracy

It is used as a statistical measure of how well a classification model correctly identifies a class. It is the overall correctness of the model and is calculated as the sum of correct classifications divided by the total number of classifications.

1.7 Thesis objectives

Following objectives had been approved for the work that was to be carried out in this thesis:

i) To study and explore data mining techniques for various web services, semantic

web services and existing frameworks for web services discovery mechanisms.

ii) To design a framework for enhancing web services discovery mechanism.

iii) To develop and deploy web services on the proposed framework.

iv) To test and validate the web services on the proposed framework.

1.8 Thesis contributions

The major contributions of this thesis can be summarized as follows:

i) A detailed literature survey on different approaches for web services discovery has been conducted. The various classification approaches for the web services classification have been discussed.

ii) A framework *ACRWebS* has been proposed to help users for web services discovery. This framework is two level services discovery mechanism and uses classification and ranking mechanisms to achieve better results.

iii) This approach for web services discovery process is based on the use of machine learning methods for the services classification. A novel approach for automatic classification of web services using the machine learning algorithms and leveraging NLP text processing techniques has been presented.

iv) We evaluated state-of-the-art machine learning classifiers such as NB, *k*-NN, ANN, SVM and RF and suggested the best classifier which can help to improve web services discovery process. The classified information can be used further as semantic metadata in a web service.

v) Cross-validation technique has been used to analyse the performance of machine learning models and the results have been statistically analysed.

vi) The impact of various parameters including dataset size, data pre-processing techniques has been studied exhaustively, in order to get the optimized configuration of classification models.

vii) Services are ranked based on the path-length based semantic similarity algorithm. The semantic similarity tool *WordNet* has been used to measure the semantic relatedness.

viii) The proposed framework does not require users to have technical or domain knowledge in order to consume web services. This framework is based on automatic classification of web services. Ranking of web services has been done based on information provided by a user.

ix) The problem of finding relevant services has been confined to a particular category. Proposed framework *ACRWebS* is based on a simple approach and does not recommend any changes in the existing standards of web services.

x) A comparison of the proposed approach with the well-accepted work of Bennaceur *et al.* (2011) has been done.

xi) New web services have been created to validate the results.

1.9 Thesis organisation

This thesis has been divided into 7 chapters. A brief outline of each chapter is given in the succeeding paragraphs.

Chapter 1 introduces the architecture, various components and life cycle of web services. In this chapter, we have discussed various issues related to web services discovery and presented an overview of the various existing challenges in this area. This chapter also touches upon existing solutions to address the discovery problem of web services. An introduction to machine learning techniques and their growing usage in web services discovery area has also been covered in this chapter.

Chapter 2 discusses the related work and various mechanisms for web services discovery. This chapter introduces the research background of this thesis work and provides a comparative study of research done by eminent researchers using various techniques. This chapter also provides reported results of other researchers in this area.

Chapter 3 provides details of the proposed framework for accomplishing the objective of this thesis work. The implementation and testing details of the proposed framework *ACRWebS* have been presented in this chapter. This chapter outlines proposed framework

implementation design, which comprises of various components such as NLP, Classifier, Ranking *etc.* Various diagrams of *ACRWebS,* such as system flow, functional architecture and use case have been presented in this chapter. The semantic similarity algorithm is also depicted in this chapter.

Chapter 4 describes the classification of web services using soft computing techniques. In this chapter, various prominent machine learning algorithms such as Naïve Bayes, Artificial Neural Network, Support Vector Machine, Random Forest, *etc.* have been evaluated for the classification of web services. The classification models have been evaluated statistically using cross-validation technique. In order to find the optimized configuration of a classifier to attain its best performance, classifiers have also been evaluated with their optimized configurations and combination of NLP based processing techniques. The results have been validated in order to suggest an effective and consistent classification model.

Chapter 5 describes the web services semantic similarity approach. A *WordNet* based semantic similarity approach has been presented. A comprehensive approach for test data preparation and experimentation work has been presented. The test results are statistically evaluated and compared with other approaches.

In chapter 6, we further carried the analysis of the proposed framework on the newly created web services. We have created some new web services and inputted these services to the proposed framework. The experimental results have been statistically evaluated.

Chapter 7 presents the conclusions drawn from the result of the experiments carried out in the research work. Some indicators for the future research work are also discussed. Various benefits of the proposed research work have been discussed in this chapter.

Chapter 2
Related Work

This section provides an overview of various prevailing approaches for web services discovery. The current vision of researchers is semantic web and they have proposed and devised many algorithms, architectures and standards to achieve this vision.

A brief introduction to syntactic based web services discovery has been presented in section 2.1 of this chapter. We present some previous work in the area of semantic web and ontology in section 2.2. Section 2.3 of the chapter describes hybrid approaches in this area. Section 2.4 looks at the work in the area of web services classification. Section 2.5 provides an overview of ANNs and fuzzy logic based approaches in making the web services discovery process efficient. Finally, section 2.6 provides the summary of the chapter.

2.1 Syntactic based approaches

The simplest data model is the keyword based which follows the legacy UDDI standard and the discovery mechanism. The textual description that accompanies each web service describes its functionality and is stored in the UDDI catalogue along with the tModel. The retrieval stage comprises of a user or a search program, which enters a query to the catalogue. A number of corresponding web services are returned as a response and the user browses them in order to find which one of them suits his needs. Since, there are numerous functionally similar web services available, it is required to find some criteria to distinguish

them. One of the prominent criteria in syntactic based web services discovery is QoS attributes. The services consumer requests the services repository with the QoS information and based on their requirements, UDDI is searched for the registered services. In the response, a list of services is offered based on the QoS attributes that are acceptable to both consumer and provider. There exist approaches with different methodologies in this context as given below.

Various techniques to address the QoS requirements for web services had been defined by Weider *et al.* (2007) and Liu *et al.* (2004). They proposed a framework which is dynamic and used client's feedback and monitoring to evaluate the QoS parameters for the web services. They used a model and a selection technique as the major criteria for web services discovery.

Ran (2003) proposed a web services discovery model with QoS by extending the UDDI model with the QoS information. A QoS broker based model is proposed and there are four roles in their proposed model: web services supplier, web services consumer, web services QoS certifier and the new UDDI registry. The new proposed UDDI registry is a repository of registered web services with lookup facilities. The new certifier's role is to verify services provider's QoS claims. The proposed new registry is having information about the functional descriptions of the web services and their associated quality of services attributes. Lookup could be made by the functional descriptions of the desired web services, with the required quality of services attribute as lookup constraints. The certifier verifies the advertised QoS of web services before registration.

D'Mello *et al.* (2008) proposed a mechanism for selection and ranking of web services based on the QoS broker architecture. The broker ranks the web services based on the user's QoS constraints and preferences. Some other similar broker based architectures were presented by TSAI *et al.* (2011) and Thomas (2008). These architectures focused more on the QoS specifications, using dynamic QoS mapping between servers and performance of network.

Maximilien and Singh (2004) proposed an agent framework and ontology for dynamic web services selection. QoS data of different services was collected from agents, aggregated and shared. Services quality can be determined collaboratively by services consumers and agents via the proposed framework. They also proposed QoS ontology, which captured and defined the most generic quality concepts.

Garcia and Toledo (2006) suggested an extended architecture for web services to support QoS. They proposed a QoS information derived from WS-Policy encapsulated into UDDI based repositories. UDDI is extended to include both the QoS policies and enhanced services description to improve the services discovery process. Furthermore, the UDDI extension provided the flexibility of using brokers to facilitate the web services selection process using both the functional and QoS parameters.

Zeng *et al.* (2004) proposed a simple QoS model containing five QoS criteria, namely, price, duration, availability, reliability and reputation. They applied linear programming for optimization of selection problem in services composition. However, their approach was often too complex to run-time decisions.

Farkas and Charaf (2003) proposed software architecture to provide QoS-enabled web services. They added a QoS broker in between consumers and services providers to select the web services based on QoS attributes defined in UDDI.

Serhani *et al.* (2005) proposed an extended web services architecture using a certification approach to provide services selection based on QoS attributes. A certification approach has been used to verify the quality of web services.

It has been observed that above mentioned approaches have a few challenges: these approaches do not provide proper QoS specifications. Some of the approaches have used certification based approach in order to validate the QoS values. These also lack information about the accuracy of the used QoS values and do also not provide information on the latest values of QoS.

Another disadvantage is that for discovering a specific web service, a consumer needs to provide a precise and correct query to limit the size of search space. Otherwise, the consumer is not able to take the advantage of complete search space.

2.2 Semantic based approaches

This has been observed that semantics can play a big role in improving the web services composition and discovery. But the lack of web services standards to incorporate the semantics has been a challenge for services providers and consumers. There exist several web services description frameworks which support semantics, namely, OWL-S (Martin *et al.*, 2004), WSDL-S (Akkiraju *et al.*, 2005) and WSMO (Fensel *et al.*, 2007). These proposed frameworks are based on the standard of combining WSDL files and semantics data. A semantic layer is integrated with the WSDL file or in some cases an extra semantic layer is added on top of the WSDL file. These layers help web services consumers or agents, to find the web services easily based on the provided capabilities. Semantic web services provide a platform for integrated technology by combining semantic web technologies and web services. Therefore, it becomes essential to devise a framework which can integrate semantic web design principles with web services oriented architecture.

Web services with semantics descriptions are playing a big role in improving automatic discovery and composition in homogeneous and heterogeneous environments. A semantics based services require same life cycle operations such as publishing, discovery, invocation and deployment, as that required by normal web services.

An overview of various prevalent semantics based frameworks is given in subsequent sections. Many researchers have applied ontologies on web services discovery and found these promising to semantically enrich web services. They also observed that ontologies improve the web services standards and bring structure to web services. The ontology based standards such as OWL-S, WSMO and DAML are used to describe web services and related resources. They replace existing web services by incorporating semantics, which is understandable by machines. Currently, semantic ontology languages are being used to

enhance accuracy and recall of web services selection process. However, ontology also has its own disadvantages such as maintenance, ontology matching, ontology mapping, ontology creation *etc.* which requires extra human efforts.

OWL-S is standard for OWL to describe web services and enable services providers in automatic services discovery, composition and monitoring. OWL-S uses ServiceProfile, ServiceModel and ServiceGrunding ontology to describe a service. OWL-S adds an extra layer on top of WSDL file in order to describe the web services. See Figure 2.1 for the relationship among these classes.

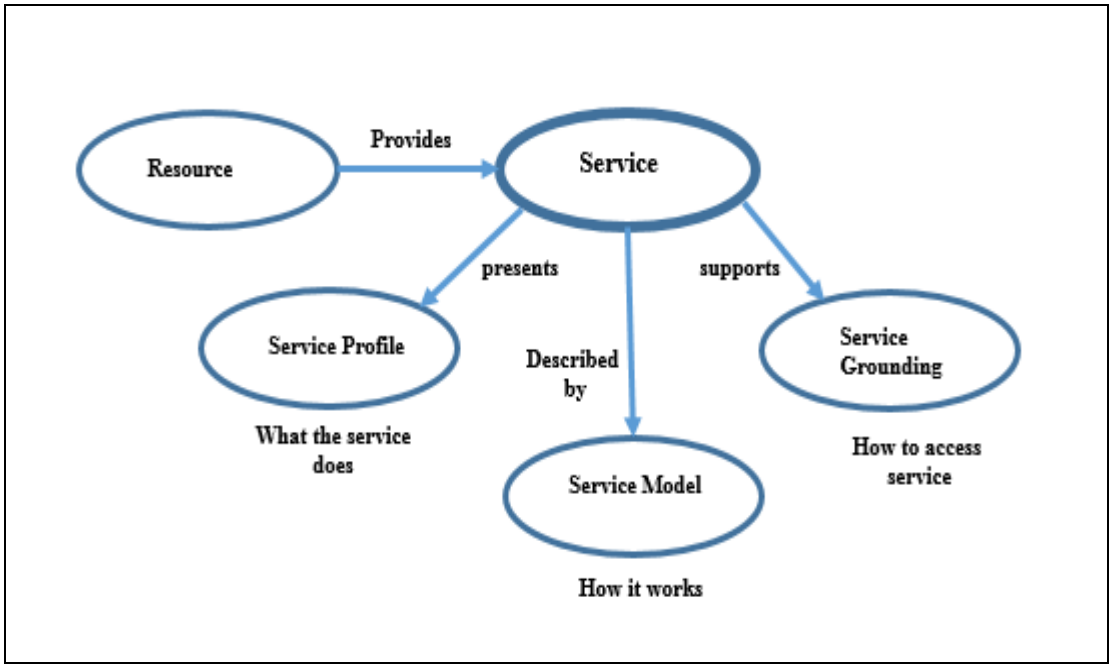


Figure 2.1: Web services ontology

2.2.1 Services profile

Web services profile defines *what a service does* and each instance of web services can have multiple profiles. The profile is specification of the functionality offered by a service. This can also comprise non-functional aspects such as references to existing ontologies,

provider information and the quality rating for the services. Additionally, a service may require other external conditions to be fulfilled. The services profile also describes the pre-conditions to be satisfied and the expected effects that result after the execution of the services.

2.2.2 Services grounding

Grounding provides information about how to interact with a web service. It provides details about the supported transport protocols. A services grounding specifies the details of how an agent can access a service. Typically grounding will specify a communication protocol, message formats, and other services specific details such as port numbers used in contacting the services.

Subsequent sections provide an overview of various semantics based approaches for web services discovery.

Lara *et al.* (2005) proposed criteria to define the web services capabilities that should be met in a semantic web services description. This criterion could be its textual description, pre-condition, post-condition and identifier.

Bruijn *et al.* (2008) proposed an empirical approach for semantic web services discovery where a collector service gathers all the WSDL file data from services and this extracted information is parsed to get the relevant information. A document of terms is generated which is transferred to a Vector Space Model and indexed using Latent semantic analysis (LSA).

A hybrid approach has been proposed by Shafiq *et al.* (2010) towards dynamic web services discovery technique. This approach is based on Bayesian classification and light-weight semantic that classifies different available web services.

A semantic web services mining model that allowed finding services dynamically, from OWL-S based services has been presented by Ramu *et al.* (2011). Their work detected the services based on semantic relevance using an ontology analysis. They also proposed an

online search model to detect the semantically relevant web services on the web. They have mainly concentrated on the discovery in OWL-S integrated development environment (IDE) and changes to the UDDI registry. Their proposed technique provided algorithms for the efficient use of OWL-S ontologies in UDDI, which could be easily applied to any existing OWL ontology.

The semantic description of web services supports an effective web services discovery process. Nayak and Lee (2007) extended the semantic representation of web services. They grouped the similar web services using the semantic web services clustering (SWSC) method in order to improve the services discovery. SWSC leveraged OWL-S ontology and *WordNet* lexicon to enhance the descriptions with semantics. SWSC utilized the clustering algorithm to group web services into domains. Their similarity function also considered the matching of text and the context of the terms. The context captured the semantic influences as the similarity relatedness between the web services.

Sotolongo *et al.* (2008) proposed an algorithm based on information that applied the lexical database *WordNet* together with a linear discriminant function. This algorithm calculated the degree of similarity between words and their relative importance to support the development of distributed applications using web services. The proposed algorithm used the semantic information contained in the WSDL specifications and ranked web services based on their semantic similarity.

Karimpour and Taghiyareh (2009) proposed a solution that allowed annotating semantic tags on the message parts of web services. They solved the web services discovery problem with a domain neutral annotation technique. They presented a new approach based on *WordNet* concept for enhancing web services semantically. The advantage of the proposed approach was that it facilitated developers to add semantic information without using any specific ontology and this had an advantage of cost effectiveness as the efforts for the ontology management and semantic annotation were reduced.

Ganapathy and Surianarayanan (2010) have proposed a two-stage filtering approach. Their approach is based on the implicit semantics of web services description. *WordNet* is

proposed to identify candidate services using semantic reasoners. The first phase of the approach filters out irrelevant web services. In the second phase of the approach, the services are further filtered based on their trust score. In this phase, the trust score of each web service is compared with a user-defined trust score using a trust ranking algorithm. Only those candidate services with trust score greater than the user-defined trust score are considered for semantic matching of web services.

Peng (2010) proposed a two level semantic web services discovery method. This method used semantic word from *WordNet* to annotate services and their interfaces. They used two levels to discover the target services. At the first level, they computed services similarity degree. At the second level, they computed services interface similarity degree. A threshold value is used to eliminate irrelevant services in each of the levels.

Becker *et al.* (2010) presented Themis-S, a prototype of an ontology based natural language services discovery engine. In a series of experiments, they evaluated the retrieval effectiveness of Themis-S in comparison with other information retrieval models. The experiments show that Themis-S, using *WordNet* as general purpose ontology, can outperform other systems by applying syntactic information retrieval models such as the Vector Space Model (VSM) or the Probabilistic Relevance Model (PRM).

Wang *et al.* (2006) designed a QoS ontology language in the context of WSMO (Web Services Modeling Ontology). They presented an associated selection mechanism using optimum normalization algorithm.

In these approaches, it has been observed that ontologies techniques require extensive processing time and are more expensive in terms of the design and implementation. An expert is required to maintain and create ontologies, which is a cumbersome task.

2.3 Hybrid approaches

It has been observed from the various surveys that main approaches for web services discovery are based on a combination of keyword and ontology matching methodologies. Some of the prominent approaches are mentioned in the below sections.

Tsai *et al.* (2003) used both the approaches of keyword match and ontology for the discovery of web services. They considered following attributes of web services while match-matching the user query such as services description, provider's information, services annotation by users, operational description, categories and QoS. The similarity between the user query and web services is calculated using ontology of web services. They tested the semantic similarity at three levels as mentioned below.

Exactly: where two classes of web services match exactly.
Subsume: where one concept of web services is the super class of the other web services.
Mismatch: where two classes of web services are not related at all.

Tian *et al.* (2004) assigned weights to all the attributes of web services for measuring the overall similarity. They proposed a method called analytical hierarchy process (AHP) for multiple criteria decision making. The overall similarity score is calculated as a weighted sum of similarity of all the considered attributes. They compared proposed approach with other text and ontology based approaches. It has been shown that the hybrid based approach provided better results than using an individual approach.

2.4 Classification approaches in web services discovery

Classification is the process of partitioning a given dataset into disjoint classes using a class attribute. Recently, machine learning models have been playing a significant role in web services discovery mechanisms and it has been shown that use of these models improves classification accuracy. Web services features are transformed into a format which can be fed to machine learning algorithms.

In the process of services classification, a machine learning model is trained using any of the classification methods *i.e.* NB, *k*-NN, ANN, SVM and RF. A feature vector is extracted from the functional descriptions of WSDL files, such as input and output descriptions, function arguments *etc.* for the training purpose. In some cases, QoS features are also extracted for the services selection process. Extracted features from the WSDL files are pre-processed using stemming, tokenization vector and filter stop words to create a clean training data. The created training data is input to the machine learning models for classification purpose.

There are three types of categories for classification models, namely, supervised, un-supervised and semi-supervised models. Figure 2.2 shows various models under these categories. In literature, different architectures using machine learning have been proposed to enhance web services discovery process. Figure 2.3 shows the taxonomy of web services discovery mechanisms using machine learning methods. We have shown taxonomy for supervised and un-supervised based machine learning methods.

There are many proposed methodologies in the literature using machine learning methods for the web services classification area. The proposed approaches can be classified based on the type of machine learning technique used for match-making. Below sections provide an overview of various classification approaches used by eminent researchers.

The web services discovery mechanisms are using semantics based techniques extensively. Shafiq *et al.* (2010) highlighted the fact that both the complex semantic descriptions and reasoning methods take significant processing time. They invented a hybrid approach with the combination of semantics and machine learning to improve the web services discovery process. They used non-functional properties of web services as light-weight semantics and used the Bayesian classification technique to classify web services dynamically. They presented that their approach is simple and efficient in terms of processing time.

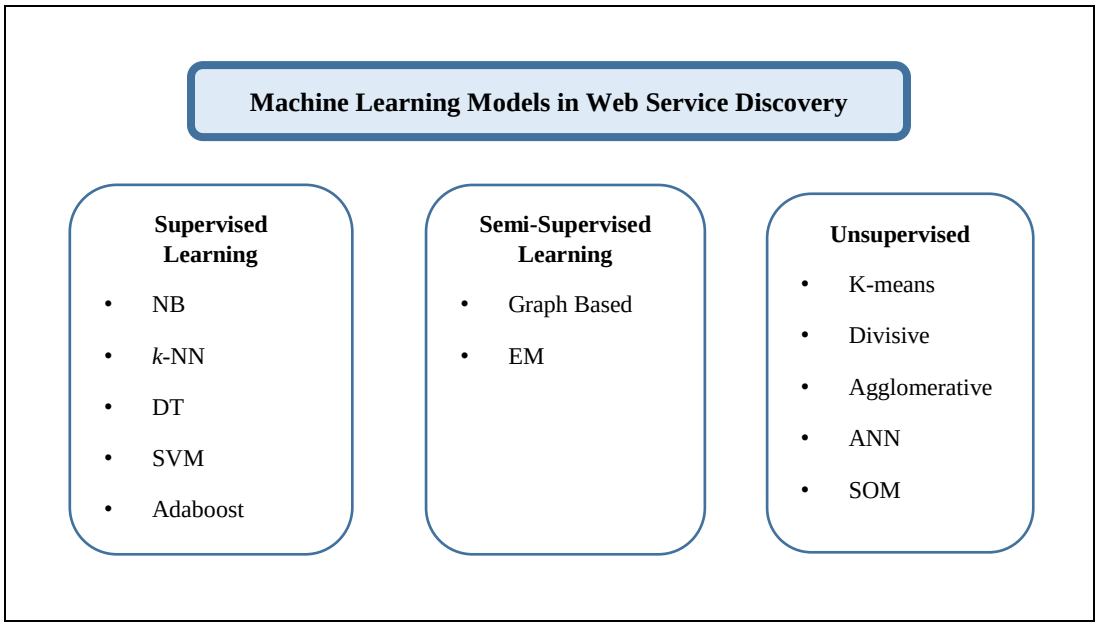


Figure 2.2: Machine learning models for web services discovery

The automatic determining of services category is useful in semantics annotation and matching of web services. Crasso *et al.* (2008) presented an AWSC (Automatic Web Service Classification) framework. They have used a combination of text mining and machine learning technique to improve classification process precision.

When the category set is large, the conventional classification methods usually require a large training sample collection, which is hardly available in real world settings (Wang *et al.*, 2010). They have proposed a novel method which classifies medium or big category data set by utilizing the descriptive information of the sub-categories as the sample data. They have presented a new feature selection approach based on semantic similarity of concepts to reduce feature space dimensions, without the loss of document features. They have used a relationship between categories of web services and standard descriptions.

A generic non-functional based web services classification algorithm has been proposed by Jamous and Safaai (2011). This classification algorithm depends on information supplied by web services provider at the time of services registration. They proved mathematically as well as experimentally the usefulness and efficiency of the proposed algorithm.

The semantic annotation of services plays an important role for classification purpose. The challenge is annotation should be in agreement with a specific ontology and services description should be related to other services description by Bruno and Canfora (2005). They proposed an approach to classifying services to the specific domains automatically, identify key concepts inside services description and build a lattice of the relationship among services annotation. The obtained results are promising for getting useful insights in the services publication and retrieval.

The services interface descriptions have been used by Bennaceur *et al.* (2011) for automatic classification of web services and this has improved the services discovery process considerably. They have used SVM model to build categorizers of web services interface descriptions. They evaluated a number of machine learning methods and observed that SVM provided the best results. They had also evaluated how the performance is influenced by the design of the feature extraction components.

The researchers realized that more expressive description of web services are the need of the hour. Most of the recent approaches suggest that ontology should be used to describe web services and for the annotation purpose. Researchers felt the need of automatic annotation mechanisms and created a framework called METEOR-S (Oldham et al., 2004). They used Naïve Bayesian classifier and showed that their framework can match ontology in less time with improved accuracy.

A new approach to grouping web services into clusters of similar functions by mining web services documents has been proposed by Ismaili *et al.* (2009). They have generated ontology via hidden semantic patterns and proposed an approach to identify the cluster center that combines services similarity with the term frequency-inverse document frequency (TF-IDF) values of services names to improve the utility of clusters.

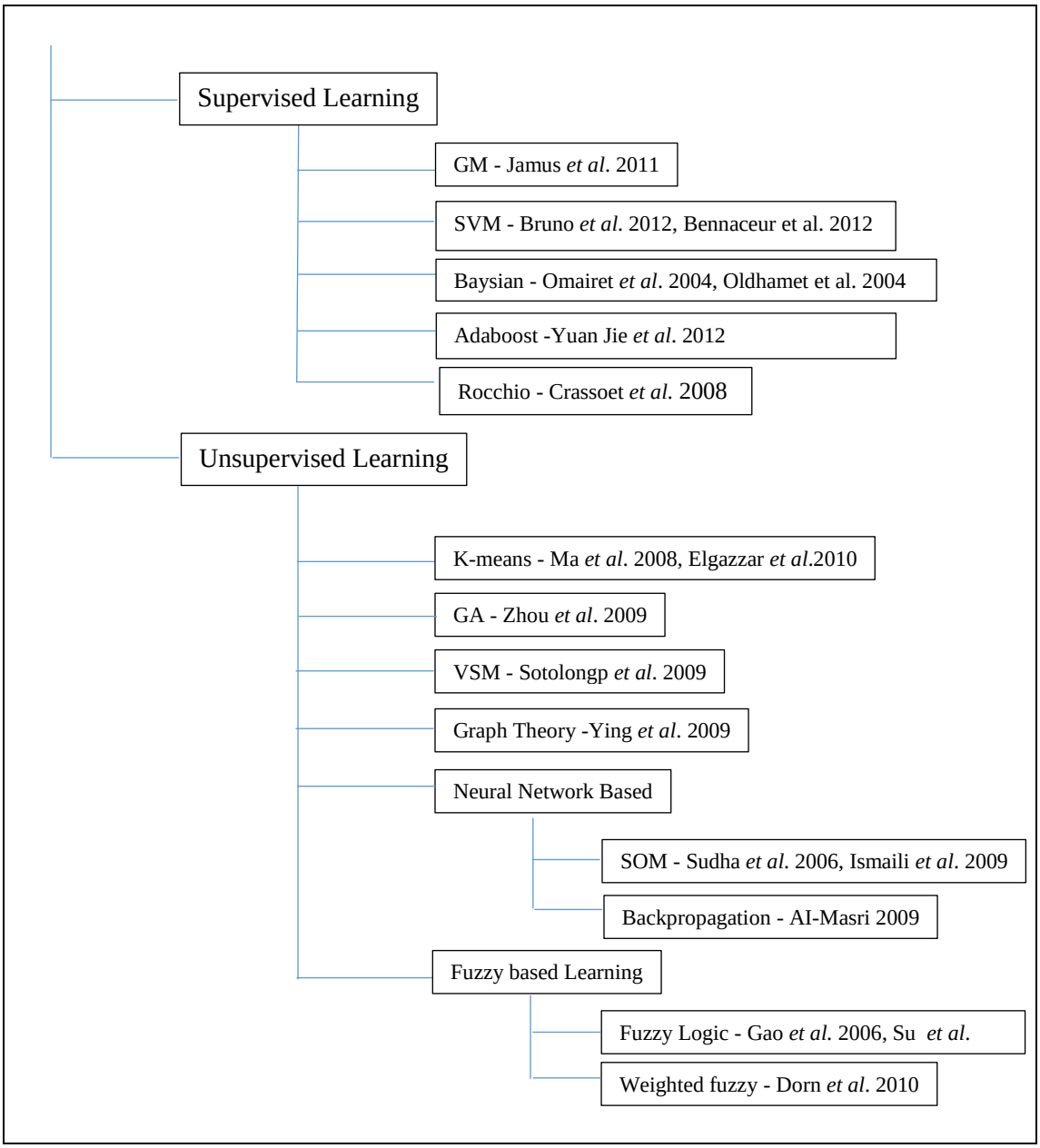


Figure 2.3: Taxonomy of web services discovery mechanisms using machine learning methods

The ensemble methods have also been used widely for the web services discovery problem. These are learning algorithms that combine a set of classifiers and then classify data by taking a weighted vote of each classifier prediction. A new approach has been proposed by Yuan-jie and Jian (2012) where they have applied automatic web services semantic annotation and evaluated the accuracy of the ensemble learning classification

method. They have observed that the returned accuracy is much higher than any other classification method. It has been shown that annotation and the mapping between concepts and ontology directly affected the outcome of the classification accuracy. After manual calibration, the accuracy of each classifier was improved to some degree. They emphasized that WSDL file annotation is necessary to improve accuracy and efficiency of classification models.

The AdaBoost ensemble algorithm has also been used for the web services classification. AdaBoost is a short form of “Adaptive Boosting" and is a machine learning meta-algorithm. This algorithm can be used in conjunction with other learning algorithms to improve classification performance. Varguez-Moo *et al.* (2013) proposed a web services discovery approach using machine learning algorithms NN, SVM and AdaBoost to classify web services. They have used QoS parameters as the semantic information and concluded that AdaBoost algorithm provides the best mean percentage of precision. Naïve Bayes algorithm provides the best mean percentage of recall and accuracy.

Saha *et al.* (2008) used tensor space model and rough ensemble classifier for web services classification. They have used two steps approach to improving the web services classification results. In the first step, they used tensor space model and achieved better classification results compared to the existing results. In the second step they further improved results by using rough set based ensemble classifier.

A comparison of various prominent machine learning based approaches for web services discovery has been shown in Table 2.1. These approaches are categorized based on parameters such as the used approach, semantic support, QoS support, ontologies, ranking method, machine learning method, annotation capability, self-learning and NLP. It is evident from the results that different approaches have been composed by using a combination of different methodologies. Analysis of the above matrix suggests that most of the proposed approaches are using various classification techniques with semantics and ontologies to get better results for web services discovery.

Table 2.1: Comparison of machine learning approaches for web services discovery with various parameters

Contributors	Approach	Semantic support	QoS support (y/*n*)	Ontology language	Ranking method (y/*n*)	Machine learning method	Annotation capability	Self-learning	NLP used
Lu	Unsupervised	√	×	×	√	×	×	×	×
Ram *et al.*	Unsupervised	√	×	×	×	ANN	×	√	×
Nayak and Lee	Unsupervised	√	×	OWL-S	×	HA	×	×	×
Zhou and Lee	Unsupervised	√	√	WSMO	√	GA	×	×	×
Sotolongo *et al.*	Unsupervised	√	×	×	√	VSM	×	×	×
Ying	Unsupervised	√	√	√	×	GT	×	×	×
Jamous *et al.*	Supervised	×	√	×	×	GM	×	×	×
Su *et al.*	Unsupervised	√	×	OWL	×	×	×	×	×
Mohanty *et al.*	Supervised	×	√	×	√	NB, MB and TB	×	×	×
Satish and Wahidabanu *et al.*	Supervised	√	×	OWL-S	√	SVM	×	×	×
Wang and Stroulia	Supervised	√	×	×	×	SVM	×	×	×
Shafiq *et al.*	Supervised	√	√	×	×	Bayesian	×	×	×
Yuan-jie and Jian	Supervised	√	×	√	×	Ensemble learning-Adaboost	√	×	√
Bruno and Canfor	Supervised	√	×	√	×	SVM	√	×	×
Bennaceur *et al.*	Supervised	×	×	√	×	SVM	×	×	×
Oldham *et al.*	Supervised	√	×	√	×	NB	√	×	×

Continued on next page

Markov Blanket - MB

Tabu Based - TB

Generic method - GM

Entropy-Based - EB

Hierarchical Agglomerative – HA

Graph theory – GT

Vector Space Model – VSM

Genetic Algorithm – GA

Table 2.2 shows the performance comparison of various machine learning algorithms, as has been observed from various studies.

Table 2.2: Performance of machine learning algorithms in web services classification

Reference	Algorithm Used	Accuracy achieved	Remarks
Swami *et al.*	Augmented Naïve Bayes	80.72 %	Augmented Naïve Bayes, Tree Naïve Bays, Sons & Spouses approach predicted better classification accuracy of 80.72% than other techniques.
	Naïve Bayes	79.89%	
	Markov Blanket	72.73%	
	Augmented Markov Blanket	72.73%	
	Sons & Spouses	80.72%	
	k-NN	39.59%	
Bennaceur *et al.*	Naïve Bayes	79.38%	They used text mining and machine learning for classification of web services. Results show that Rocchio algorithms provide better accuracy in comparison to other algorithms.
	Rocchio	85.08%	
	Naïve Bayes	86%	
Heß *et al.*	SVM	73%	They concluded that an ensemble approach that treats web services as structured objects is more accurate than an unstructured approach.
	Naïve Bayes	69.71 %	
Yuan-jie *et al.*	SVM	77.31 %	They concluded that ensemble learning classification method provided the best accuracy.
	AdaBoost	89.15 %	
Mohanty *et al.*	Naïve Bayes	85.62%	Study shows that Naïve based Bayesian network performs better than other two models.
	Markov blanket	81.36%	
	Tabu	82.45%	

2.5 Artificial neural networks and fuzzy logic based approaches

The Artificial Neural Networks (ANN) and fuzzy logic have also been playing an important role in making the services discovery process efficient. An ANN is made up of artificial neurons that are connecting with each other. Many research articles have appeared recently for the efficient and effective selection of services using ANN based techniques. This section provides a summarized view of literature study for web services discovery using ANN and fuzzy Logic.

Ram *et al.* (2006) proposed an approach using an un-supervised artificial neural network and empirically evaluated the approach using real web services descriptions. They have combined clustering techniques with string matching and leveraged the semantics of XML based services specification described in WSDL documents. They have used *q*-grams string matching method for creating input features for cluster analysis. They further suggested other approaches using vector-space model for string/document matching. They used a single clustering method for each experiment and suggested that future research may evaluate the effect of combining multiple clustering methods. Their clustering based approach can be integrated with keyword search and predefined category based browsing, so that users can combine multiple search strategies in a flexible manner.

Huang *et al.* (2006) presented a moderated fuzzy web services discovery approach to subjective model and fuzzy opinions, and to assist service consumers and providers in reaching a consensus. Their method achieved a common consensus on the distinct opinions and expectations of service consumers and providers. Their process is iterative such that further fuzzy opinions and preferences can be added to improve the precision of the process.

A new approach that combines GHSOM (Growing Hierarchical Self Organizing Map) clustering technique with Latent Semantic Indexing has been proposed by Ismaili *et al.* (2009). The overall process of discovering web services included: web services information processing, clustering heterogeneous services according to semantic similarity and ranking

web services by their relevance. They suggested that further research should concentrate on building a tool for evaluation of the effectiveness of proposed method.

Goyal *et al.* (2009) proposed a density based clustering algorithm which is an incremental version of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Their approach provides the capability for discovering clusters of arbitrary shape. They have used *R*-trees as the data structure to hold the multidimensional data to cluster. DBSCAN is an Incremental algorithm which supports data update to and from clusters considering one data point at a given time. They proved that the proposed approach is efficient in terms of region queries performed.

The back-propagation based neural network is also useful for discovering web services of user interest as has been proposed by Al-Masri and Mahmoud (2009). The use of neural networks provided a way to optimize the selection of the best available web services. They have used non-functional (QoS) properties of web services (response time, throughput, availability and compliance) to improve the probability of having relevant output results. The observed results suggested that the use of neural networks in discovering the most suitable web services is quite encouraging.

The hybrid approaches based solutions are widely used in web services discovery as proposed by Gao and Stucky (2009). They populated a taxonomy structure by combining human knowledge with the technique of artificial intelligence. The web services were converted into standard vector format through WSDL document. The self-learning neural network based learning algorithm and clustering algorithm were designed and implemented to classify the web services automatically. Laplace operator has been playing an important role in numerical analysis. It is common to use this operator in machine learning for semi-supervised learning and clustering. Deep *et al.* (2009) proposed a methodology using Laplace probability density function to share information between two particles. They introduced two new algorithms, namely, Laplace Crossover PSO with inertia weight (LXPSO-W) and Laplace Crossover PSO with constriction factor (LXPSO-C). They presented empirical results, which validates the effectiveness of new approach.

Shehzad and Javed (2010) proposed web services mining framework based on fuzzy set and rules that proactively uncovers the interesting and useful individual web services. The fuzzy set has been generated based on the specified scope and weights assigned to each member of fuzzy set in order to optimize the mining process.

A new fuzzy semantic clustering algorithm which assisted in discovering a group of similar web services through an individual query has been proposed by Gholamzadeh and Taghiyareh (2010). They used semantic analysis based on supported ontology. A similar approach to limit the number of required services has been proposed by Dorn and Dustdar (2010). Their approach fulfilled the required capabilities with the use of functional specification of individual services. This approach stroked a balance between finding one service that matched all required capabilities and having one service for each required capability. They have introduced a weighted fuzzy clustering algorithm which detected implicit services capability groups. The clustering algorithm considered capability importance and services fitness to support those capabilities. They have used a real world dataset to successfully demonstrate the effectiveness and applicability of service aggregation.

Su *et al.* (2012) proposed a combination of multiple techniques for services discovery framework which incorporated semantic web, fuzzy logic, linguistic variable and multi-phase matching to get the most appropriate services against a consumer's request. The main advantages of the proposed discovery method are: (i) it can formally describe not only the capability information, but also the vague information of web services and implemented approximately reasoning based on ontology semantic, linguistic variable and fuzzy logic. (ii) Multi-phase matching has been executed on different services abstract level that could improve the efficiency and accuracy of services discovery.

Chakraverty and Sukanta (2016) have explored fuzzy approaches for the solution of uncertain Stochastic Differential Equation (SDE) model. This model plays an important role in various areas including computers, physics, chemistry, mechanics, biology,

microelectronics *etc*. They have analyzed the SDE with fuzzy parameters. They combined the fuzziness and stochasticity to model the problem and proposed methods to handle them.

ANN and Fuzzy Logic based approaches have a larger computational burden. Models based on ANN and Fuzzy Logic approaches are not appropriate if the training dataset has less number of input features. These models are also prone to overfitting.

2.6 Chapter summary

This chapter provides a literature survey of machine learning models in web services discovery. One can observe that worldwide tremendous development and exploration work has been done in this area. It is a challenging task to effectively discover the desired services that logically or contextually matches the needs of their consumers.

As evident from literature, current web services discovery mechanisms focus more on how web services can be accessed rather than what their offered capabilities or functions are. Semantics can express capabilities of web services and improve the searching for consumers. This match can be further improved by narrowing down the web services space using classification techniques, which should provide significant opportunities to make the search more efficient.

In this chapter, we have highlighted and summarized the key points of work done by various eminent researchers using conventional techniques. Most of the literature studies suggest using the machine learning methods along with combination of multiple strategies can provide better results. These combinations could be keyword based search in category based browsing, natural language processing techniques, semantics representation of concepts, searching based on non-functional parameters.

Chapter 3

Proposed Framework for Web Services Discovery

This chapter outlines the framework proposed in this work for web services discovery. A two level framework called *ACRWebS* has been proposed using classification and ranking mechanisms to address the problem of web services discovery. This framework offers an automatic categorization and selection of best web service as per the query of web services users.

The framework *ACRWebS* initiates the web services discovery process by taking input as services category, *e.g.*, *weather*, *currency etc.* from the users. Once the user selects a business category, they can query the repository to select the most appropriate web service. On entering the user query, a set of web services from the repository are shown to the user. The returned services are ranked according to the similarity of user query with the web services. If the user changes the query, the rank of the returned services changes accordingly. In the following sections, we will discuss the presented approach in detail. Section 3.1 presents design and architecture of the proposed framework. Sections 3.2 presents use case view of the proposed approach. Section 3.3 presents the functional architecture. Section 3.4 presents the sequence diagram of *ACRWebS*. Section 3.5 shows the comparative analysis of *ACRWebS* with other approaches and Section 3.6 provides a summary of the chapter.

3.1 *ACRWebS*: Design and architecture

This section introduces the framework of *ACRWebS*. This framework provides semantic based discovery of web services in two phases. The first phase of *ACRWebS* is responsible for assigning a web service to one of the candidate categories using a classification method. The second phase of the framework enables users to rank services as per the specifications provided in the user query.

Figure 3.1 shows the high level diagram of the proposed framework. As shown, the main components of the framework are feature extractor, NLP techniques, classification and ranking of web services. The details about the various parts of the framework are described in below sections.

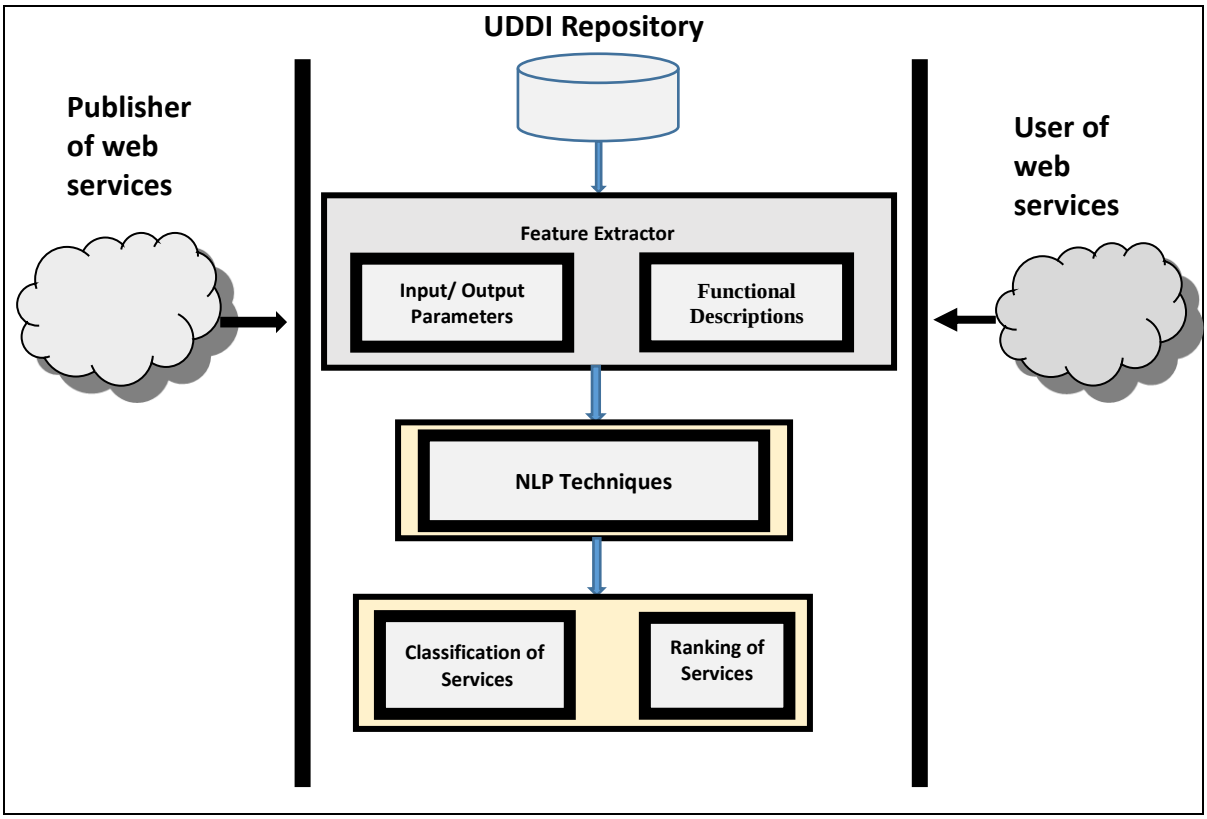


Figure 3.1 High level architecture of *ACRWebS*

Publisher of web services: This is a standard procedure for publishing a web service into the repository of web services. A services provider describes their services using the WSDL files. These definitions are published into a repository of services, which could be a UDDI. Other forms of private repositories can also be used.

User of web services: A services user issues one or more queries to the repository to find a suitable service and determines how to communicate with that service using the WSDL file of the service. Once a service has been selected by the user, he can consume the service in his application.

Feature extractor: This module is responsible for extracting the features of web services such as input/output parameters and functional descriptions from the WSDL files of web services. These features represent semantic information of web services.

NLP techniques: This module is responsible for pre-processing of extracted features from WSDL files, in order to refine the collected dataset for the experimental work.

Classification of services: This module is responsible for the automatic classification of web services into various business domains using the machine learning algorithms.

Ranking of services: This module is responsible for ranking of selected web services based on the similarity score with the user query.

3.1.1 System flow diagram

Figure 3.2 depicts the system flow chart for the proposed framework *ACRWebS*. This flow chart diagram consists of two main processes. Data from the WSDL repository is pre-processed and system assigns a pre-defined category to a newly registered web service in the repository using a classification model. During the process of services selection, a user can select a category and the user query is matched with different services from the selected category. A similarity score for each service is calculated and the services are ranked based on their score. A user can choose the highest rank service and consume in his application.

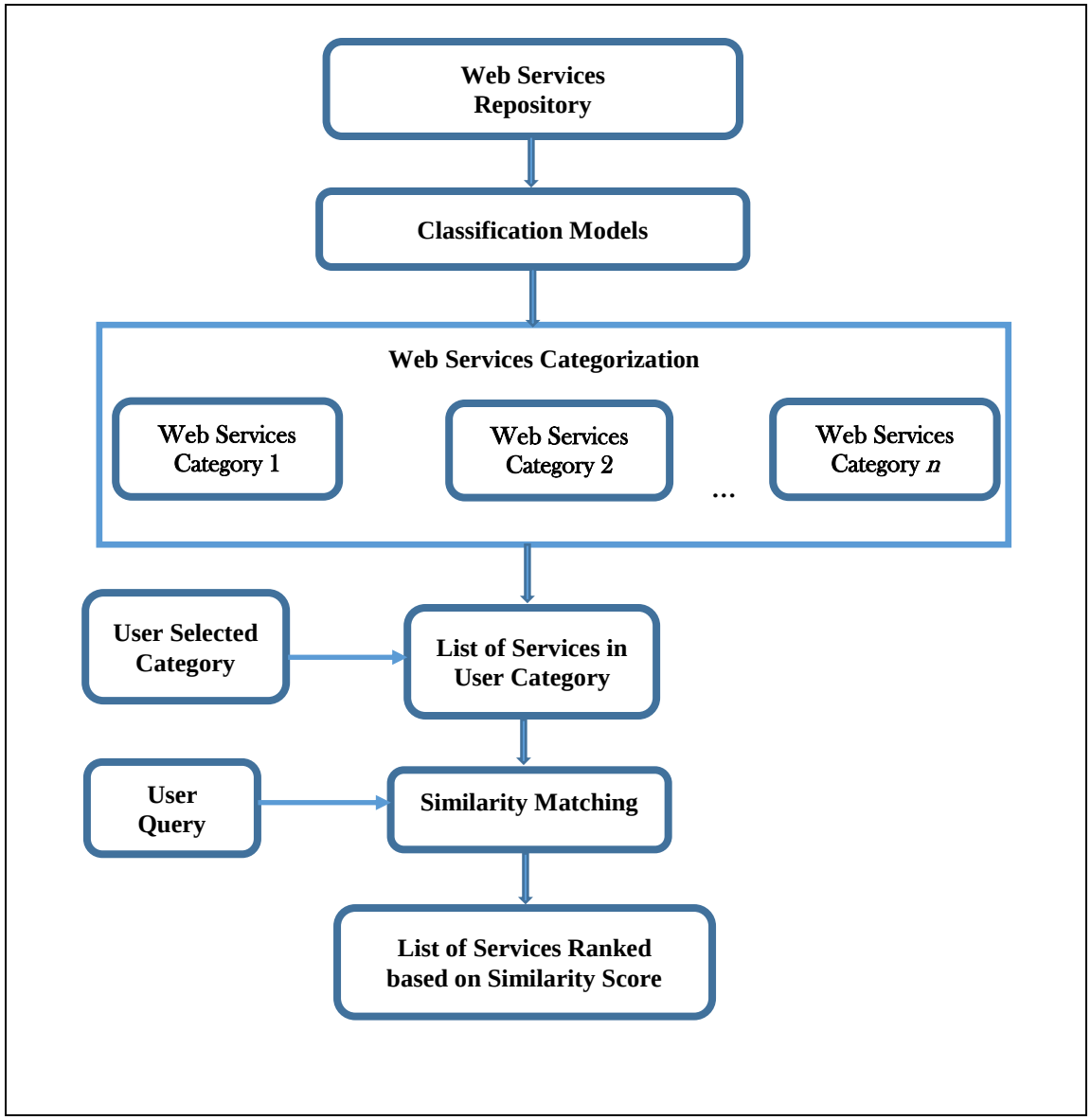


Figure 3.2: System flow diagram

3.2 Use case view of *ACRWebS*

Figure 3.3 depicts a use case diagram of *ACRWebS*.

Brief description

This use case describes how a web services user can use *ACRWebS* to discover a required web service.

Actors of use case:

- Web services user
- Web services repository with annotated classification

Preconditions of use case:

- Web services repository has registered web services

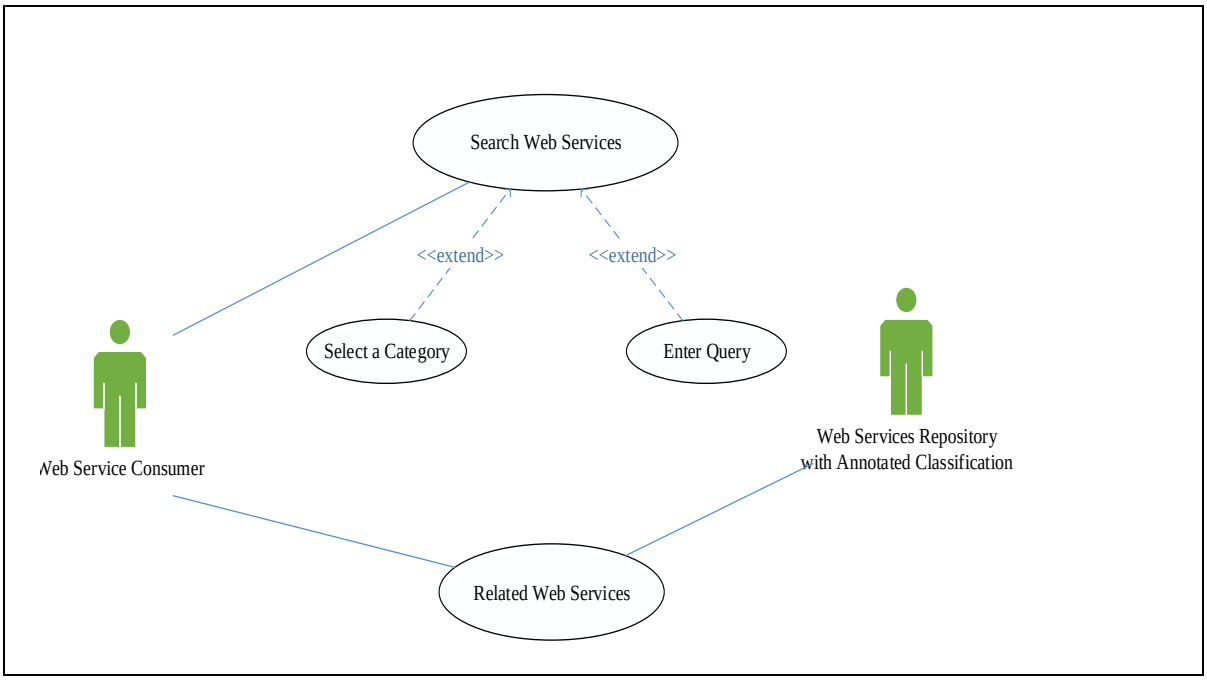


Figure 3.3: Use case of *ACRWebS*

Basic flow of events

i) The use case begins when the user (consumer) initiates the web services discovery.

ii) The user interface provides a list of service categories.

iii) The user selects a category based on the business need.

iv) The user enters a query with required services specifications.

v) The services discovery framework returns a list of web services, ranked in the order of similarity score with the user query.

vi) The use case ends successfully.

Post-conditions

- The user has a list of ranked web services.

3.3 Functional architecture of *ACRWebS*

A functional architecture depicts architecture of a system from a user perspective. The various functions in this architecture help domain business experts to develop an enterprise IT systems of an organization. Figure 3.4 shows the proposed functional architecture of *ACRWebS*.

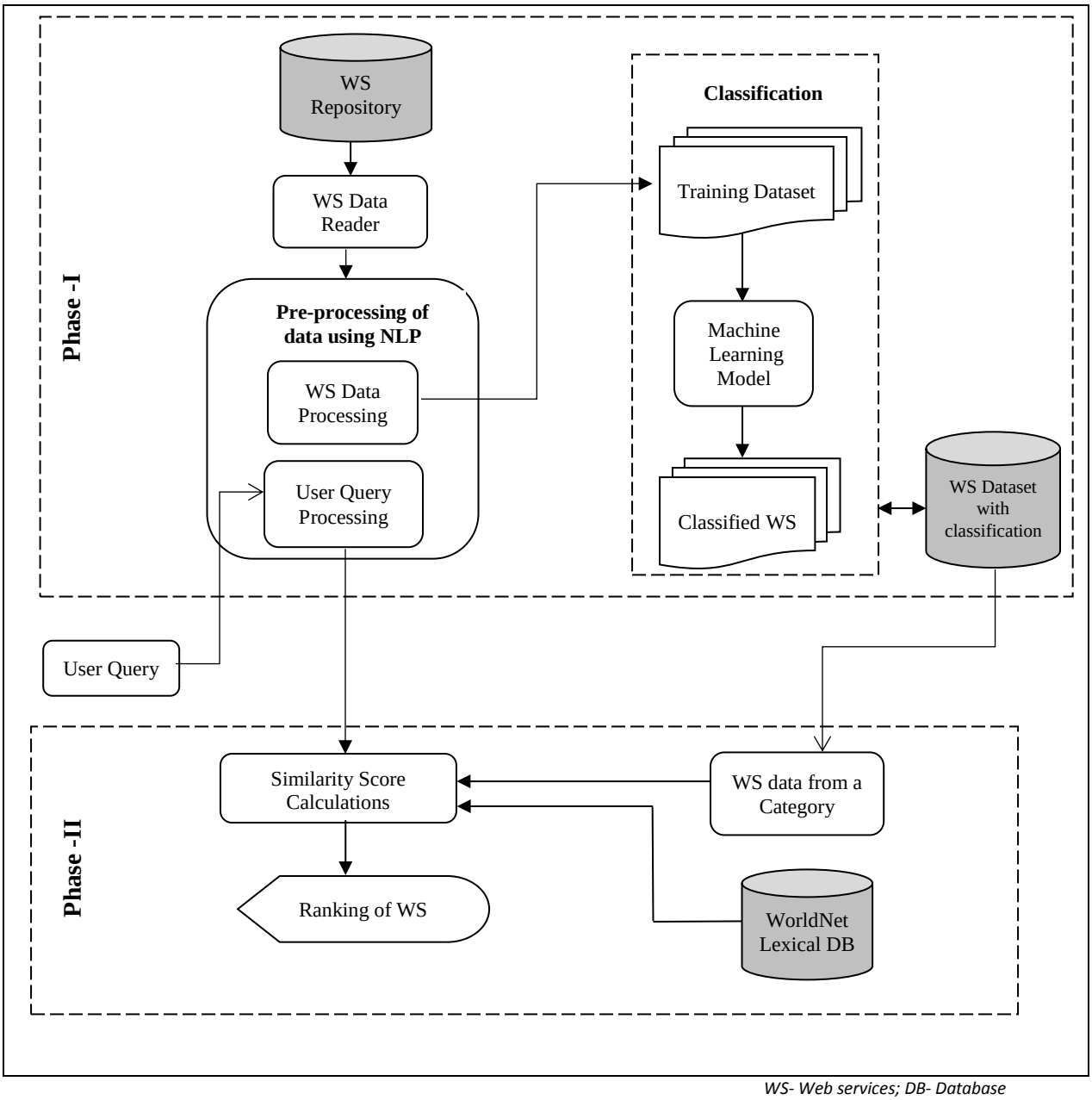
The proposed framework consists of two phases. The modules covered in these phases include web service data reader, pre-processing of data using NLP, classification, user query processing, ranking of web services, similarity score calculations. The description of these modules is given in below sections.

3.3.1 Web services data reader

A services data reader component is required to extract the operation names and arguments from respective WSDL files (all terms under tag <element name>, <documentation> and <message name>). One can either use an existing XML parser tool or this task can be done manually.

3.3.2 Pre-processing of data using NLP

Web services description is processed in order to extract useful information. We have used NLP technique for the pre-processing of web services descriptions. Following steps have been used for preparation of training data for experimentation purpose.



WS- Web services; DB- Database

Figure 3.4 Functional architecture of *ACRWebS*

Tokenization: This step splits the text into a sequence of tokens. The option of splitting the text based on *non-letter* characters mode has been used.

Word stemming: This step retains the root form of a word using the *porter stemmer* algorithm. For example, the labeling word is converted to a label. This step removes the prefixes and suffixes of each word.

Filter stop word: This step removed words that have no essential meaning and appears very frequently in a WSDL file. For example, articles (a, an, the), prepositions (in, of, at, *etc*.), conjunctions (and, or, not *etc*.) were removed from the services data.

Dimension reduction: This step is used to reduce the dimension of the data space by removing irrelevant or less relevant features of data.

3.3.3 Classification

This module of phase I uses machine learning methods to train a classification algorithm. Pre-processed data of web services is used to train a machine learning algorithm. This part of the framework, annotates a new web service with a category. We have exploited the web services internal structure to improve classification accuracy. There exist several independent sets of features which provide a hint about the class of web services. The WSDL file tags contain functional features, which semantically represents the information about a web service and can be used to classify web services into relevant categories as shown in Figure 3.5.

A classifier is trained with the data of web services, collected with the help of feature extraction process. The training data has a pre-defined label for each web service and is fed to a classification model to train the model. Un-labeled web services test data is fed to the trained model and the model assigns a label based on its learning. This step has the advantage that it reduces the number of candidate services for the ranking phase. Only the services in a specific category are passed to the ranking phase for further processing.

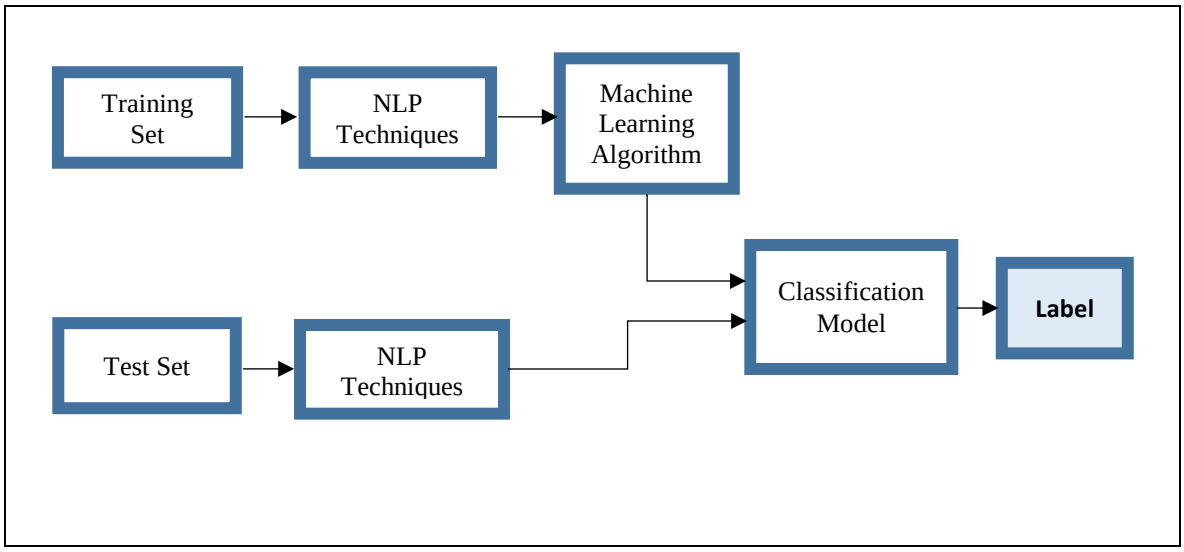


Figure 3.5: Classification in *ACRWebS*

One of the challenging activities for classification problem is the selection of most suitable machine learning method. We have carried different experiments to evaluate the classification models such as NB, *k*-NN, ANN, SVM and RF as these models have been very successful in classification problems. The performance of a classification model depends on various parameters, such as configuration of a model and the characteristics of the data to be classified. Chapter 4 covers detailed description of various techniques for web services classification area.

3.3.4. User query processing

A user query is pre-processed via the same steps as that has been used for web services data pre-processing for training a classifier. The NLP techniques such as creating tokens and stemming of words have been used for user query processing.

3.3.5 Ranking of web services

This component of phase II returns the ordered list of web services based on the similarity score. This section provides detail about the ranking mechanisms used in the proposed framework.

Once the services have been assigned a category, the task of services discovery is done to some extent. The existing approaches may yet provide a bunch of services and users have to manually go thru each service, before making a decision for selecting a particular service. So, there remains some complexity while selecting the correct user services, which accomplishes their requirements.

The proposed ranking approach is based on the concept that how much two words match each other and accordingly calculates their similarity score. We have used a similarity matching process based on the semantic distance measure. Chapter 5 provides a comprehensive analysis of this part of *ACRWebS*. A threshold level is also considered in the match-making process to decide whether a service should be sorted for selection or not, based on minimum score criteria. If a web service score is below the threshold level, that service is assumed to be irrelevant and is rejected by the approach.

3.3.6 Similarity score calculations

Considering the capabilities and performance of *WordNet* for calculating the semantic similarity, we have introduced an approach based on *WordNet* to find semantically similar web services with a user query. We have used WordNet as this is a proven tool and is also known for giving good results on lexical data. This tool is extensively accepted in many fields where NLP based techniques are employed. WordNet has also been leveraged in IR systems, document structuring and for automated sense-disambiguation.

Semantic similarity is a confidence score that reflects the semantic relation between the meanings of two words. This approach provides a more precise similarity measure between a user request and published web services. Finally, all web services are listed based on their calculated similarity score.

A fragment of *is-a* relation between various concepts in *WordNet* is shown in Figure 3.6. In the taxonomy, the deeper concept is more specific and the upper concept is more abstract. Therefore *vehicle* is more abstract than *bicycle* and *transport* is more abstract than the *vehicle*. The *Object* is the most abstract concept. This Figure shows an example of the hyponym taxonomy in *WordNet* used for path-length similarity measurement. The *WordNet* based semantic similarity measures can be classified into three categories, as mentioned in below sub-sections.

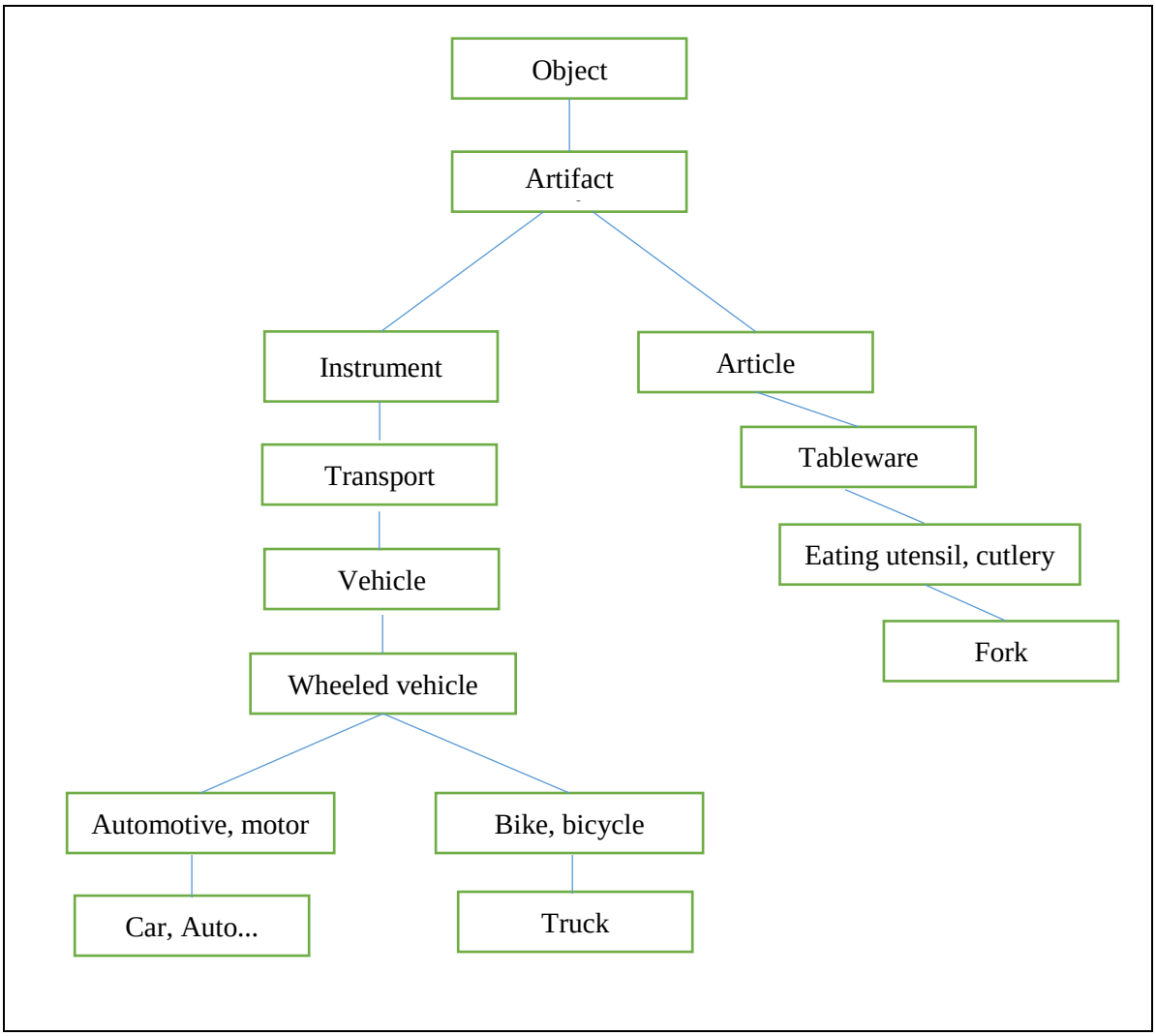


Figure 3.6: Example of *WordNet* taxonomy

There are several other methods of *WordNet* to find similarity between words.

3.3.6.1 Information Content methods

This method is based on similarity measure of the information content of each concept. The weakness of this method is that it is a time-consuming work to analysis the corpora for estimating the information content values. Also, the unbalanced contents of the employed corpora may significantly decrease the accuracy of measurement.

3.3.6.2 Hybrid methods

These methods combine the information content theory and the structure information from *WordNet* to estimate the semantic similarity between the given words. But, these methods may underperform as the handling of the corpus is not well balanced.

3.3.6.3 Edge based methods

These methods assess the semantic similarity by calculating the length of edges on the shortest path between the words in *WordNet.*

We have used the path based approach of *WordNet* for calculating the similarity. This approach has been preferred for calculating the similarity between various words, as it is faster in comparison to other methods.

The path based semantic similarity between word senses is calculated by finding the shortest path between word senses and finding number of nodes along this path. Smaller the number of nodes from a path, the more similar are two words.

In the proposed approach, Semantic similarity score S between words W_1 and W_2 is calculated with the following formula. Here, path_length is the shortest path between two words. This is the number of nodes on the path from W_1 to W_2.

$$S(W_1, W_2) = 1 / \text{path\_length}(W_1, W_2) \quad (3.1)$$

Figure 3.7 shows the semantic similarity score calculation process of *ACRWebS*. WSDL file data and user query data are combined with the *WordNet* based concepts to identify the semantic relationship between the words of user query and web services dataset.

The average similarity score for a given user query and web services data is calculated with the help of following algorithm.

Following notations have been used in this algorithm.

q_i: i^{th} query, $i = 1, 2, \ldots, l$

s_j: j^{th} service, $j = 1, 2, \ldots, m$

c_r: r^{th} category, $r = 1, 2, \ldots, o$

$q_i w_u$: u^{th} word in query q_i, $u = 1, 2, \ldots, p_i$

$s_j w_v$: v^{th} word in service s_j, $v = 1, 2, \ldots, t_j$

Algorithm 1. Similarity score calculations using WordNet

i) Input user query q_i and web service s_j data for a category c_r.

ii) For all words $s_j w_u$ in the web service data, calculate similarity score with each word of user query $q_i w_u$ using (3.1). Similarity score of ≤ 0.2 has not been considered as it represents score of unrelated words.

iii) Take all other web services in category c_r and find the similarity score against the user query q_i.

iv) Calculate average similarity score S_{avg} between web service s_j and user query q_i using formula:

$$S_{avg}(q_i, s_j) = \frac{1}{p_i} \sum_{u=1}^{p_i} \sum_{v=1}^{t_j} S(q_i w_u, s_j w_v) \quad (3.2)$$

v) Sort similarity scores in ascending order and suggest the service with top scores to user to consume in her applications.

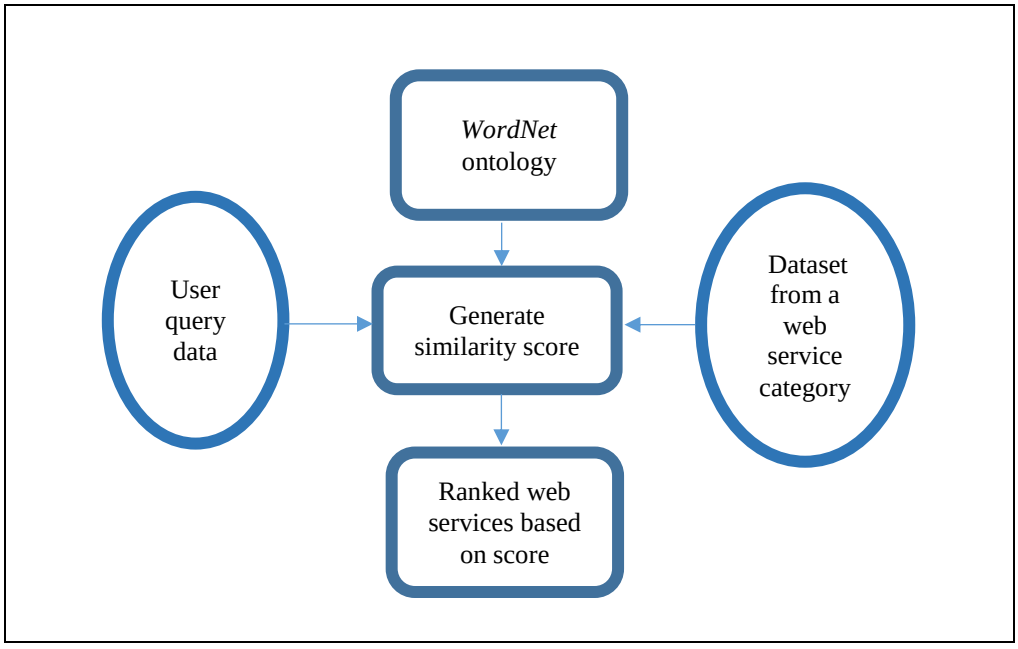


Figure. 3.7: *ACRWebS*: Semantic similarity process

3.4 Sequence diagram of *ACRWebS*

Figure 3.8 depicts the sequence diagram of the *ACRWebS*. A sequence diagram shows interactions among the various objects of a system. This interaction is shown in sequential order and helps developers or users of the system for better understanding of the framework.

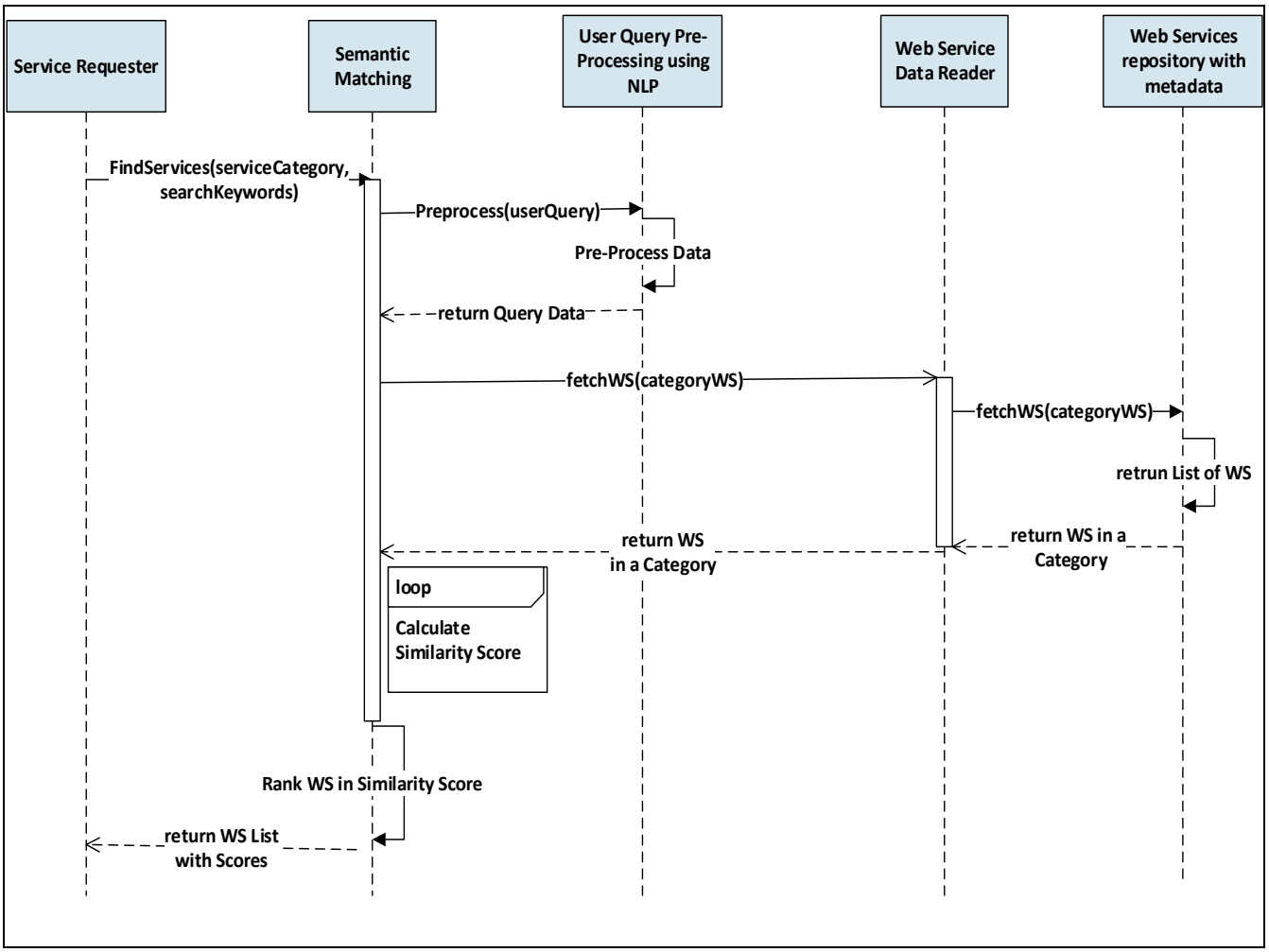


Figure 3.8: ACRWebS: Sequence diagram

3.5 Comparative analysis of *ACRWebS* with other approaches

The proposed *ACRWebS* framework provides many advantages in comparison with other approaches. Table 3.1 provides a comparison of the *ACRWebS* with other approaches, which primarily are syntactic based approaches. It can be observed that *ACRWebS* framework outperforms other approaches on many parameters.

Table 3.1: Comparison of *ACRWebS* with other approaches

S. No.	Technique used	Other syntactic approaches	*ACRWebS* Approach
1	Use of semantic association technique	Majority of the approaches uses syntactic based techniques.	Semantic relatedness is used for finding the service similarity.
2	Service classification	Most of the existing approaches use manual approaches for service categorization, which is an error prone and time consuming activity.	Used automated techniques for classification of web services using the machine learning methods.
3	Need for additional semantics	Some of the current approaches recommend addition or annotation of web services data to improving the discovery process.	*ACRWebS* does not require additional semantics or data for the discovery process.
4	Change in web services WSDL file structure	Some of the approaches recommend changes in existing WSDL structure or standards to ease the discovery problem.	*ACRWebS* does not recommend changes in existing WSDL file structure or standards.

The objective of proposed framework is to make the job of user more efficient while finding required services. The salient features offered by the proposed framework are:

i) This framework provides automatic classification and ranking of web services.
ii) The proposed framework refines the search for efficient retrieval of web services.
iii) Web services specific to the requester context are retrieved.

iv) This framework is simple and does not expect technical or domain knowledge from the users.

v) We have included a new ranking algorithm for user queries. For a given user query q_i, the framework provides a list of target services which are relevant to the user requirements.

vi) The proposed framework does not require any changes in the existing WSDL file structure or standards of web services.

3.6 Chapter summary

One of the issues of the existing web services discovery mechanism is that they demand a number of technical details from users to retrieve the services. However, most of the times, users do not have knowledge about the domain and technical specifications of the area. As such, the services discovery mechanism should be robust enough to carry out web services discovery, even if the user is not having the domain or technical knowledge of the desired web services.

The aim of this chapter is to demonstrate how *ACRWebS* can address the web services discovery problem with a simple, but effective methodology without imposing any further load on consumers or changing existing structure of web services. The proposed framework can support services users in finding their required web services in an easy and efficient way.

ACRWebS is a two phase framework to improve upon the web services discovery mechanism. Services classification can significantly improve the discovery process and increase the framework efficiency and usefulness. The use of semantic technique provided a way for web services ranking. This two-stage framework will be discussed in detail in Chapters 4 and 5, along with experimental results.

Chapter 4

Classification of Web Services using Machine Learning Models

The web services discovery mechanism has continuously evolved during the last few years. There exists techniques and methods for meeting the challenge of improving web services discovery. Automatic classification of web services description into a predefined set of categories, saves effort for computationally expensive search processes. This can significantly enhance the speed and efficiency of finding required web service. In this chapter, we present the classification phase of *ACRWebS*. Web services classification task has been performed by leveraging the techniques used for automatic classification of documents.

This chapter provides detailed analysis of various machine learning techniques in the area of web services discovery. We have performed three experiments, in order to evaluate the best classifier for web services. Experiment-I provides empirical analysis of the classification models without optimizing their configuration.

Experiment-II provides further analysis using optimized configuration of classification models and combination of the NLP technique parameters. A different data set has been used for these experiments. In Experiment-III, we have validated the outcome of Experiment-I and II, in order to check if the recommendations of Experiment-I and II still hold true when Experiment-I is executed with optimized configuration of classification models and with different NLP techniques.

Section 4.1 of the chapter provides details about the Experiment-I. This section presents an empirical analysis of prominent machine learning algorithms without configuration optimization. This includes training data preparation and design of experimental work. The classification models have been evaluated using the cross-validation technique. Section 4.2 describes another set of experimentation using optimized configuration of classifiers alongwith configuration of various NLP parameters. A different data set has been used in order to evaluate the performance of classification models from the existing work of researchers. Section 4.3 presents the experimentation using the optimized configuration of classifiers on the dataset of Experiment-I. Section 4.4 concludes the chapter.

RapidMiner software (version 5.3008) has been used as a tool for experimentation. This tool provides an environment for machine learning and data mining processes. We have used its text processing plug-in for pre-processing of WSDL files for the data mining tasks including tokenisation, stemming, and removing stop words.

4.1 Experiment-I: Web services classification without parameter optimization of classification models

In this experimentation, various empirical tests have been performed to compare classifier's performance when applied on the web services data. Classification algorithms, namely, NB, *k*-NN, ANN, SVM and RF have been considered in this work. The performance of these classifiers has been compared using *k*-fold cross-validation technique and by varying the size of training data.

4.1.1 Data preparation

Data for experimental work has been collected from the websites, namely, *www.xmethods.net, www.service-repositoy.com* and *www.SALCentral.com*, containing functionally active web services. A dataset of 200 web services was created under the

categories, namely, country, communication, finance, business, news and currency. We have captured keywords from WSDL file tags related to operations and arguments as these tags semantically represent the information about web services. This data was manually categorized based on the type and information of the web services for the training purpose.

NLP techniques have further been used for pre-processing of the extracted data and to reduce data space and increase the effectiveness of classifiers.

4.1.1.1 An example dataset

This section provides an example of processing a WSDL file to prepare the training dataset. A WSDL file's operations names and attributes are extracted and splitted using CamelCase and Underscores criteria of RapidMiner. For the classification purpose, we have used the common feature extractor method called bag-of-words (BoW). A term weighing approach called Term Frequency-Inverse Document Frequency (TF-IDF) has been used for calculating importance of a word in a data set. As an example, let us consider the following piece of code from a WSDL file (Figure 4.1).

```
<message name="FindRouteSoapOut">
<part name="parameters" element="s0:FindRouteResponse" />
</message>
<message name="FindRouteAsDatasetSoapIn">
<part name="parameters" element="s0:FindRouteAsDataset" />
</message>
```

Figure 4.1: An example WSDL file

In this WSDL file, XML identifier *FindRoute* is splitted into tokens *Find* and *Route*. Accordingly, the complete BoW for this example is {*Find*: 4, *Route*: 4, *Soap*: 2, *Out*: 1, *In*:

1, *As*: 1, *Dataset*: 1 *Response*: 1}.

4.1.2 Experiment for web services classification

This section explains the process undertaken for web services classification. The performance of a classifier depends on various parameters including the characteristics of the data to be classified. We have used web services data, namely, operation and argument attributes of web services for training and validation of the proposed approach. The performance of classifiers is evaluated through cross-validation technique, which involves determination of classification accuracy for multiple partitions of the input data used in training. In cross-validation technique with *k*-folds, training and testing are performed *k* times. The dataset is randomly partitioned into *k* mutually exclusive subsets or folds D_1, D_2, …, D_k, each with same size. For iteration *i*, partition D_i is reserved as the test data and the remaining partitions are used to train the classifier. Figure 4.2 shows various steps considered in Experiment-I.

Experiments have been conducted by considering three different values of *n* (number of services) as 100, 150 and 200. Table 4.1 contains the distribution of these web services in different categories. This data is used to train the classifiers. Also, the experiments have been performed with different values of *k* (= 3, 5, 7 and 10) in *k*-fold cross-validation to observe the effect of number of folds on the performance of a classification model.

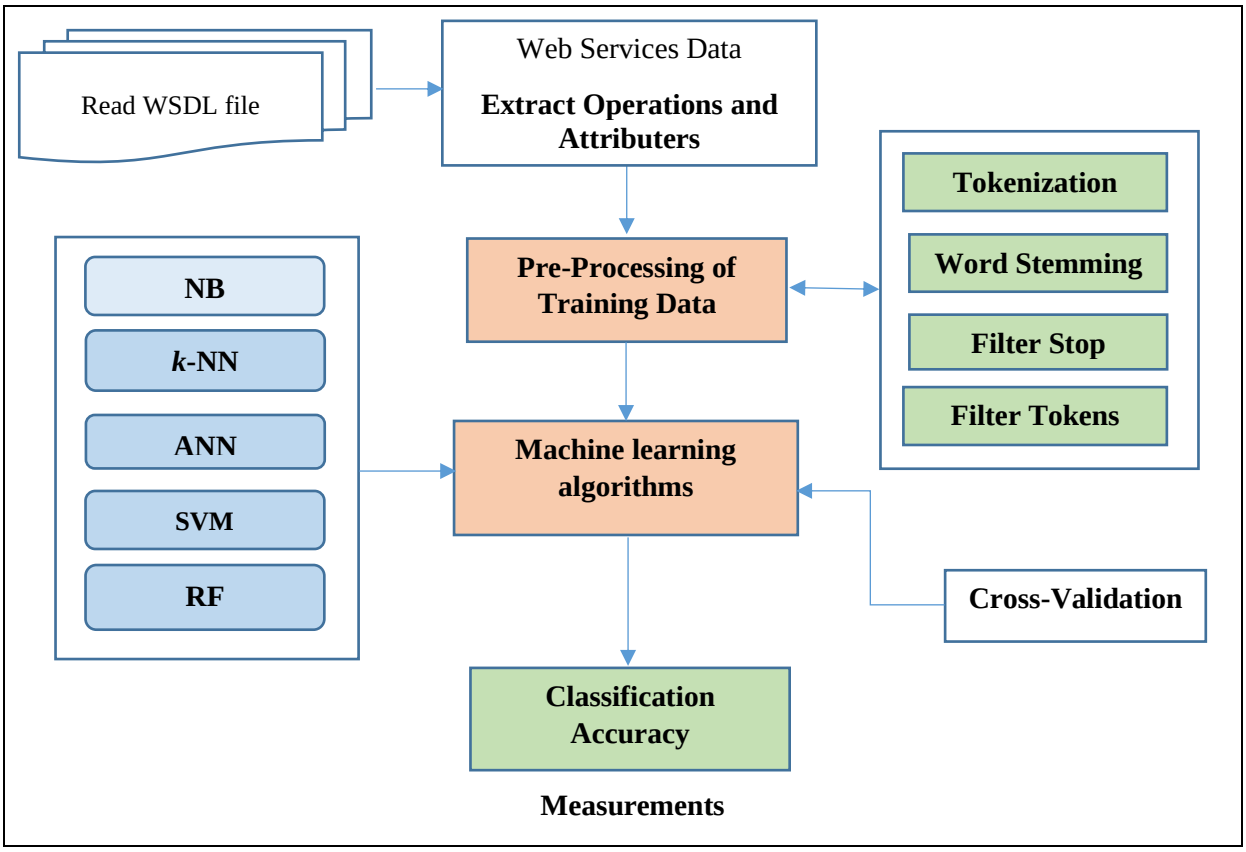


Figure 4.2: Machine learning model in web services classification

Table 4.1: Distribution of web services

n	Number of services in different categories					
	Country	Comm	Finance	Business	News	Currency
100	44	26	7	10	5	8
150	50	38	20	13	16	13
200	65	50	27	17	22	19

Communication - Comm

4.1.3 Evaluation of classification models

This section presents experimental results for evaluation of classification models when applied to the web services classification problem. The performance of classification models have been evaluated using the criteria of accuracy, precision and recall.

Table 4.2 contains the results of 3-fold cross-validation for n = 100, 150 and 200. It can be observed from this table that SVM classifier achieves the highest accuracy of 78.7% when n is 100. When the value of n is increased to 200, k-NN dominates other classifiers. The maximum precision of 86.2% is achieved by RF when n is 200. Maximum recall of 94.8% is achieved by SVM when n is 100. On an average, with k = 3, the accuracy achieved by k-NN is 72.8%, by ANN is 52.9%, by NB is 75.3%, by SVM is 78.7% and by RF is 65.6%. It has been observed that on an average, highest accuracy and recall are achieved by SVM.

Table 4.3 contains the results for 5-fold cross-validation. SVM achieves the highest accuracy of 82.7% and highest precision of 82.7% when n is 150. RF achieves the highest recall of 93.2% when n is 100. It has also been observed that on an average, the highest accuracy of 79.3%, highest recall of 68.1% and highest precision of 79.9% has been achieved by SVM.

Table 4.4 shows the results for 7-fold cross-validation. SVM outperforms all other classification models here and achieves the highest accuracy of 82.7% when n is 150. SVM has attained highest precision of 86.5% when n = 100 and highest recall of 76.7% when n = 200. On an average, SVM achieved highest accuracy of 81.1%, highest recall of 70.5% and highest precision of 85.5%.

Table 4.5 shows experimental results for 10-fold cross-validation. It has been noticed that SVM achieves the highest accuracy of 82.7% when n = 200. Highest precision of 82.3% and highest recall of 92.8% has also been achieved by SVM. On an average, the highest accuracy of 80.9% and maximum recall of 79.9% has been achieved by SVM classification model. Highest average precision of 79.9% has been achieved by RF.

Table 4.2: Comparison of classifiers with 3-fold cross-validation with different data sizes

	***n* = 100**			***n* = 150**			***n* = 200**		
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
NB	74.2	71.8	3.19	75.2	68.7	66.6	76.7	69.6	67.7
***k*-NN**	72.3	68.4	62.7	69.1	62.0	62.3	77.2	70.5	68.9
ANN	65.3	52.4	52.7	56.9	35.1	31.8	36.7	32.4	25.5
SVM	78.7	60.7	94.8	76.3	81.3	59.2	76.7	78.4	72.3
RF	68.0	63.4	37.2	76.1	80.7	62.1	52.6	86.2	34.4

Table 4.3: Comparison of classifiers with 5-fold cross-validation with different data sizes

	***n* = 100**			***n* = 150**			***n* = 200**		
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
NB	73.3	62.9	64.1	73.1	65.7	63.9	78.2	3.5	70.5
***k*-NN**	71.3	57.3	60.4	75.8	70.5	68.3	78.2	72.2	70.5
ANN	48.0	40.6	27.2	45.9	24.2	24.7	48.0	35.8	26.9
SVM	77.3	77.2	59.6	82.7	82.7	70.7	78.0	80.0	74.0
RF	69.3	38.3	93.2	72.4	82.0	55.0	65.3	84.5	53.0

Table 4.4: Comparison of classifiers with 7-fold cross-validation with different data sizes

	***n* = 100**			***n* = 150**			***n* = 200**		
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
NB	74.2	69.0	64.6	75.8	71.3	67.4	78.2	71.1	69.9
***k*-NN**	74.3	69.4	64.6	74.5	60.9	63.6	81.2	74.5	72.6
ANN	57.3	43.7	29.8	57.8	33.0	34.2	35.3	18.6	23.2
SVM	80.0	86.5	64.6	82.6	82.7	70.6	80.7	81.4	76.7
RF	69.3	73.4	38.9	50.4	20.3	10.1	65.3	80.6	54.1

Table 4.5: Comparison of classifiers with 10-fold cross-validation with different data sizes

	***n* = 100**			***n* = 150**			***n* = 200**		
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
NB	73.2	66.4	64.1	76.5	71.6	68.5	78.2	73.4	69.8
***k*-NN**	73.2	68.4	64.1	77.1	69.4	68.3	78.2	71.3	69.9
ANN	61.3	52.4	25.6	51.4	17.4	21.3	42.7	13.9	21.3
SVM	82.7	68.3	92.8	79.9	76.6	70.2	80.2	82.3	76.7
RF	70.6	73.4	39.2	76.1	80.7	62.1	67.3	80.1	56.1

Figures 4.3-4.5 summarize the results received for accuracy, precision and recall by applying 3-, 5-, 7-, and 10-fold cross-validation on different web services dataset.

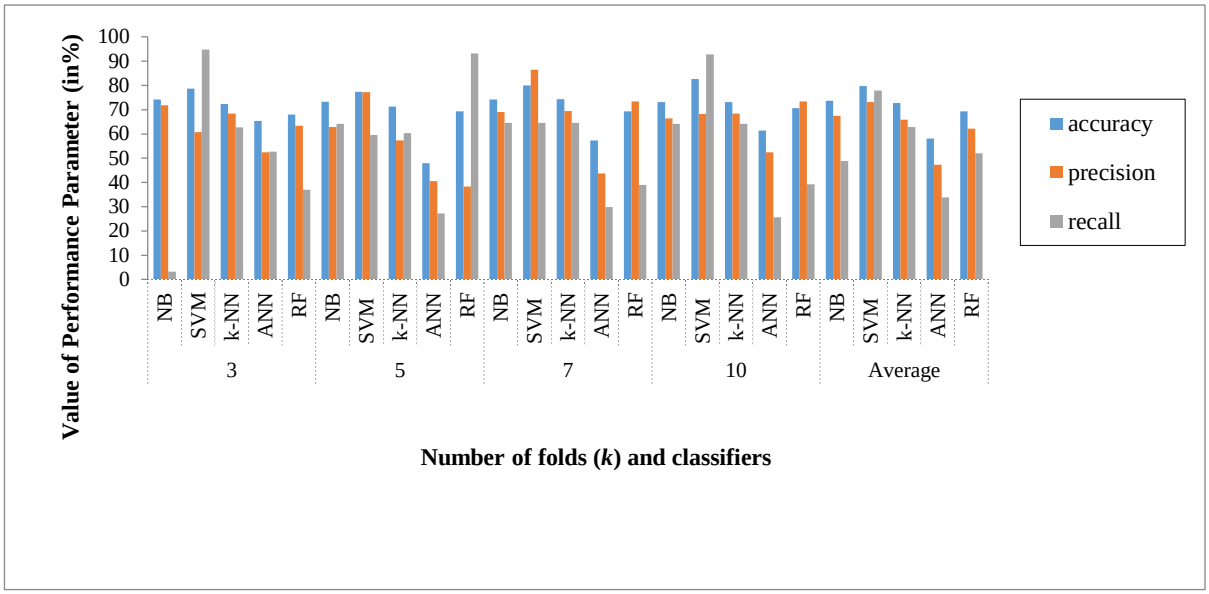


Figure 4.3: Effect of number of folds on the performance of classifiers with *n* = 100

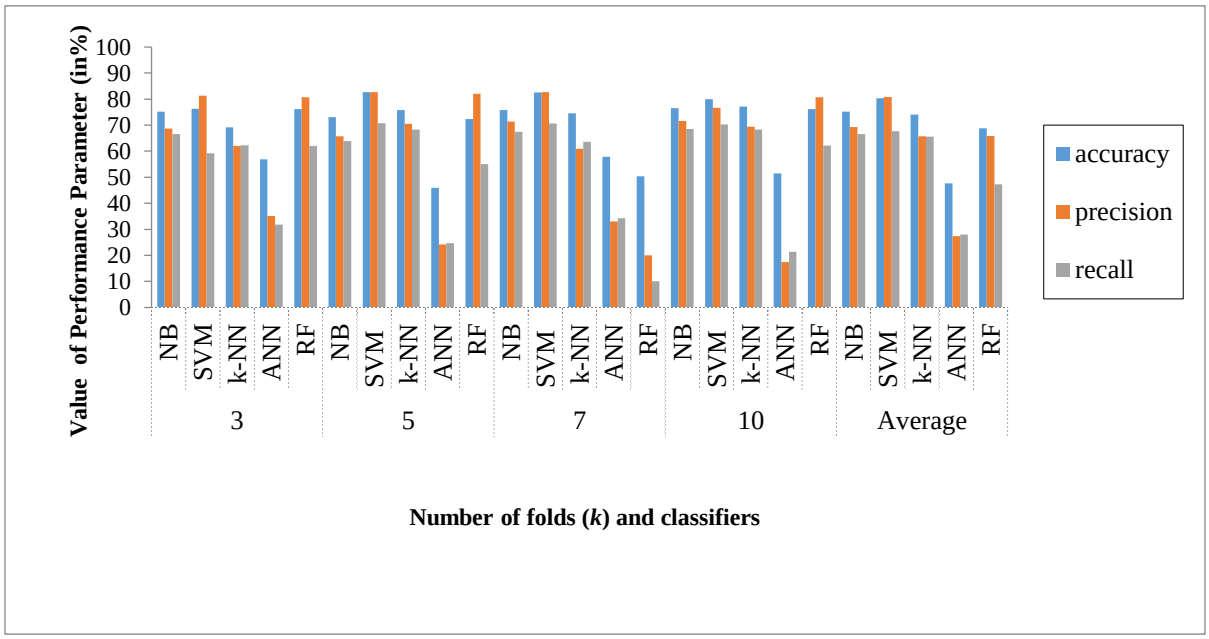


Figure 4.4: Effect of number of folds on the performance of classifiers with *n* = 150

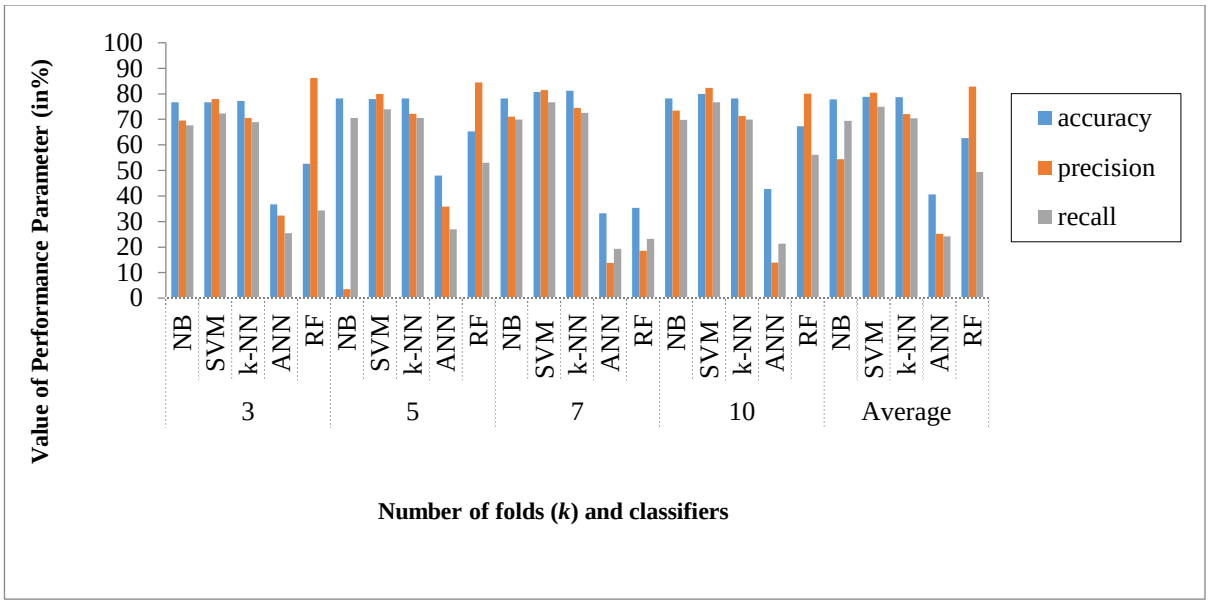


Figure 4.5: Effect of number of folds on the performance of classifiers with *n* = 200

The results obtained from Experiment-I suggest that SVM model generally provides a better performance than other classification models included in this work. As such, this can be considered as a potential classifier for the web service classification problem. As mentioned earlier, these results are based on the experimentation when the parameters of classification models have not been optimized.

4.2 Experiment II: Web services classification with parameter optimization of classification models

Section 4.1 presented the classification results without optimum configuration of classification models. It is important to refer the optimised parameter settings for the classification models. This section provides a study on the selected machine learning models with their optimized parameters to further improve the classification results and take best out of a classifier. Also, the different combination of NLP techniques has been used for pre-processing of training data, which have immense impact on the performance of a classification model.

4.2.1 Framework evaluation

This section describes the steps used for the proposed process for evaluating the classification models. A feature vector, created by extracting the functional descriptions, such as the input and output information, and arguments from the WSDL files of web services are used for the training of classification data. These extracted features are pre-processed using NLP techniques to create a clean training data. These training data vectors are fed to the machine learning model for the classification task.

Figure 4.6 contains the proposed process for web services classification. Cross-validation technique has been used to evaluate the performance of machine learning classifiers in the work. Subsequent sections elaborate the steps used in the proposed approach.

Data Pre-processing*:* In order to reduce data space and increase the performance of the proposed algorithms, we have carried out pre-processing of data extracted from web services using NLP based techniques.

N-gram scheme: N-gram is extensively used in text processing tasks. This operator creates term *n*-gram of tokens in a document. In this work, three types of *n*-gram are considered: *Unigram*, *Bigram* and *Trigram*.

We have evaluated the performance of the classification model on the training dataset to get an unbiased measure of the model performance. The performance of proposed approach has been evaluated using criteria of accuracy, precision and recall. These criteria have mostly been used to evaluate machine learning techniques. Samples of size 132, 264 and 397 have been created, in order to evaluate the effect of size of dataset on a classification model. Significance tests have been used for evaluating the results statistically.

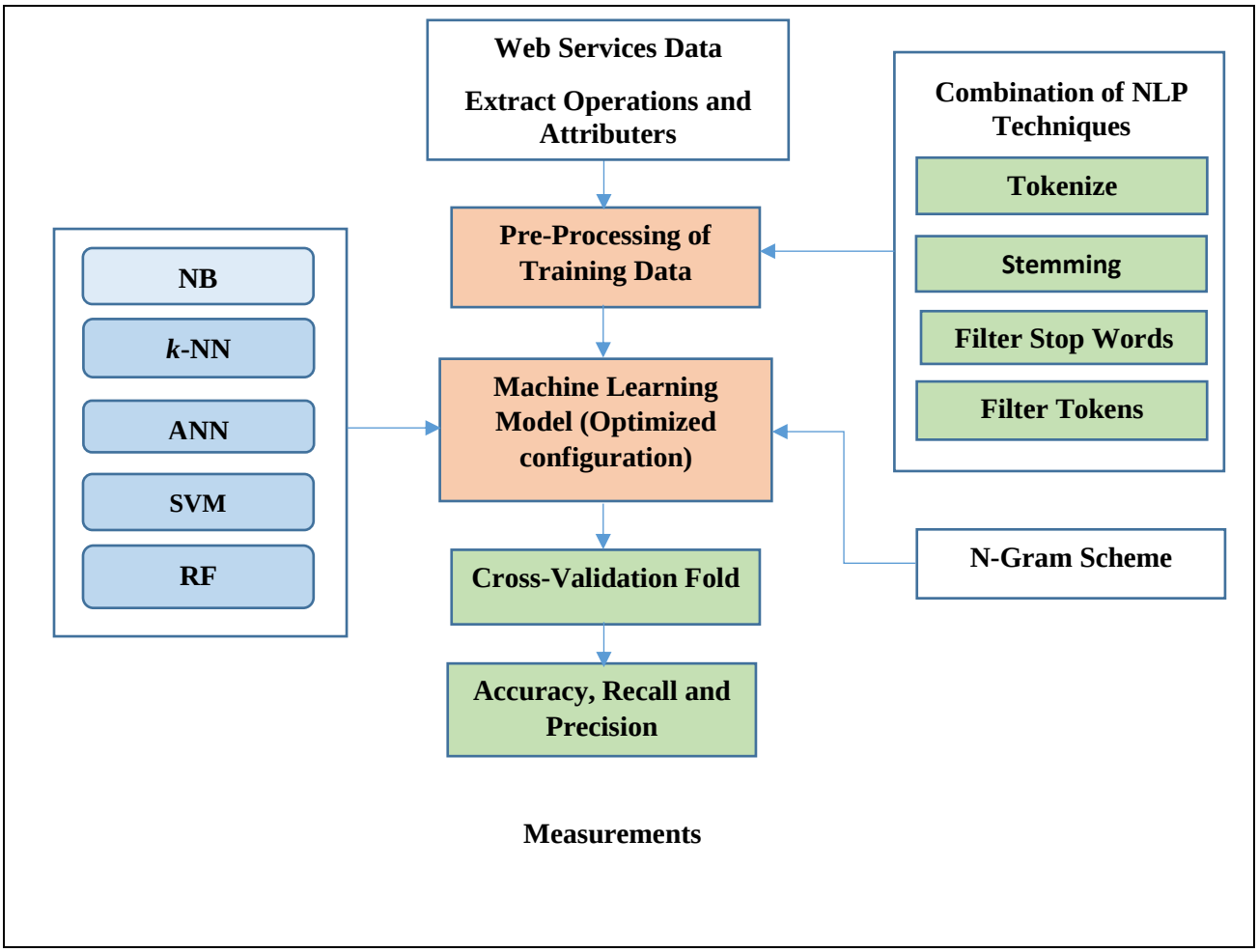


Figure 4.6: Web services classification process

4.2.2 Training dataset

The dataset for experimentation work has been collected from the WSDL files of repository www.andreashess.info/projects/annotator/ws2003.html. This dataset is publically available and has been created by Bennaceur *et al.* (2011). This training dataset has been chosen for experimentation, so that we can take their results as a benchmark for performance comparison purpose. This reference data is a collection of 397 WSDL files and is pre-categorised in 10 categories based on the type and information contained in the files. Table 4.6 shows the collected dataset for the training purpose and each of the dataset is described by its category and number of service instances in a particular category.

The background for using web service features such as name of service, input/output arguments and operations name is that these features usually represent semantically the capabilities or offered functionality of a particular web service.

Table 4.6: Web services data for experimentation

Service Category	Service Instances
cnt	64
money	54
conv	49
finder	46
comm	45
web	39
dev	37
news	30
busi	23
math	10

Communication - comm
CountryInfo - cnt
Converters - conv
Developer - dev
Business - busi
Mathematics- math

4.2.3 Experimental results

This section presents result of several experimental observations. We have carried multiple experiments with various combination of parameters, including NLP based pre-processing technique, *n*-gram scheme and varying dataset size.

We have used the optimal configuration for each of the used classifier as mentioned in below sections. These configurations have been evaluated after multiple experiments with different configuration parameters of a classification algorithm. The experiment for *k*-NN based classification model have been performed with $k = 5$, as it provided the best results. For the NB classifier, the Laplace Correction has been selected. Both of these classifiers are easy

to configure, as they have only one parameter to be set.

The configuration chosen for ANN classifier is shown in Table 4.7. The configurations used for the SVM and RF models are shown in Tables 4.8 and 4.9 respectively.

Table 4.7: Configuration for ANN

Parameter	Values
Tracing cycles	100
Learning rate	0.3
Momentum rate	0.2
Hidden layers	5

Table 4.8: Configuration for SVM

Parameter	Values
Kernel type	Linear
Gamma	0
Max iterations	100000
Cache size	80
Epsilon	0.001
SVM type	C-SVM

Table 4.9: Configuration for RF

Parameter	Values
Number of trees	50
Number of features	0
Seed for random number generator	1
Maximum depth	20

Several performance evaluation measures have been taken into account to quantify and compare the selected classification algorithms. Experiments are conducted with various NLP based pre-processing techniques with their different combinations.

Following combination of different parameters of NLP techniques have been considered for the experimentation purpose.

- Tokenisation + Filter Token
- Tokenisation + Filter token + Filter stop words + Stemming + Unigram
- Tokenisation + Filter token + Filter stop words + Stemming + Bigram
- Tokenisation + Filter token + Filter stop words + Stemming + Trigram

All the experimental results have been observed using a 10-fold cross-validation process. Training data has been distributed in equal proportion among all the categories for best results and to observe if there is substantial difference of performance between the larger and the smaller dataset. Since, most of the classification problems encounter large data size, we have taken the larger dataset for creating the confusion matrix.

Table 4.10 shows experimental results of NB classifier. NB achieves maximum accuracy of 69.3%, maximum precision of 66.0% and maximum recall of 64.9% respectively, when the used operators are *tokenisation*, *stemming*, *stop word* and *unigram* with a *dataset* size of 379. Table 4.11 shows the confusion matrix of NB classifier. It has been observed that 110 instances out of the 397 are misclassified and the *business* class seems a bit harder to predict in comparison to other classes.

Table 4.12 shows experimental results of *k*-NN classifier with different data size (*n*) and data processing schemes. It has been noticed that *k*-NN achieves maximum accuracy of 69.9%, maximum precision of 69.5% and maximum recall of 63.7% respectively, when the used NLP operators are *tokenisation*, *stemming*, *stop word* and *unigram* with a dataset size of 397. Table 4.13 shows the confusion matrix for *k*-NN. It can be seen that 178 instances out of the 397 are misclassified by this model. It is also seen that *developers*, *web* and *mathematics* classes are a bit harder to predict by the *k*-NN classifier.

Similarly, the ANN classifier achieves maximum accuracy of 68.3%, maximum recall of 65.5% and maximum precision of 65.6% respectively, when used different operators as depicted in Table 4.14. Table 4.15 shows confusion matrix for ANN and it misclassifies 109 instances out of 397 instances. It has been observed that *mathematics* class is most hard to predict for ANN classifier.

Table 4.16 shows experimental results of SVM classifier. It has been noted that SVM achieves maximum accuracy of 73.6%, maximum precision of 66.8% and maximum recall of 67.4% respectively, when used operators are *tokenisation, stemming, stop word* and *unigram* with a dataset size of 379. Table 4.17 shows the confusion matrix of SVM classifier and it can be seen that only 90 instances out of the 397 are misclassified. It is also noted that *business* class is a bit harder to predict for SVM classifier.

Similarly, Table 4.18 shows the experimental results for the RF classifier. The result shows that it achieves maximum accuracy of 71.3%, maximum precision of 72.0% and maximum recall of 67.4% respectively with the combination of various operators and optimum configuration. Table 4.19 shows confusion matrix for RF classifier and it has been observed that *business* class is a bit harder to predict for RF classifier model.

If compared our work with the similar work by Bennaceur *et al.* (2011), we can conclude that our experimental results are promising. They experimented with various variant of the SVM machine learning classifier such as boinarised L2-SVM, binarised L1-SVM, and perception *etc.* They achieved a maximum recall of 52.9% and precision of 58.0% in their various experiments. It can be noted that the results of our proposed approach has significantly higher accuracy, precision and recall values. Overall, all the classification models provided good results with their optimised parameters. But, SVM outperforms all classification algorithms, followed by ANN and RF. *k*-NN and NB are easy to use, however, this advantage is outweighed by their classification results, which are comparatively poor.

This has been observed that various NLP techniques have immense impact on the performance of classification models. The various experiments depict that consideration of some of the NLP techniques such as tokenisation or filtering is not sufficient for getting the best results of a classifier. Stop filter and stemming have also significant effect on a classifier performance and should be part of the processing technique.

Table 4.10: Results using NB classifier

Operations combination	*n* = 132			*n* = 264			*n* = 397		
	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)
NB + Tokenization + Filter Tokens	54.9	51.9	55.7	52.7	52.7	57.6	62.6	63.0	66.5
NB + Tokenization + Filter Tokens + Stemming + Stop words + Unigram	54.2	50.9	54.1	53.4	51.2	55.9	66.0	64.9	69.3
NB + Tokenization + Filter Tokens + Stemming + Stop words + Bigram	57.5	54.7	57.3	54.6	52.1	56.8	64.7	64.4	69.2
NB + Tokenization + Filter Tokens + Stemming + Stop words + Trigram	57.6	54.7	57.4	55.4	52.9	57.6	64.6	64.4	69.2

Table 4.11: Confusion matrix for NB

	True cnt	True comm	True news	True conv	True money	True finder	True web	True dev	True math	True busi	Class precision (%)
pred. cnt	50	0	1	4	1	2	2	0	0	1	81.9
pred. comm	4	41	1	1	1	1	1	1	0	1	78.8
pred. news	0	0	19	0	0	2	5	0	1	4	61.3
pred. conv	0	0	0	32	0	1	0	1	1	2	86.5
pred. money	5	0	1	4	52	0	7	1	3	2	69.3
pred. finder	1	1	1	2	0	27	2	2	0	1	72.9
pred. web	2	2	0	1	0	3	18	4	0	2	56.2
pred. dev	1	1	2	2	0	7	2	24	0	3	57.1
pred. math	1	0	2	1	0	2	0	0	5	0	45.5
pred. bus	0	0	3	2	0	1	2	4	0	7	36.8
class recall (%)	78.1	91.1	63.3	65.3	96.3	58.7	46.1	64.8	50.0	30.4	

Table 4.12: Results using *k*-NN classifier

Operations combination	*n* = 132			*n* = 264			*n* = 397		
	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)
***k*-NN + Tokenization + Filter Tokens**	66.8	54.9	62.3	61.1	54.7	61.1	56.4	47.8	55.9
***k*-NN + Tokenization + Filter Tokens + Stop words + Stemming + Unigram**	68.9	61.3	69.9	69.5	62.2	69.0	66.4	46.4	53.9
***k*-NN + Tokenization + Filter Tokens + Stemming + Stop words + Bigram**	68.4	60.0	68.0	68.4	63.6	68.1	64.9	42.9	49.9
***k*-NN + Tokenization + Filter Tokens + Stemming + Stop words + Trigram**	66.5	58.8	66.4	67.6	63.7	67.2	64.7	42.3	48.3

Table 4.13: Confusion matrix for *k*-NN

	True cnt	True comm	True news	True conv	True money	True finder	True web	True dev	True math	True busi
pred. cnt	52	4	2	4	5	7	6	2	1	1
pred. comm	0	21	1	0	0	0	0	1	0	2
pred. news	4	1	22	1	0	0	1	2	1	4
pred. conv	0	0	0	23	0	1	0	0	1	0
pred. money	0	0	1	2	40	0	0	0	0	1
pred. finder	7	17	3	19	9	32	19	22	6	14
pred. web	1	0	1	0	0	5	13	0	0	1
pred. dev	0	0	0	0	0	1	0	10	0	0
pred. math	0	0	0	0	0	0	0	0	1	0
pred. busi	0	2	0	0	0	0	0	0	0	0
class recall(%)	81.2	46.6	73.3	46.9	74.0	69.5	33.3	27.0	10.0	0.0

Table 4.14: Results using ANN classifier

Operations combination	*n* = 132			*n* = 264			*n* = 397		
	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)
ANN + Tokenization + Filter Tokens	33.1	25.9	32.7	49.4	49.2	62.0	61.2	60.6	69.7
ANN + Tokenization + Filter Tokens + Stemming + Stop words + Unigram	38.5	35.3	45.9	56.0	54.7	68.8	65.5	63.8	72.5
ANN + Tokenization + Filter Tokens + Stemming + Stop words + Bigram	28.4	28.5	36.1	54.6	56.6	69.0	64.7	65.5	74.8
ANN + Tokenization + Filter Tokens + Stemming + Stop words + Trigram	16.6	18.9	23.3	51.6	50.2	62.9	65.9	63.8	71.7

Table 4.15: Confusion matrix for ANN

	True math	True comm	True cnt	True news	True dev	True conv	True money	True finder	True web	True busi	Class precision (%)
pred. math	0	0	0	0	0	0	0	0	0	0	0.0
pred. comm	0	39	0	1	1	1	0	1	0	2	86.6
pred. cnt	0	0	51	0	2	0	1	1	2	3	85.0
pred. news	1	0	0	25	0	0	0	0	0	0	96.1
pred. dev	2	0	1	0	25	0	0	2	0	1	80.7
pred. conv	2	1	1	0	1	41	0	0	0	0	89.1
pred. money	1	0	0	0	0	1	48	1	0	1	92.3
pred. finder	3	3	7	2	4	5	5	26	7	9	36.6
pred. web	0	2	3	2	4	1	0	11	29	3	52.7
pred. busi	1	0	1	0	0	0	0	4	1	4	36.4
class recall (%)	0.00	86.6	79.7	83.3	67.6	83.6	88.9	56.5	74.3	17.4	

Table 4.16: Results using SVM classifier

Operations combination	*n* = 132			*n* = 264			*n* = 397		
	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)
SVM + Tokenization + Filter Tokens	65.3	49.8	59.8	63.1	57.5	68.1	73.1	66.5	73.3
SVM + Tokenization + Filter Tokens + Stemming + Stop words + Unigram	64.4	54.6	66.4	74.8	66.6	73.3	78.6	73.4	78.1
SVM + Tokenization + Filter Tokens + Stemming + Stop words + Bigram	66.3	49.2	60.6	69.7	64.6	72.0	78.1	72.9	78.3
SVM + Tokenization + Filter Tokens + Stemming + Stop words + Trigram	69.5	45.3	56.5	69.9	64.7	72.1	76.6	71.5	77.3

Table 4.17: Confusion matrix for SVM

	True cnt	True comm	True news	True conv	True money	True finder	True web	True dev	True math	True busi	Class precision (%)
pred. cnt	58	2	3	1	2	1	5	1	0	4	75.3
pred. comm	0	38	0	1	0	0	2	0	0	3	86.3
pred. news	1	0	21	0	0	0	0	1	0	0	91.3
pred. conv	1	0	0	41	2	1	0	0	2	1	85.4
pred. money	0	0	1	1	50	1	0	0	1	0	92.6
pred. finder	1	1	4	3	0	40	7	6	0	7	57.9
pred. web	2	1	1	0	0	1	24	3	0	3	68.5
pred. dev	1	1	0	0	0	1	0	25	0	1	86.2
pred. math	0	0	0	1	0	0	0	0	6	0	85.7
pred. busi	0	2	0	1	0	1	1	1	1	4	36.3
class recall	90.6	84.4	70.1	83.7	92.6	86.9	61.5	67.5	60.0	17.4	

Table 4.18: Results using RF classifier

Operations combination	*n* = 132			*n* = 264			*n* = 397		
	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)	Precision (%)	Recall (%)	Accuracy (%)
RF + Tokenization + Filter Tokens	55.8	49.7	51.3	66.5	63.6	64.3	68.0	65.8	69.9
RF + Tokenization + Filter Tokens + Stemming + Stop words + Unigram	59.9	53.1	59.3	69.8	68.0	69.0	67.9	65.8	69.2
RF + Tokenization + Filter Tokens + Stemming + Stop words + Bigram	58.3	51.3	54.8	65.6	64.6	66.7	72.0	67.4	71.0
RF + Tokenization + Filter Tokens + Stemming + Stop words + Trigram	52.5	51.5	53.1	65.6	65.2	65.2	66.8	67.4	71.3

Table 4.19: Confusion matrix for RFs

	True cnt	True comm	True news	True conv	True money	True finder	True web	True dev	True math	True busi	Class precision (%)
pred. cnt	58	2	3	1	2	1	5	1	0	4	75.3
pred. comm	0	38	0	1	0	0	2	0	0	3	86.3
pred. news	1	0	21	0	0	0	0	1	0	0	91.3
pred. conv	1	0	0	41	2	1	0	0	2	1	85.4
pred. money	0	0	1	1	50	1	0	0	1	0	92.6
pred. finder	1	1	4	3	0	40	7	6	0	7	57.9
pred. web	2	1	1	0	0	1	24	3	0	3	68.5
pred. dev	1	1	0	0	0	1	0	25	0	1	86.2
pred. math	0	0	0	1	0	0	0	0	6	0	85.7
pred. busi	0	2	0	1	0	1	1	1	1	4	36.3
class recall (%)	90.6	84.4	70.0	83.6	92.6	86.9	61.5	67.5	60.0	17.4	

It has also been observed that result of classification algorithms was influenced with the *n*-gram scheme. Analysing the results, we can conclude that for the SVM and *k*-NN classifiers, the unigram scheme outperforms the bigram and trigram schemes. The plot of precision, recall and accuracy for the various classification models is shown in Figures 4.7-4.11.

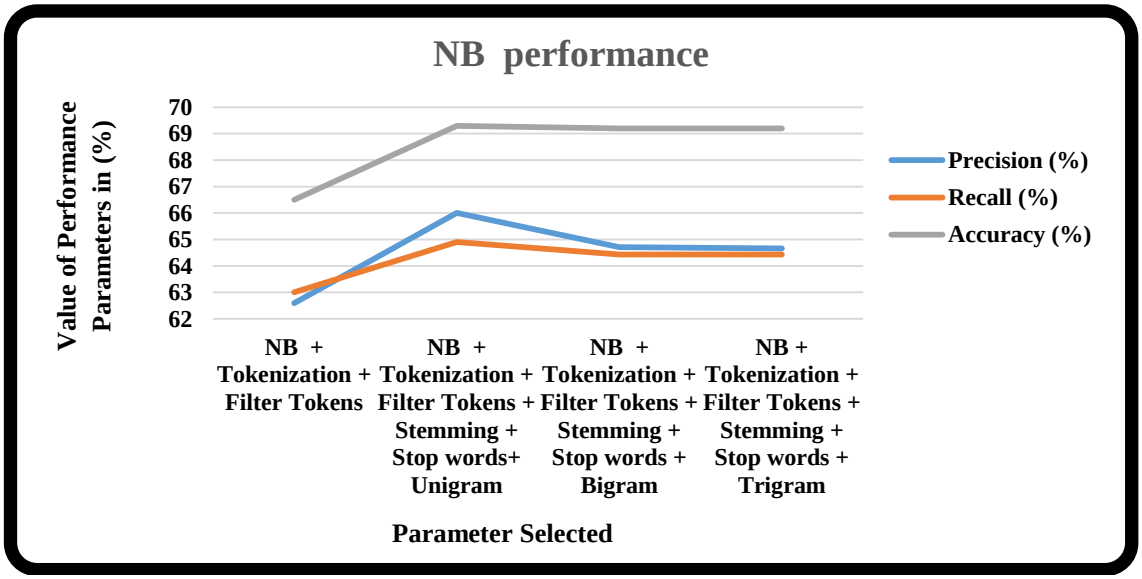


Figure 4.7: Effect of NLP processing steps on NB classifier with *n* = 397

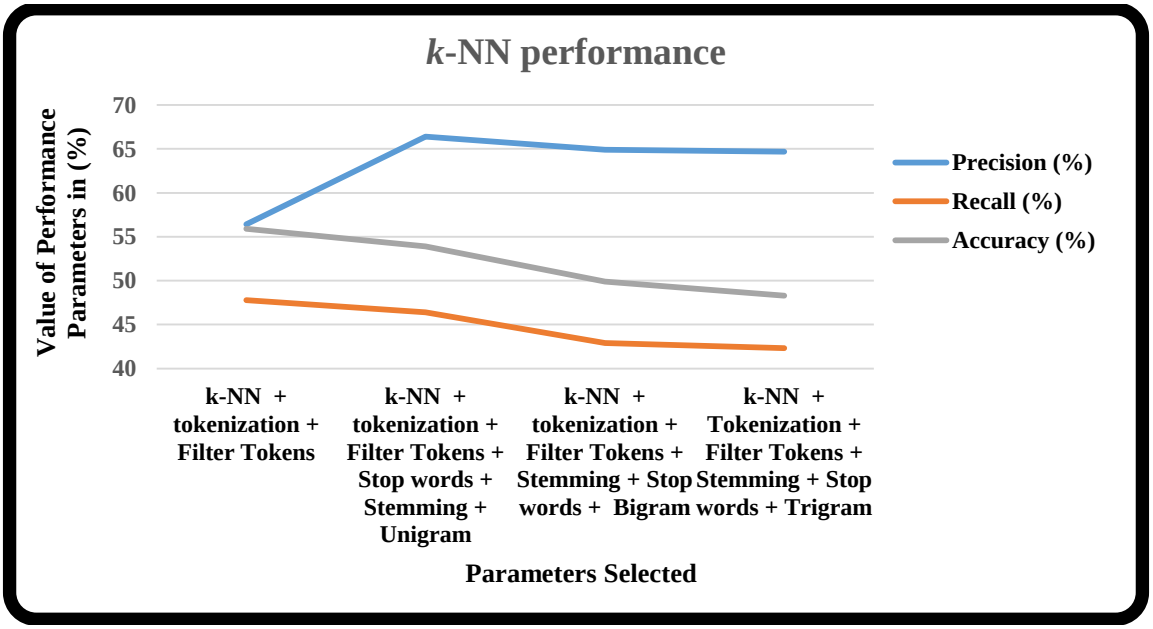


Figure 4.8: Effect of NLP processing steps on *k*-NN classifier with *n* = 397

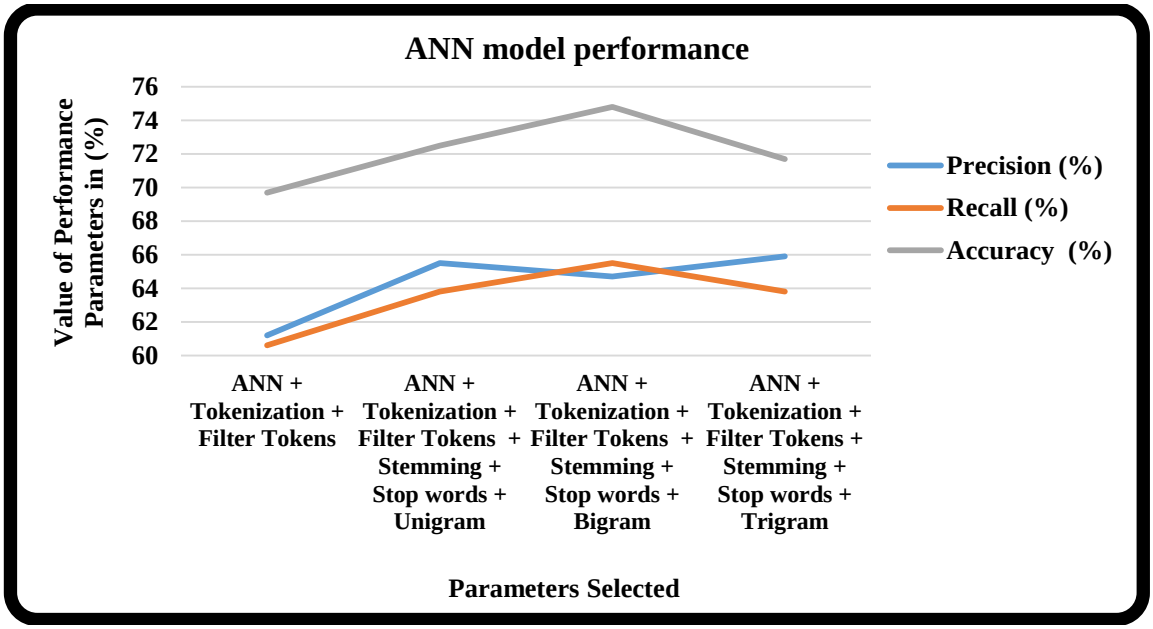


Figure 4.9: Effect of NLP processing steps on ANN classifier with *n* = 397

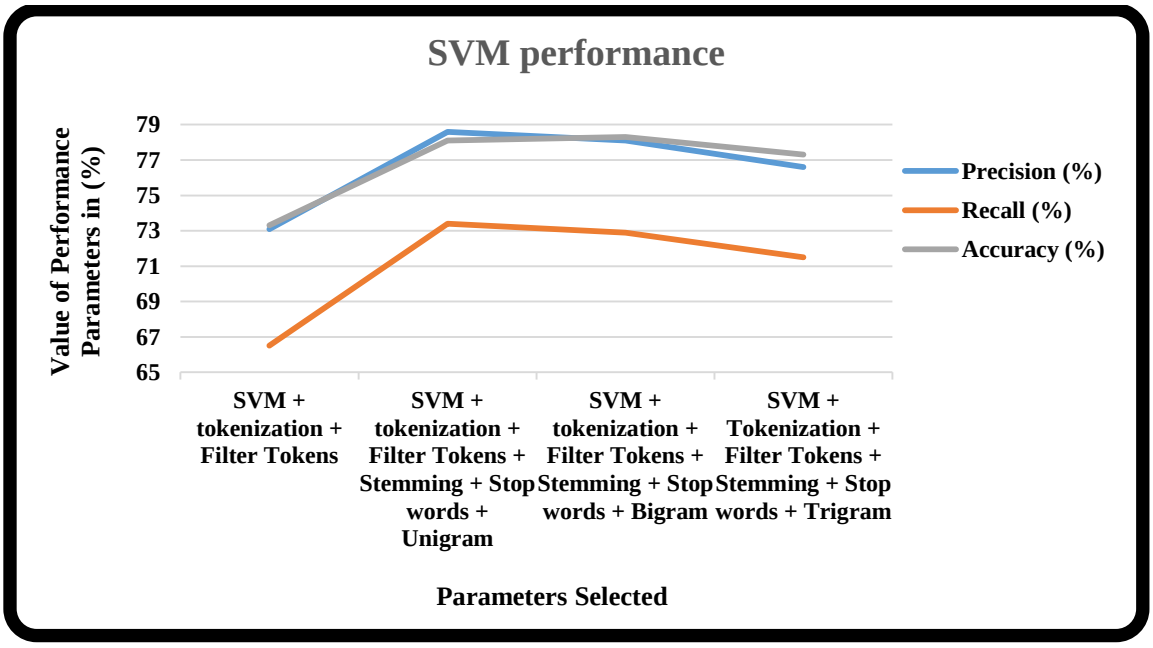


Figure 4.10: Effect of NLP processing steps on SVM classifier with *n* = 397

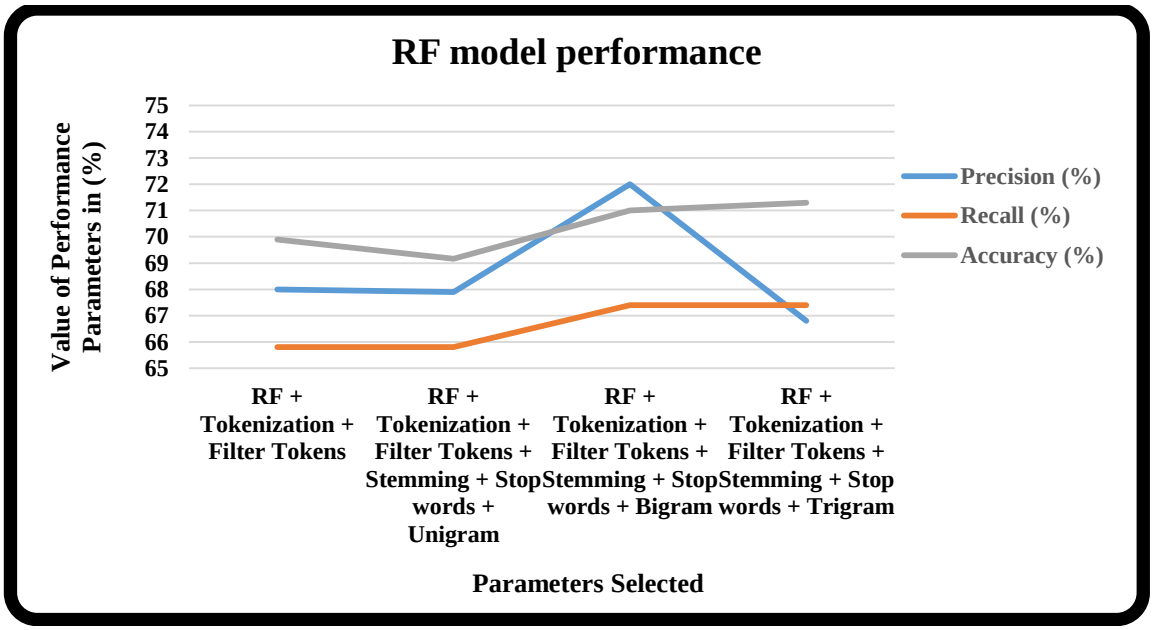


Figure 4.11: Effect of NLP processing steps on RF classifier with *n* = 397

The bigram scheme has been more successful, when used with ANN classifier. RF and NB are also benefitted with the bigram and trigram schemes and provided the improved results. It can be observed that classification results are influenced by the dataset size and most of the classifiers provided improved results with the large dataset size.

We have also performed Analysis of Variance (ANoVA) test in order to statistically establish the difference between the classifiers' performances. Table 4.20 contains the outcome of this test in the form of *p*-values, assuming that all the classifiers (NB, *k*-NN, ANN, SVM and RF) have the same averages. Accordingly, the null hypothesis H_0 is defined as:

$$H_0: \mu_{kNN} = \mu_{NB} = \mu_{SVM} = \mu_{RF} = \mu_{ANN}$$

The ANoVA suggests that mean values of all the classifiers are not equal and we have next considered the following null hypothesis.

$$_1H_0: \mu_{k\text{-}NN} = \mu_{NB},\ _2H_0: \mu_{k\text{-}NN} = \mu_{SVM},\ _3H_0: \mu_{k\text{-}NN} = \mu_{RF},\ _4H_0: \mu_{k\text{-}NN} = \mu_{ANN},\ _5H_0: \mu_{NB} = \mu_{SVM}$$

$$_6H_0: \mu_{NB} = \mu_{RF},\ _7H_0: \mu_{NB} = \mu_{ANN},\ _8H_0: \mu_{SVM} = \mu_{RF},\ _9H_0: \mu_{SVM} = \mu_{ANN} \text{ and } _{10}H_0: \mu_{ANN} = \mu_{RF}$$

From the results of ANoVA, it has been observed that $_1H_0$ is false as $p > 0.05$, so there is not any significant difference between the performances of *k*-NN and NB classifiers. Similarly,

$_2H_0$, $_4H_0$, $_8H_0$, $_9H_0$ and $_{10}H_0$ have $p > 0.05$, which shows that these classifiers have no significant difference between their performances. The null hypothesis $_3H_0$, $_5H_0$, $_6H_0$ and $_7H_0$ have $p < 0.05$, which suggests that these classifiers have significant difference between their performances. Though the performance of SVM classifier has been the best, it is not statistically different from the other two classifiers, namely, RF and ANN.

Table 4.20: *p*-values obtained from ANoVA when performances of classifiers are compared

	NB	**SVM**	**RF**	**ANN**
***k*-NN**	0.82	0.08	0.01	0.05
NB		0.03	0.002	0.02
SVM			0.66	0.93
RF				0.71

4.3 Experiment-III: Validation of classification results

This section validates the result obtained in experiments I and II for the classification of web services. It has been observed from the previous experiments that SVM classifier emerges as the best classification model for web services with both optimization and without optimization of configuration parameters.

In order to validate that SVM works better in all conditions, we have applied the performing classification models, namely, NB, *k*-NN, SVM and RF on the dataset of Experiment-I with parameters optimization, in order to validate that SVM still holds its place. In this part, we have considered dataset size *n* as 200, $k = 10$ and bi-gram for the experimentation work, and considered the optimum configuration of the classification models from the Experiment-II.

Table 4.21 shows the result for NB. It has been observed that accuracy obtained is 75.3%, recall is 70.5% and precision achieved is 71.9% by this classifier.

Table 4.22 shows the results for *k*-NN classifier. It has been observed that *k*-NN classifier achieved accuracy of 78.0%, recall of 74.3 % and precision of 74.8%. Similarly, Table 4.23 shows the confusion matrix for SVM. It has been observed that SVM achieves accuracy of 79.5%, recall of 76.7% and precision of 84.5%.

Table 4.21: Confusion matrix for NB

	True cnt	True news	True finance	True business	True currency	class precision (%)
pred. cnt	55	1	1	4	1	88.7
pred. news	2	12	3	2	0	63.1
pred. finance	4	6	20	3	1	58.8
pred. business	0	1	2	9	0	75.0
pred. currency	3	2	1	0	17	73.9
class recall (%)	85.9	54.5	74.0	50.0	89.4	

Table 4.22: Confusion matrix for *k*-NN

	True cnt	True news	True finance	True business	True currency	class precision (%)
pred. cnt	57	3	4	4	0	83.8
pred. news	0	17	4	2	0	73.9
pred. finance	3	1	15	3	0	68.1
pred. business	3	0	1	9	0	69.2
pred. currency	1	1	3	0	19	79.1
class recall(%)	89.0	77.2	55.5	50.0	100	

Table 4.23: Confusion matrix for SVM

	True cnt	True news	True finance	True business	True currency	class precision (%)
pred. cnt	54	0	1	1	0	96.4
pred. news	0	15	1	0	0	93.7
pred. finance	9	7	25	7	3	49.0
pred. business	2	0	0	10	0	83.3
pred. currency	0	0	0	0	16	100
class recall (%)	83.1	68.2	92.6	55.5	84.2	

Similarly, Table 4.24 shows the result for RF. This model has achieved accuracy of 58.0%, recall of 43.0% and precision of 90.1% respectively. It can be observed from the experimental results that SVM outperforms all the experimental results. This validates that SVM can be the best choice for web services classification.

Table 4.24: Confusion matrix for RF

	True cnt	True news	True finance	True business	True currency	class precision (%)
pred. cnt	64	18	24	15	6	50.4
pred. news	0	4	0	0	0	100
pred. finance	0	0	3	0	0	100
pred. business	0	0	0	3	0	100
pred. currency	0	0	0	0	13	100
class recall (%)	100	18.2	11.1	16.6	68.4	

We have further noticed that performance of SVM is better on the selected test data, this model performs better on sparse data in comparison to other models. Other models such as Random Forest *etc.* do perform slightly worse in scenarios when data has high dimensions. Performance of these models reduces as they require higher number of trees. This is the situation exactly when SVMs work well. SVMs also stand out for their robustness to high dimensional data.

4.4 Chapter summary

In this chapter, we have presented an approach for automatic classification of web services using machine learning methods and leveraging NLP based text processing techniques. This chapter presents the performance analysis of prominent machine learning models NB, *k*-NN, ANN, SVM and RF for the web services classification problem.

Three types of experiments have been conducted in order to evaluate the classification model and recommend the best model for web services discovery. Experiment-I was conducted without parameter optimisation of classification models. Experiments-II was conducted with the optimized parameters of classification models and combination of NLP techniques. Experiment-III was performed on the dataset of Experiment-I with optimized parameters of models and NLP processing in order to validate that the results still hold correct.

The result shows that there are differences between classifier's performances when applied to the web services classification area. This chapter demonstrates that how the performance is influenced by dataset size, the number of folds using cross-validation and NLP techniques. It has been observed that SVM outperforms all the classifiers and provided best results in all the experiments.

It has also been observed that the selection of the pre-processing techniques is effective in improving the classification results. The experimental results are quite consistent and suggest that SVM can be configured optimally to provide the best performance for web services classification and can be considered as a competitive model.

Chapter 5
Semantic Similarity Based Web Services Ranking

Selecting the most appropriate service from discovered services is the challenging activity in web services discovery. Some of the existing approaches are primarily based on syntactic descriptions, which ignore semantic relations between words, leading to unsatisfactory results. In this chapter, we propose a new approach for web services ranking using *WordNet* based ontology to capture the relations between the words. This phase of the framework aims to associate a similarity score to the web services and provides the most relevant web services to the users.

The proposed framework uses path-length based similarity algorithm in order to find a similarity score for ranking and ordering of web services. This approach is more concerned with the meaning of words and is based on the concept that how much two words match semantically. This chapter is organized into five sections. Section 5.1 describes the preparation of the test data for the experimental work. Section 5.2 provides detail about the ranking phase of *ACRWebS* framework. Section 5.3 provides summary of the chapter.

5.1 Test data preparation

We have used 22 web services for the experimentation purpose in this work (Table 5.1). Various web services features, such as services names, operation names and input/output parameters are considered for the experimental work. These features define behavior of web services and are used for calculating the similarity score with the user query. For example, the features associated with the service *weather* are: *weather, information, City, Forecast, Zip, Decryption, Picture, State, Station, Date, Temperature, Precipitation, Night, Time, Day, Humidity, Wind, Pressure, Visibility, Wind, Chill, US,* and *Hourly.* Similarly, the features associated with other web services have been identified.

Table 5.1: List of web services for experimentation

S. No.	Service Name	Category
1	Weather	Weather
2	Globalweather	Weather
3	Weatherforecast	Weather
4	USCityweatherinfo	Weather
5	Weathernet	Weather
6	WeatherInfo	Weather
7	Usweather	Weather
8	Graphical.weather.com	Weather
9	Currencyexchange	Currency
10	Globalcur	Currency
11	Foreignexchange	Currency
12	CurrencyConverter	Currency

Continued on next page

13	Currencyws	Currency
14	IEuroservice	Currency
15	CcurrencycService	Currency
16	PwspNoCentrbankCurRates	Currency
17	Stockexchangeinfo	Stock
18	Stockinfo	Stock
19	StockExchangeInfo	Stock
20	StockValue	Stock
21	NYSEstockexchange	Stock
22	NASDAQstockexchange	Stock

5.2 *ACRWebS*: Ranking phase

This phase of the framework is responsible for ranking of web services based on the similarity score of user queries and web services. Most of the current techniques for discovery of web services use an exhaustive search into the full repository, which is quite time consuming as irrelevant services are also processed for the matching purpose. This exhaustive search can be reduced to a great extent by reducing the search space to functional domains or businesses area as per user requirement context.

A group of categories can be created where each category may comprise information about all the services supported and list of all available categories. For an example, a category related to travel will comprise services related to taxi services, rail travel, hotel reservation, air travel and a user search of the travel category may provide user a referral directing to all of them. In such scenario, the general travel category will accept the query to the rail travel or air travel on behalf of the requester. Such grouping will make the discovery process easy as services requester will have the option to select one of the required categories from available list of web services categories. Also, while calculating the level of relevance

between a user query and web services, we require a match between the user query and the data on the services, which qualifies and rank them in the order from higher relevance score to lower relevance score.

To show the effectiveness of proposed approach, we have conducted an experiment with user queries from each of the service category of *weather*, *currency* and *stock.* For example, let us consider a query q_1 such as *weather of USA*, and the first service ws_1 picked for similarity match as *USCityweatherinfo*. After pre-processing q_1, the word vector in user query as {*weather, USA*}. Similarly, word vector in web service WSDL file is {*city, zip, state, weather, forecast, date, temperatures, wind, pressure, chill, US*}. As per the proposed semantic similarity algorithm, the first word of q_1, *i.e., weather* is matched with each word of ws_1. The similarity score for *weather* and *city* is 0.08, similarity score for *weather* and *zip* is also 0.08 and so on. In the same fashion, the similarity score of word *USA* of q_1 is matched with all words in ws_1. The cumulative similarity score obtained is 3.18 and average similarity score achieved for user query is 1.59. Now, q_1 is compared with ws_2 and the process is repeated with other web services data.

We have further fixed a semantic similarity threshold to decide for the minimum similarity score that should exist between user query and web service to infer that web service should be accepted. We required a value which is small, but not too small to avoid the irrelevant web services in the result. We empirically calculated the level of threshold with the help of plot for recall, precision and *f*-measure as shown in Figures 5.1-5.3 for the selected queries. It has been observed from the plots that value of *f*-measure is higher when threshold value is less than or equal to 0.5. As such, the threshold has been fixed at 0.5.

5.2.1 Experimental results and discussion

In order to evaluate the effectiveness of proposed framework, we have considered different test scenarios. The effectiveness of the approach is evaluated using the statistical measures, namely, precision and recall as these are the basic measures for evaluating a search strategy. In each of the test scenario, we have considered different user queries related to a service category, *e.g.*, we have taken a user query as *weather forecast* from category *weather*. In the experiment, we have executed this query against the *weather* category of services data and the experimental results are shown in Tables 5.2. It can be seen from results that system provides 100% recall and 100% precision for the queries *weather forecast* and *weather report*. For the query *weather by zip code*, three services, namely, *Globalweather*, *Weathernet* and *Graphical.weather.com* are rejected out of the eight services. This is due to the fact that these services lack *zip code* related information in the description of web services and accordingly, a less similarity score is obtained, though the services belongs to the *weather* domain. We have achieved an average recall of 93.7% and average precision of 100% for various queries in *weather* domain.

Table 5.3 contains the similarity score for the *currency* domain area. It has been observed that proposed approach achieves an average recall of 100% and average precision of 100% for the queries *currency convert* and *currency of country*, respectively. Similarly, for the *stock* domain, Table 5.4 shows the result for user queries *share price*, *share quote of a company* and *stock information by date* from the *stock* domain. It has been noted that proposed approach achieves an average recall of 84.1% and average precision of 100% for the various user queries from *stock* domain.

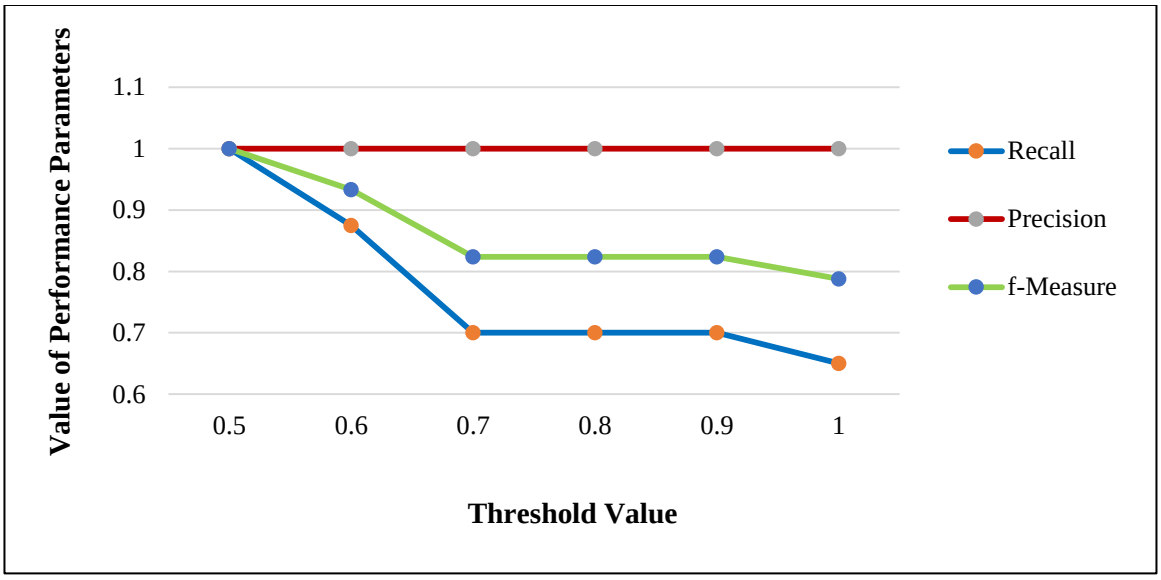


Figure 5.1: Recall, precision and *f*-measure for query weather of USA in *weather* domain

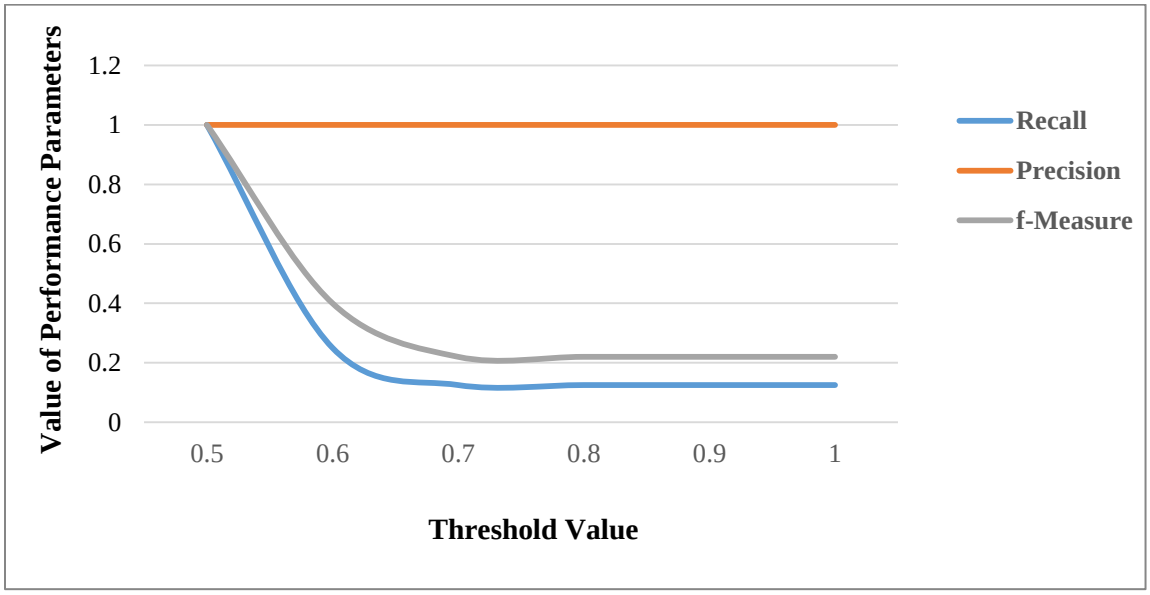


Figure 5.2: Recall, precision and *f*-measure for query Currency Converter in *currency* domain

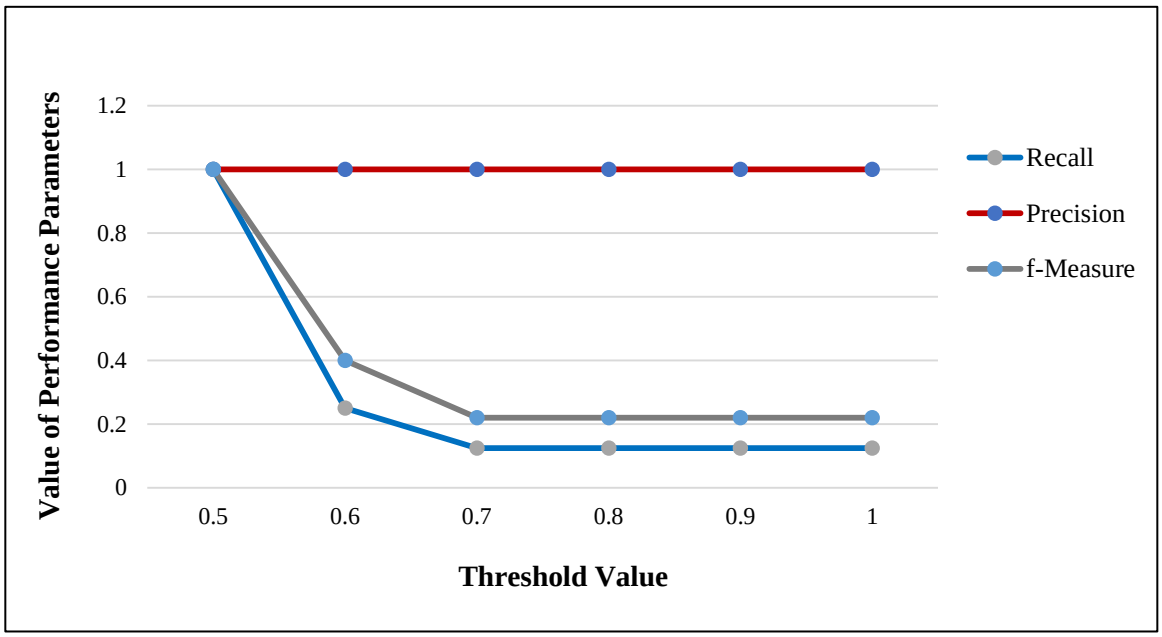


Figure 5.3: Recall, precision and *f*-measure for query Share Price in *stock* domain

On an average, the proposed approach achieves precision of 100% and recall of 93.5%, respectively. The performance comparison of our proposed approach and the similar approaches by Maheswari *et al.* (2014) and Ganapathy and Surianarayanan (2010) are shown in Table 5.5. We have selected these prominent approaches from the cited references as these two are similar to our work and provided their detailed experimental results in terms of recall and precision.

If our approach is compared with the work of Maheswari *et al.* (2014), we can see that our proposed approach provides better results in terms of the recall and precision. They achieved an average precision of 44% and average recall of 65% using semantic structure matching methods. Comparatively, our approach provided an improvement in precision by 47.7% and a marginal decrease in recall by 4.9%. This decrease in recall is due to the fact that some of the web services contain keywords, which do not contain information relevant to the user query. They have not provided details about the used test dataset in their research work.

Similarly, if we compare our approach with the results of Ganapathy and Surianarayanan (2010), it has been observed that our approach improved recall by 5.3% and it also improves precision by 26.3%. They have experimented on a dataset of 100 web services from multiple business areas, namely, communication, education, film, food, medical and travel. They have used a set of ten test queries for experimentation purpose and evaluated performance using precision and recall.

Table 5.2: Similarity scores in *weather* domain

Query	Service name	Similarity score	Decision
Weather forecast	Weather	1.89	Accepted
	USCityweatherinfo	1.37	Accepted
	Graphical.weather.com	1.25	Accepted
	Weatherforecast	1.00	Accepted
	Globalweather	0.50	Accepted
	Weathernet	0.50	Accepted
	WeatherInfo	0.50	Accepted
	Usweather	0.50	Accepted
Weather USA	Weather	1.89	Accepted
	Weatherforecast	1.67	Accepted
	USCityweatherinfo	1.64	Accepted
	Usweather	1.00	Accepted
	Graphical.weather.com	0.91	Accepted
	Weatherinfo	0.66	Accepted
	Weathernet	0.65	Accepted
	Globalweather	0.50	Accepted

Continued on next page

Weather Report	Weather	1.50	Accepted
	WeatherInfo	1.00	Accepted
	Usweather	1.00	Accepted
	USCityweatherinfo	0.87	Accepted
	Graphical.weather.com	0.79	Accepted
	Globalweather	0.50	Accepted
	Weatherforecast	0.50	Accepted
	Weathernet	0.50	Accepted
Weather by Zip Code	Weatherforecast	1.63	Accepted
	WeatherInfo	1.53	Accepted
	Usweather	1.53	Accepted
	USCityweatherinfo	1.20	Accepted
	Weather	1.16	Accepted
	Graphical.weather.com	0.40	Rejected
	Globalweather	0.40	Rejected
	Weathernet	0.33	Rejected

Table 5.3: Similarity scores in *currency* domain

Query	Service Name	Similarity Score	Decision
Currency Convert	CurrencyConverter	1.00	Accepted
	PwspNoCentrbankCurRates	0.60	Accepted
	Currencyexchange	0.50	Accepted
	Globalcur	0.50	Accepted
	Foreignexchange	0.50	Accepted

Continued on next page

	Currencyws	0.50	Accepted
	IEuroservice	0.50	Accepted
	CcurrencycService	0.50	Accepted
Currency of Country	CcurrencycService	1.00	Accepted
	Globalcur	1.00	Accepted
	PwspNoCentrbankCurRates	0.80	Accepted
	Currencyexchange	0.70	Accepted
	Foreignexchange	0.50	Accepted
	CurrencyConverter	0.50	Accepted
	Currencyws	0.50	Accepted
	Ieuroservice	0.50	Accepted

Table 5.4: Similarity scores in *stock* domain

Query	**Service name**	**Similarity score**	**Decision**
Share Price	NYSEstockexchange	1.55	Accepted
	NASDAQstockexchange	1.12	Accepted
	StockValue	1.10	Accepted
	StockExchangeInfo	0.50	Accepted
	Stockexchangeinfo	0.50	Accepted
	Stockinfo	0.36	Rejected
Share Quote of acompany	NASDAQstockexchange	1.20	Accepted
	NYSEstockexchange	1.16	Accepted
	StockValue	1.13	Accepted

Continued on next page

	Stockexchangeinfo	0.75	Accepted
	Stockinfo	0.50	Accepted
	StockExchangeInfo	0.25	Rejected
Stock information by date	NYSEstockexchange	2.00	Accepted
	StockValue	2.00	Accepted
	NASDAQstockexchange	1.50	Accepted
	StockExchangeInfo	0.91	Accepted
	Stockexchangeinfo	0.66	Accepted
	Stockinfo	0.33	Rejected

Table 5.5: Recall and precision measurement

	Recall (%)	**Precision (%)**
Our Approach	93.5	100.0
Maheswari *et al.* (2014)	98.4	52.3
Ganapathy *et al.* (2010)	88.2	63.7

We have also experimented the situation when the user fires a query in the irrelevant domain. The query "*weather forecast*" is compared with the *currency* domain. As given in Table 5.6, the similarity score in irrelevant category is very low, as expected. In this scenario, the achieved values of recall is 83.3% and precision is 71.4% respectively.

Table 5.6: Similarity scores in irrelevant domain

Query	Service Name	Similarity Score
Weather forecast	CcurrencycService	0.00
	Globalcur	0.00
	PwspNoCentrbankCurRates	0.00
	Currencyexchange	0.00
	Foreignexchange	0.00
	CurrencyConverter	0.00
	Currencyws	0.00
	Ieuroservice	0.01
Stock quote of a company	currencyexchange	0.11
	globalcur	0.08
	foreignexchange	0.16
	CurrencyConverter	0.23
	Currencyws	0.42
	Ieuroservice	0.083
	ccurrencycService	0.23
	pwspNoCentrbankCurRates	0.53
Currency of country	Stockexchangeinfo	0.00
	Stockinfo	0.00
	StockExchangeInfo	0.1
	StockValue	0.35
	NYSEstockexchange	0.82
	NASDAQstockexchange	0.41
	Stockexchangeinfo	0.00

5.3 Chapter summary

In this chapter, a *WordNet* based semantic similarity approach has been presented. A study to demonstrate the efficiency of the proposed framework has been presented and experimental observations yielded satisfactory results. Experimental results on the selected dataset of web services show that incorporating *WordNet* in order to capture the semantic similarity between keywords is very effective in web services discovery process.

Chapter 6

Testing of *ACRWebS* for New Web Services

This chapter provides further experimentation on the proposed framework *ACRWebS* for four new web services created by the author. These web services are created for three business domains, namely, *currency*, *weather* and *stock*. The objective is to validate the framework on the newly created web services. Section 6.1 of the chapter provides description of these new web services and section 6.2 provides results in terms of statistical measurements. Section 6.3 provides the summary of the chapter.

6.1 Description of new web services

Table 6.1 shows the details for newly created web services. We have created four new web services from the business domains of *currency*, *weather* and *stock* and tried to validate the proposed approach using these web services.

Table 6.1: List of newly created web services

S. No.	Service Name	Domain
1.	USCurrencyExachange	currency
2.	IndiaWeatherByZip	weather
3.	IndiaMetroWeather	weather
4.	NSEStockExchange	stock

6.2 Experimental results

The test dataset of the newly created web services is prepared after extraction of their WSDL file information. This test dataset is then inputted to the framework for calculating the similarity score of the web services with the user queries.

To evaluate the effectiveness of proposed framework, different test scenarios have been considered. In each of the scenario, we have considered different user queries and different web services category. We have calculated similarity score based on the same steps as presented in Section 5.4. Tables 6.2-6.4 contain the results of the experiments conducted. In this chapter, we have taken two decisions about the user query and fetched service. First decision is based on the similarity score and second decision is taken by a human being. We have divided the similarity scores in three ranges: Little Similar, when $0 \leq$ Similarity Score ≤ 0.5; Similar, when $0.5 \leq$ Similarity Score ≤ 0.8; and Very Similar, when Similarity Score > 0.8. The human interpreter is also provided with these three types of similarity, and one decision for the category (Little Similar or Similar or Very Similar) without any knowledge of the similarity score or the above mentioned ranges of the similarity scores.

Tables 6.2-6.4 also contain semantic scores of the web services with the corresponding user query. *ACRWebS* based decision on the web services and human expert analysis is also included in these tables. We have abbreviated Little Similar, Similar and Very Similar by LS, S and VS, respectively in these tables.

It has been validated that the newly services are correctly scored by the proposed approach. As shown in Table 6.2, it has been observed that web service *IndiaMetroWeather* with attributes (*IndiaMetroWeather, getDelhiTemprature, getMumbaiTemperature, getBangaloreTemperature*) and user query *Weather forecast*, the score achieved is 0.50. Similarly, the score achieved with the service *IndiaWeatherByZip* against the same query is 1.00.

For the user query *Weather USA*, the similarity score achieved with service *IndiaMetroWeather* is 0.50 and with *IndiaWeatherByZip*, the similarity score is 1.00

respectively. As can be noted, the service *IndiaWeatherByZip* is more relevant to use query as it is having more similar keywords for operation parameters.

Similarly, for the user query *Weather Report,* the similarity score achieved with the services *IndiaMetroWeather* and *IndiaWeatherByZip* is 0.50 and 1.10 respectively.

For the user query *Weather by Zip Code*, the similarity score achieved with the service *IndiaMetroWeather* is 1.80 and with the service *IndiaWeatherByZip* score is 4.20. As can be noted, the service *IndiaWeatherByZip* is more relevant to use queries, as it is having more similar operations. This validates our proposed approach.

For the user query *India Weather,* the similarity score achieved with service *IndiaMetroWeather* is 1.00 and with *IndiaWeatherByZip* is 1.50. The similarity score for the user query *Metro City Weather* are 0.80 and 0.75 respectively with the services *IndiaMetroWeather* and *IndiaWeatherByZip*

Similarly, it can be seen from the Table 6.3 that the service *USCurrencyExachange* from category *currency* provided similarity score of 2.00 with respect to the user query *Currency Convert.* When the user query is *currency exchange*, the similarity score achieved is 15.00. For the query *USA Currency Exchange,* the similarity score is 5.30. As can be observed, this is the highest similarity score achieved for any web service given in the testing dataset. This is due to the reason that the operation attributes of the service *USCurrencyExachange* are very much similar to the user query (*Convert Currency*) and hence is the most relevant service returned to the user. This validates that the proposed approach is successful and meets human expert interpretations.

Table 6.2: Similarity measurement in *weather* domain

Query	Service Name	Similarity Score	Decision	Human Interpretation
Weather forecast	Weather	1.89	VS	VS
	USCityweatherinfo	1.37	VS	VS
	Graphical.weather.com	1.25	VS	VS
	Weatherforecast	1.00	VS	VS
	IndiaWeatherByZip	1.00	**VS**	**S**
	Weathernet	0.50	S	S
	WeatherInfo	0.50	S	S
	Usweather	0.50	S	S
	IndiaMetroWeather	0.50	S	S
Weather USA	Weather	1.89	VS	VS
	Weatherforecast	1.67	VS	VS
	USCityweatherinfo	1.60	VS	VS
	IndiaWeatherByZip	1.00	**VS**	**S**
	Usweather	1.00	VS	VS
	Graphical.weather.com	0.91	VS	VS
	Weatherinfo	0.66	S	S
	Weathernet	0.60	S	S
	IndiaMetroWeather	0.50	S	S
	Globalweather	0.50	S	S

Continued on next page

Weather Report	Weather	1.50	VS	VS
	IndiaWeatherByZip	1.10	VS	VS
	WeatherInfo	1.00	VS	VS
	Usweather	1.00	VS	VS
	USCityweatherinfo	0.87	VS	VS
	Graphical.weather.com	0.79	VS	VS
	Globalweather	0.50	VS	VS
	Weatherforecast	0.50	S	S
	Weathernet	0.50	S	S
	IndiaMetroWeather	0.50	S	S
Weather by Zip Code	IndiaWeatherByZip	4.25	VS	VS
	IndiaMetroWeather	1.80	VS	VS
	Weatherforecast	1.63	VS	VS
	WeatherInfo	1.53	VS	VS
	Usweather	1.53	VS	VS
	USCityweatherinf	1.20	VS	VS
	Weather	1.16	VS	VS
	Globalweather	0.40	**LS**	**S**
	Graphical.weather.com	0.40	LS	LS
	Weathernet	0.33	LS	LS
India Weather	IndiaWeatherByZip	1.50	VS	V S
	Weather	1.25	VS	VS
	IndiaMetroWeather	1.00	VS	VS
	USCityweatherinfo	0.87	VS	VS
	Graphical.weather.com	0.60	S	S
	Globalweather	0.50	S	S
	Weatherforecast	0.50	S	S

Continued on next page

	Weathernet	0.50	S	S
	WeatherInfo	0.50	S	S
	Usweather	0.50	S	S
Metro City Weather	Weather	1.20	VS	VS
	Weathernet	1.10	VS	VS
	USCityweatherinfo	1.00	VS	S
	Globalweather	0.80	VS	VS
	IndiaMetroWeather	0.80	VS	VS
	WeatherInfo	0.77	VS	VS
	IndiaWeatherByZip	0.75	S	S
	Weatherforecast	0.56	S	S
	Graphical.weather.com	0.50	VS	VS
	Usweather	0.40	LS	LS

Table 6.3: Similarity measurement in *currency* domain

Query	**Service Name**	**Similarity Score**	**Decision**	**Human Interpretation**
Currency Convert	USCurrencyExchange	2.00	VS	VS
	CurrencyConverter	1.00	VS	VS
	PwspNoCentrbankCur Rates	0.60	S	S
	Globalcur	0.50	S	S
	Foreignexchange	0.50	S	S
	Currencyws	0.50	S	S

Continued on next page

	IEuroservice	0.50	S	S
	CcurrencycService	0.50	**S**	**VS**
	Currencyexchange	0.50	**S**	**VS**
Currency of Country	UsCurrencyExchange	1.50	VS	VS
	CcurrencycService	1.00	S	S
	Globalcur	1.00	S	S
	PwspNoCentrbankCur Rates	0.80	S	S
	Currencyexchange	0.70	VS	VS
	Foreignexchange	0.50	VS	VS
	CurrencyConverter	0.50	VS	VS
	Currencyws	0.50	S	S
	Ieuroservice	0.50	S	S
USA Currency Exchange	UsCurrencyExchange	5.30	S	S
	Foreignexchange	0.73	S	S
	Currencyexchange	0.70	S	S
	CurrencyConverter	0.66	S	S
	CcurrencycService	0.56	S	S
	PwspNoCentrbankCurRates	0.48	LS	LS
	Ieuroservice	0.46	LS	LS
	Globalcur	0.43	**LS**	**S**
	Currencyws	0.40	LS	LS

From the Table 6.4, the similarity score achieved by the user query *share price*, with the newly added service *NSEStockExchange* is 0.88, which shows that it is very similar to the human interpretation. For the user query *Share quote of a company*, the score achieved for the same service is 0.90, which is again similar to human interpretation.

For the user query *Stock information by date*, the score achieved is 0.66 with the service *Stockexchangnfo*. As per the system, this is similar to the user query and matches to human interpretation. Similarly, the score achieved for the user query *NSE Stock Exchange Info* with service *NSEStockExchange* is 1.30. As per the proposed system recommendation, this is very similar to the user query.

Table 6.4: Similarity measurement in *stock* domain

Query	Service name	Similarity score	Decision	Human Interpretation
Share Price	NYSEstockexchange	1.55	VS	VS
	StockValue	1.10	VS	VS
	NASDAQstockexchange	1.10	VS	VS
	NSEStockExchange	0.88	VS	VS
	StockExchangeInfo	0.50	S	S
	Stockexchangeinfo	0.50	S	S
	Stockinfo	0.36	**LS**	**S**
Share Quote of a company	NASDAQstockexchange	1.20	VS	VS
	NYSEstockexchange	1.16	VS	VS
	StockValue	1.13	VS	VS
	NSEStockExchange	0.90	VS	VS
	Stockexchangeinfo	0.75	S	S
	Stockinfo	0.50	S	S
	StockExchangeInfo	0.25	LS	LS
Stock information by date	StockValue	2.00	VS	VS
	NYSEstockexchange	2.00	VS	VS
	NASDAQstockexchange	1.50	VS	VS

Continued on next page

	NSEStockExchange	1.00	S	S
	StockExchangeInfo	0.91	**VS**	**S**
	Stockexchangeinfo	0.66	S	S
	Stockinfo	0.33	**LS**	**S**
NSE Stock Exchange info	NYSEstockexchange	1.50	VS	VS
	NSEStockExchange	1.30	VS	VS
	NASDAQstockexchange	1.20	VS	VS
	StockValue	1.00	VS	VS
	StockExchangeInfo	0.83	**VS**	**S**
	Stockexchangeinfo	0.57	S	S
	Stockinfo	0.25	LS	LS

In order to evaluate how effective is our proposed system, we have computed the precision and recall based on the confusion matrix. Table 6.5 shows the confusion matrix for *stock* domain.

Table 6.5: Confusion matrix for *stock* domain

	LS	**S**	**VS**
LS	2	2	0
S	0	7	1
VS	0	2	13

The rows correspond to the known classes of the data. The columns represent corresponding predictions made by the system. Thus, the elements along the diagonal show the number of correct classifications. The off-diagonal elements show the incorrectly made classifications. As such, the value of recall for *stock* domain services in category LS is 100%, for the services in category S is 63.0% and for the services in category VS is 92.0%. The average value of recall for the class *stock* is 83.0%

The value of precision for *stock* domain services in category LS is 100%, for the services in category S is 92.0% and for the services in category VS is 78.0%. The average value of precision for the class *stock* is 90.0%.

Table 6.6 shows the confusion matrix for the *weather* domain.

Table 6.6: Confusion matrix for *weather* domain

	LS	**S**	**VS**
LS	3	0	0
S	1	20	1
VS	0	0	33

The value of recall for *weather* domain services in category LS is 100%, for the services in category S is 90.0% and for the services in category VS is 100.0%. The average value of recall for the class *weather* is 96.0%.

The value of precision for *weather* domain services in category LS is 75.0%, for the services in category S is 100.0% and for the services in category VS is 97.0%. The average value of precision for the class *weather* is 90.0%.

Table 6.7 shows the confusion matrix for *currency* domain.

Table 6.7: Confusion matrix for *currency* domain

	LS	**S**	**VS**
LS	3	0	0
S	1	15	0
VS	0	2	6

The value of recall for *currency* domain services in category LS is 75.0%, for the services in category S is 88.0% and for the services in category VS is 100.0%. The average value of recall for the class *currency* is 88.0%.

The value of precision for *currency* domain services in category LS is 100.0%, for the services in category S is 93.0% and for the services in category VS is 75.0%. The average value of precision for the class *currency* is 89.0%.

The experimental result shows that use of *WordNet* can be very effective in the web services discovery process. Table 6.8 contains the performance results for the three categories of services considered in this work. Average recall of the proposed approach is 89.7% and the average precision is 89.0% for the three categories considered in this work. The experimental results suggest that use of *WordNet* based approach is effective and improves web services discovery process.

Table 6.8: Recall and precision measurements

	Recall (%)	**Precision (%)**
Stock	90.0	83.0
Weather	90.0	96.0
Currency	89.0	88.0

6.3 Chapter summary

In this chapter, the proposed approach has been validated on the new web services. A set of four new web services was created and tested on the *ACRWebS* framework. These web services were also interpreted by human expert to evaluate the proposed approach. Experimental results show that incorporating *WordNet* in order to capture the semantic relationship between user queries and web services data is very effective in the web services discovery process. It has been observed that results of the proposed approach and human observations match to a great extent.

Chapter 7
Conclusions and Future Scope

The work in this thesis presents a framework that provides a web services selection approach and recommends relevant services to the users based on their requirements. The two ideas investigated in this work are the semantic relatedness and classification techniques in the area of web services discovery.

In this thesis, a comprehensive review of classification techniques and semantics in the web services discovery has also been performed. A study on the various techniques such as QoS, semantics and ontologies used in match-making of web services has been conducted. A variety of learning algorithms has been explored and it has been observed that combination of classification and semantic technique is the most promising approach and can help to improve the accuracy of web services discovery approaches.

In this work, we have proposed a new hybrid approach for web services discovery. It has been shown that the proposed ranking model improves the search strategies and thus retrieves the most relevant web services. This thesis has covered two important aspects: Use of machine learning algorithms to assist in web services classification and addressing the problem of best selection using semantics based match making. The main benefit of proposed framework is that it can reduce time to discover web services and thus can have a significant impact on the software development cost.

This thesis proposes a semantic based efficient discovery framework named *ACRWebS*, which supports finding best available web services. A case study to demonstrate the feasibility of the proposed framework has been presented and the experimental observations yielded satisfactory results. The use of semantic similarity for web services, significantly improved the probability of finding the right web services. The proposed framework compares user query with web services within a specific category and provides better results as irrelevant services are not processed by the framework.

The returned services are also interpreted by human experts to validate the results. Results show that incorporating *WordNet* in the framework for capturing the semantic similarity between words is effective and can enhance the web services discovery process. It has been observed from results that there is a high similarity between the proposed approach and human expert observations.

Empirical analysis of results suggests that SVM provides better performance in comparison with other chosen models. The proposed approach is simpler and greatly reduces the complexity of calculation because there is no specific requirement to add extra taxonomy or ontology in the WSDL file of web services.

Though the use of *WordNet* seems promising area, but there are some challenges and limitations as well as outlined below:

i) Different parts of *WordNet* have different granularity for the description of word senses.
ii) *WordNet* is not multilingual, but there are *WordNet* versions for a large number of languages. These different versions of *WordNet* have different format and coverage.
iii) *WordNet* also focuses on paratactic semantic relations between single words.

7.1 Main Contributions

There are three main contributions of this work.

i) A new framework *ACRWebS* dealing with user requirements and semantic relationship between user queries and web services has been proposed.

ii) Various classification approaches have been evaluated on the web services data. Experiments have been performed using parameter optimization techniques to provide the improved web services classification results.

iii) A comparison of the proposed approach with other approaches has been performed and it has been observed that *ACRWebS* is more effective.

7.1.1 Advantages of proposed approach

i) The proposed approach allows enhancing the web services semantics information without any semantic annotation.

ii) The web services user can search their required services without providing the detailed technical specifications. This is an important point as the users of services have limited knowledge during the initial stage of the services discovery.

iii) This approach has an advantage from traditional approaches which are based on annotation and ontology. The traditional approaches are quite costly and require much effort for ontology management.

iv) *WordNet* is not domain specific.

v) The proposed approach does not require any changes in existing WSDL file structure of web services, keeping the compatibility with the structure of old web services.

vi) Advantages over ontology based approaches:

- It is well known that use of ontology based approaches has its own related challenges. Most of the web services use semantic annotation, which is based on the domain. Although ontology techniques provide semantic description, but they do not adapt well to heterogeneous domains. In the case of web services, where a large vocabulary is required, adoption of ontologies is a challenge. Various initiatives, projects and languages such as OWL-S, WSMO, METEOR-S and SWSL have been proposed for adding semantics to web services. These prefer usage of ontology as a base of mapping and matching different type of services. It is not possible to have a single ontology

that can fully cover a specific domain or which can satisfy the requirements of all users.

- Due to lack of standard methodology for ontology development, it is difficult to understand the intended meaning of concepts. Ontologies are domain dependent. In the case of mash-ups where services from different domains are involved, domain dependent annotations introduces ontology matching problem.
- The maintenance and creation of ontologies involves high cost as these require special expertise

vii) Our proposed approach works in a specified category based on the user input. This avoids extra comparison efforts with irrelevant search operations, thus saving the search and processing time.

7.2 Future scope

There is still a lot left for future work, including the use of a better classification and similarity match-making techniques. It is worth noting that available standards and technologies are not sufficient to meet the demand of web services discovery. It is foreseen that semantics will play a big role in future and leveraging semantic technology in the area of web services will be a common strategy.

Future work could be to design more exhaustive frameworks by including a more efficient strategy for ranking with the use of the semantic analysis, so that the framework can evaluate more precisely the relevance of web services with the user queries.

We can conclude that despite the quantum of work done, much needs to be done for improvement of web services discovery mechanism. As has been observed from the literature survey, we see a trend of hybrid approaches to address this pain area of web services.

In nutshell, following areas can be explored in future for making the web services discovery process even more efficient.

- Web services feature selection methods can be improved for better web service classification process.
- Use of optimized classifiers to improve performance in terms of accuracy, recall, precision and time to train a classifier.
- Better use of semantics and ontology for the web services classification, clustering and informational retrieval.
- Better ranking methods to arrange selected web services as per user's requirement.
- Hybrid approaches could be used for better results.
- There has been some progress in the area of *WordNet* development for specialized domains and multiple languages. This may be used in future for better flexibility and accuracy in the web services discovery area.

List of Publications

Journals publications

- Bhardwaj K.C. and Sharma R. K. (2015) 'Machine Learning in Efficient and Effective web Service Discovery', *Journal of Web Engineering,* vol. 14, issues 3-4, pp.196-214, SCI-indexed IF=0.444.

- Bhardwaj K.C. and Sharma R. K. (2016) 'Web Services Discovery Framework using Classification and Semantic Similarity', International *Journal of Artificial Intelligence and Knowledge Discovery,* vol. 6, no. 2. pp. 13-19**,** indexed in *COPERNICUS, DOAJ* and *Google Scholar*.

- Bhardwaj K.C. and Sharma R. K. (2016) 'Empirical analysis of machine learning models for web services classification', *International Journal of Science and Engineering,* **Accepted for Publication**, indexed in *COPERNICUS, DOAJ, Google Scholar, ScienceCentral* and *CrossRef*.

- Bhardwaj K.C. and Sharma R. K. (2016) 'Ontologies: A review of web service discovery techniques', *International Journal of Energy, Information and Communication,* **Accepted for Publication,** indexed in *EBSCO, ULRICH, DOAJ, J-Gate* and *Cabell*.

Conference presentation

Bhardwaj K.C. and Sharma R. K. (2016) 'Web Service Classification using Machine Learning Models', in Proc. of International Conference on Recent Trends in Engineering and Material Sciences (ICEMS-2016),17-19 March 2016, Jaipur.

Appendix – A

Prototype UI of ACRWebS

The prototype of *ACRWebS* includes two-level services discovery interface. The proposed framework *ACRWebS* provides a drop down category where the consumer of the services can select one of the categories as shown in Figure 3.8. Once a user selects a category, a list of services under that category is shown as per the user query specifications entered via the UI. Figure 3.9 and Figure 3.10 shows the list of returned services when the selected categories are *stock* and *weather*. The returned services are ordered by descending value of the score. A user can select the highest score services and can consume that in their application.

Figure A1: *ACRWebS* user interface

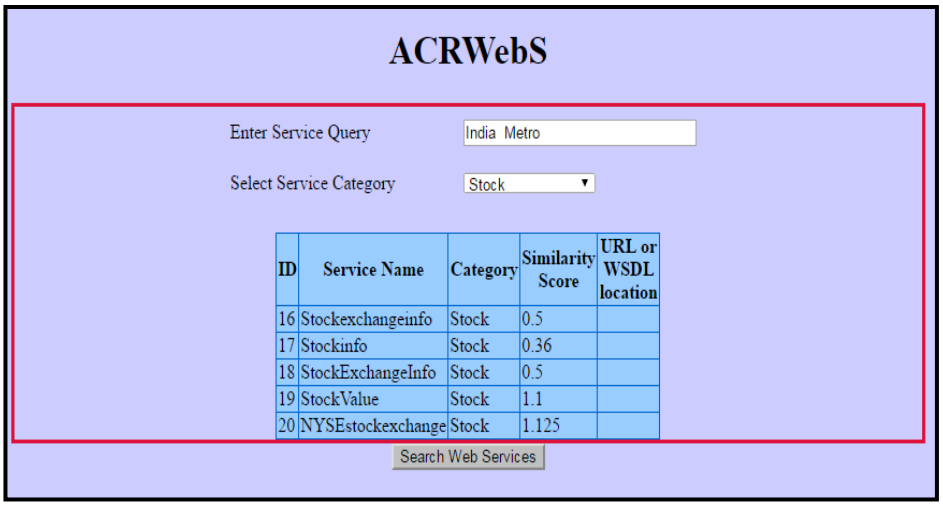


Figure A2: Selection of a category and entering query in *ACRWebS*

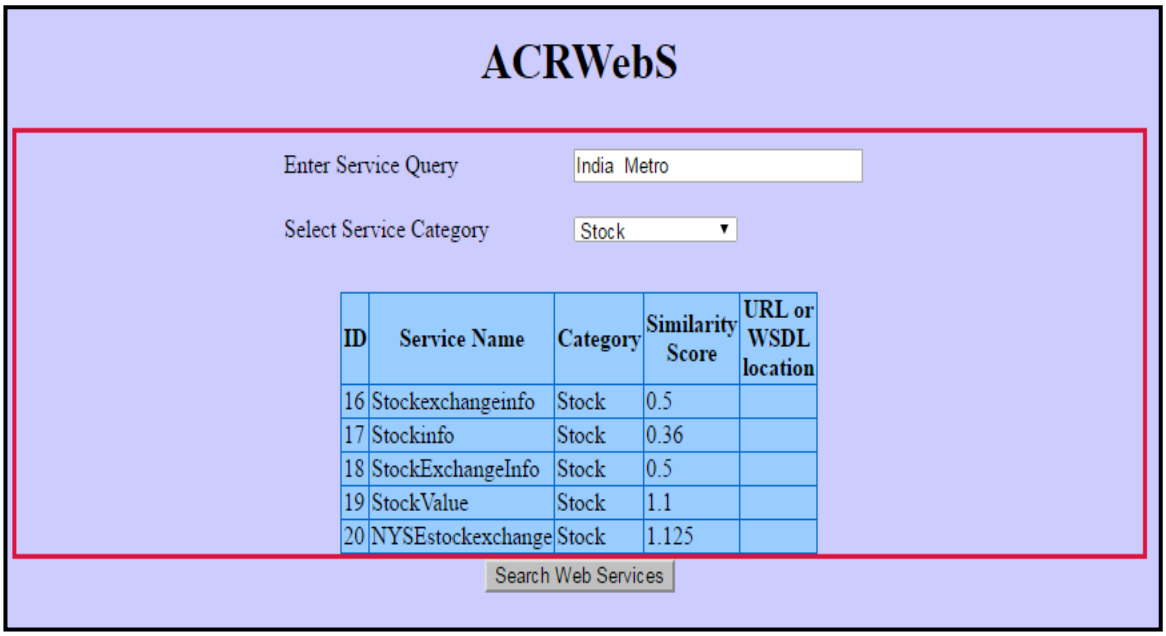


Figure A3: Results for a selected query in *ACRWebS*

References

[1] Agrawal R., Imieliński T. and Swami A. (1993) 'Mining association rules between sets of items in large databases', in *Proc. of 1993 ACM SIGMOD international conference on Management of data-SIGMOD,* vol. 22, 25-28 May, pp. 206-216.

[2] Akkiraju R., Farrell J., Miller J., Nagarajan M., Schmidt M., Sheth A. and Verma K. (2004) '*Web service Semantics-WSDL-S*', Technical report, LSDIS lab and IBM Corporation.

[3] Al-Masri E. and Mahmoud Q. H. (2009) 'Discovering the best web service: A neural network based solution', in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics,* San Antonio, pp. 4250-4255.

[4] Altman N. S. (1992) 'An introduction to kernel and nearest-neighbor nonparametric regression', *The American Statistician,* Aug. 1992, vol. 46, no.3, pp. 175-185.

[5] Banaei-Kashani F., Chen C-C. and Shahabi C. (2004) 'WSPDS: Web Services Peer to peer Discovery Service', *University of Southern California, Los Angeles, California,* April, 2004, pp.733-739.

[6] Banerjee S. and Pedersen T. (2002) 'An adapted Lesk algorithm for word sense disambiguation using Word-Net', in *Proc. of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City,* 17-23 Feb., pp. 136-145.

[7] Becker J., Mueller O. and Woditsch M. (2010) 'An ontology-based natural language services discovery engine-design and experimental evaluation', in *Proc. of 18th European conference on information systems, Pretoria, South Africa.*

[8] Bennaceur A., Issarny V., Johansson R., Moschitti A. and Sykes D. (2012) 'Machine Learning for automatic classification of web service interface descriptions', in *Proc. of ISoLA Workshops*, pp. 220-231.

[9] Bruijn, J., Lara, R., Arroyo, S., Gomez, J. M., Han, S. K. and Fensel D., 2005. A Unified Semantic Web Services Architecture based on WSMF and UPML, *International Journal of Web Engineering and Technology*, vol. 2, no. 2, pp. 148-180.

[10] Bruno M. and Canfora G. (2005) 'Using SVM and concept analysis to support web service classification and annotation. In *Proc. of IEEE international conference on e-Technology,*

e-Commerce and e-Service, pp. 138-143.

[11] Chakraverty S. and N. Sukanta (2016) 'Numerical Solution of Fuzzy Stochastic Differential Equation' , *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 1, pp. 555-563.

[12] Chang C. C. and Lin C. J. (2011) 'LIBSVM: A library for support vector machines', *Journal of ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 2, no. 3, article no. 27.

[13] Chen H., Yu T. and Lin K.J. (2003) 'QCWS: an implementation of QoS-capable multimedia web services', in *Proc. of IEEE 5th International Symposium on Multimedia Software Engineering, IEEE,* pp.38-45.

[14] Cortes C. and Vapnik V. (1995) 'Support-vector networks', *Machine Learning*, vol. 20, no. 3, pp. 273-297.

[15] Coulouris G., Dollimore and Kindberg T. (2011) 'Distributed Systems Concepts and Design', *5th edition, Addison-Wesley.*

[16] Crasso M., Zunino A. and Campo M. (2008) 'AWSC: An approach to web services classification based on machine learning techniques', *CONICET, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*. vol. 37, pp. 25-36.

[17] Cufoglu A., Lohi M. and Madani K. (2008) 'A Comparative Study of Selected Classifiers with Classification Accuracy in User Profiling', in *Proc. of the 7th international conference on Machine Learning and Application, San Diego, California, USA,* pp. 787-791.

[18] Deep K., Bansal J. C., Veeramachaneni K. and Osad L. (2009), 'Information Sharing Strategy among Particles in Particle Swarm Optimization Using Laplacian Operator', *Swarm Intelligence Symposium, 2009, SIS '09, IEEE* .

[19] D'Mello D.A. and Ananthanarayana V.S. (2008) 'A QoS Model and Selection Mechanism for QoS-Aware Web Services', in *Proc. of the International Conference on Data Management (ICDM 2008),* pp. 611-621.

[20] Dorn C. and Dustdar S. (2010) 'Weighted fuzzy clustering for capability-driven service aggregation', *IEEE International Conference on Service-Oriented Computing and Applications*, pp. 1-8.

[21] Farkas P. and Charaf H. (2003) 'Web Services Planning Concepts', in *Proc. of 1st*

International Workshop on C# and .NET Technologies on Algorithms, Computer Graphics, Visualization, Distributed and WEB Computing, vol. 1, no. 1-3.

[22] Fensel D., Lausen H., Polleres A., Bruijn J., Stollberg M., Roman D. and Domingue J. (2007) 'Enabling Semantic Web Services: the Web service Modeling Ontology', Springer-Verlag Berlin Heidelberg.

[23] Ganapathy G. and Surianarayanan C. (2010) 'An Approach to Identify Candidate Services for Semantic Web Services discovery', in *Proc. of the IEEE International Conference on Service Oriented Computing and Applications, Perth, WA*, pp. 1-4.

[24] Gao H., Stucky2 W. and Liu L. (2009) 'Web Services Classification Based on Intelligent Clustering Techniques', in *Proc. of IFITA*, vol. 5, pp. 242-245.

[25] Garcia D.Z.G. and Toledo M. B. F. (2006) ' A Web Service Architecture Providing QoS Management', *in Proc. of 4th Latin American Web Congress (LA-Web 2006), 25-27 October, Cholula, Puebla, Mexico*, pp.189-198.

[26] Gholamzadeh N. and Taghiyareh (2010) 'Ontology based fuzzy web services clustering', in *Proc. of International Symposium on Telecommunications*, pp. 721-725.

[27] Goyal N., Goyal P., Venkatramaiah K., Deepak P. C. and Sanoop P. S.(2009) ' An Efficient Density Based Incremental Clustering Algorithm in Data Warehousing', in *Proc. of 2009 International Conference on Computer Engineering and Applications IPCSIT*, IACSIT Press, Singapore, vol. 2, pp. 482-486.

[28] Green S. (1997) 'Building Hypertext Links in Newspaper Articles Using Semantic similarity', in *Proc. of Third Workshop on Applications of Natural Language to Information Systems (NLDB'97)*, pp. 178-190.

[29] Green S. (1999) 'Building Hypertext Links by Computing Semantic similarity', *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 11, no. 5, pp. 713-730.

[30] Gruber T. R. (1993) 'A Translation Approach to Portable Ontologies', *Knowledge Acquisition*, vol. 2, no. 5, pp. 199-220.

[31] Han J. (2006) 'Data Mining: Concepts and Techniques', *Second edition, Morgan Kaufmann Publishers, San Francisco*.

[32] Hentenryck P.V. and Saraswat V. (1996) 'Strategic Directions in Constraint Programming', *Journal of ACM Computing Surveys*, vol. 28, no. 4, pp. 701-726.

[33] Hess A. and Kushmerick N. (2003) 'Learning to attach semantic metadata to web services', in *Proc. of 2nd Int. Semantic Web Conference (ISWC), Florida, USA*.

[34] Ho Tin Kam (1995) 'Random Decision Forests (PDF)', in *Proc. of the 3rd International Conference on Document Analysis and Recognition, Montreal,* pp. 278-282.

[35] http://www.remotemethods.com

[36] Huang C.-L., Lo C.-C., Chao K.-M. and Younas M. (2006) 'Reaching consensus: A moderated fuzzy web services discovery method', *Information & Software Technology*, vol. 48, no. 6, pp. 410-423.

[37] Ismaili F., Zenuni X. and Raufi B. (2009) 'A novel approach for efficiently finding web services on the web', in *Proc .of the ITI 31st International Conference on Information Technology Interfaces*, 22-25 June 2009, IEEE, pp. 603-608.

[38] Jamous M. M. and Safaai B. D. (2011) 'Web services non-functional classification to enhance discovery speed', *International Journal of Computer Science Issues*, vol. 8, no. 4.

[39] Karimpour R. and Taghiyareh F. (2009) 'Conceptual discovery of web services using *WordNet', in Proc. of IEEE Asia-Pacific Services Computing Conference, IEEE Computer Society*, pp. 440-444.

[40] Kim S., Han K., Rim H. and Myaeng S. (2006) 'Some Effective Techniques for Naïve Bayes Text Classification', *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 11, pp. 1457-1466.

[41] Lara R., Bruijn J., Arroyo S., Gomez J. M., Han S. K. and Fensel D. (2005) 'A Unified Semantic Web Services Architecture based on WSMF and UPML', *International Journal of Web Engineering and Technology*, vol. 2, no. 3, pp. 148-180.

[42] Laura R.T., Sergio P.I., Bergamaschi S. and Mena E. (2011) 'Using semantic techniques to access web data', *Information Systems,* vol. 36, pp.117-133.

[43] Liu Y. and He H. (2007) 'Grid Service Selection Using QoS Model', in *Proc. of the 3rd International conference on Semantics, Knowledge and Grid, Xi'an, Shaanxi*, pp. 576-577.

[44] Liu Y., Ngu S. and Zeng L. (2004) 'QoS Computation and Policing in Dynamic Web Service Selection', in *Proc. of the 13th international World Wide Web conference on Alternate track papers & posters.* 19-21 May, pp. 66-73.

[45] Lu H. (2005) 'Semantic web services discovery and ranking', in *Proc. of the IEEE/WIC/ACM international conference on web intelligence*, pp. 157-160.

[46] Maheswari J.U., Karpagam G.R. and Indhumathy S. (2014) 'Comparison of Web Service Similarity Assessment Methods', *International Journal of Computer Applications*, vol. 98.

[47] Martin D., Burstein M., Hobbs J., Lassila O., McDermott D., McIlraith S., Narayanan S., Paolucci M., Parsia B., Payne T., Sirin E., Srinivasan N. and Sycara K. (2004) 'OWL-S: Markup for Web services', *W3C Member Submission 22 November 2004*.

[48] Maximilien E. and Singh M.P. (2004) 'A Framework and Ontology for Dynamic Web Services Selection', *IEEE Internet Computing*, vol. 8, no. 5, pp. 84-93.

[49] Meng L., Huang R. and Gu J. (2013) 'A Review of Semantic similarity Measures in *WordNet*', *International Journal of Hybrid Information Technology*, vol. 6, no. 1.

[50] Miller G. (1990) 'Nouns in *WordNet*: A Lexical Inheritance System', *International Journal of Lexicography*, vol. 3, no. 4.

[51] Mitchell T (1997) 'Machine Learning', *McGraw Hill*.

[52] Mohanty R., Ravi V. and Patra M.R. (2010) 'Web-services classification using intelligent techniques', *Elsevier, Expert Systems with Applications*, vol. 37, pp. 5484-5490.

[53] Nayak R. (2010) 'Data Mining in Web Services Discovery and Monitoring', *International Journal of Web Services Research*, vol. 5, no. 1, pp. 62-80.

[54] Nayak R. and Lee B. (2007) 'Web services discovery with additional semantics and clustering', in *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 555-558.

[55] Oldham N., Thomas C., Sheth A. and Verma K. (2004) 'METEOR-S web service annotation framework with machine learning classification', in *Proc. of the 13th international conference on World Wide Web*, pp. 553-562.

[56] Patwardhan S., Banerjee S. and Pedersen T. (2003) 'Using measures of semantic relatedness for word sense disambiguation', in *Proc. of 4th International Conference on Computational Linguistics and Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico*, pp. 16-22.

[57] Peng Y. (2010) 'Two levels semantic web services discovery', in *Proc. of 7th international conference on fuzzy systems and knowledge discovery*', Y*antai, Shandong*, vol. 6, pp.

2523-2526.

[58] Pennock D., Dave K., and Lawrence S. (2003) 'Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews', in *Proc. of the 12th International World Wide Web Conference (WWW'2003)*, ACM.

[59] Pudil P., Novovicova J. and Kittler J. (1993) 'Automatic machine learning of decision rules for classification problems', in *Proc. of 4th BMVC93*.

[60] Quinlan J. R. (1986) 'Induction of Decision Trees', *Machine Learning, Kluwer Academic Publishers*, vol. 1, pp. 81-106.

[61] Ram S., Hwang Y. and Zhao H. (2006) 'A clustering based approach for facilitating semantic web services discovery', in *Proc. of 15th Annual Workshop on Information Technologies & Systems*, pp.3-8.

[62] Ramu G. and Reddy B.E. (2011) 'A framework for semantic web mining model', *International Journal of Internet Computing*, vol. 1, no. 1, pp. 94-98.

[63] Ran S. P. (2003) 'A Model for Web Services discovery with QoS', *ACM SIGecom Exchanges*, vol. 4, no. 1, ACM Press, pp. 1-10.

[64] Ruiz-Cortés a., Martín-Díaz O., Toro A.D. and Toro M. (2005) 'Improving the Automatic Procurement of Web Services Using Constraint Programming', *International Journal on Cooperative Information Systems*, vol. 14, no. 4, pp. 439-468.

[65] Saha S., Murthy C.A. and Pal S.K. (2008) 'Classification of web services using tensor space model and rough ensemble classifier', in *Proc. of 17th international Symposium on Foundations of Intelligent Systems (ISMIS'08)*, Toronto, Canada, pp. 508-513.

[66] Sahami M., Dumais S., Heckerman D. and Horvitz E. (1998) 'A Bayesian approach to filtering junk e-mail, learning for text categorization', in *Proc. of AAAI Workshop*, pp. 55-62.

[67] Satish N. and Wahidabanu R. S. D. (2012) 'Web Service Mining Based on Annotated Capability Specifications', *European Journal of Scientific Research*, vol. 69, no. 3, pp. 416-427.

[68] Serhani M.A., Dssouli R., Hafid A. and Sahraoui H. (2005) 'A QoS broker based architecture for efficient Web services selection', in *Proc. of the IEEE International conference on web services, IEEE CS*, pp. 113-120.

[69] Sha L. (2009) 'A QoS based Web Service Selection Model', in *Proc. of the International Forum on Information Technology and Applications, Chengdu*, pp. 353-356.

[70] Shafiq O., Alhajj R., Rokne J. G. and Toma I. (2010) 'Light-weight semantics and Bayesian classification: A hybrid technique for dynamic web services discovery', in *Proc. of IEEE International conference on Information Reuse and Integration*, pp. 121-125.

[71] Shehzad K. and Javed M. Y. (2010) 'Multithreaded fuzzy logic based web services mining framework', *European Journal of Scientific Research*, vol. 41, no. 4, pp. 632-644.

[72] Skoutas D., Sacharidis D., Simitsis A. and Sellis T. (2010) 'Ranking and Clustering Web Services Using Multicriteria Dominance Relationships', *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 163-177.

[73] Sotolongo R., Kobashikawa C., Dong F. and Hirota K. (2008) 'Algorithm for web services discovery based on information retrieval using *WordNet* and linear discriminant functions', *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 12, pp. 182-183.

[74] Su Z., Chen H., Zhu L. and Zeng Y. (2012) 'Framework of semantic web services discovery based on fuzzy logic and multi-phase matching', *Journal of Information & Computational Science*, vol. 9, no. 1, pp. 203-214.

[75] Subramaniam T., Jalab H.A. and Taqa A.Y. (2010) 'Overview of textual antispam filtering techniques', *International Journal of Physics Sciences*, vol. 5, no. 12, pp. 1869-1882.

[76] Thomas, L. S., 2008. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, vol. 1(1), pp. 83-98.

[77] Tian M., Gramm A., Naumowicz T., Ritter H. and Schiller J. (2003) 'A Concept for QoS Integration in Web Services', in *Proc. of 4th International Conference on Web Information Systems Engineering*, Rome, Italy, pp. 149-155.

[78] Tian M., Gramm A., Ritter H. and Schiller J. (2004) 'Efficient selection and monitoring of QoS-aware web services with the WS-QoS framework', in *Proc. of IEEE/WIC/ACM international Conference on Web Intelligence*, Beijing, China, pp.152.158.

[79] Tsai Y.H., Hwang S.Y. and Tang Y. (2011) 'A Hybrid Approach to Automatic Web Services Discovery', in *Proc. of International Joint Conference on Service Sciences, IEEEXplore*, pp. 277-281.

[80] Vapnik V. N. (1995) 'The Nature of Statistical Learning Theory', *Springer, New York*.

[81] Vapnik V. N., Druck H. and Wu D. (1999) 'Support vector machines for spam categorization', *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1048-1054.

[82] Vapnik V. (1998) 'Statistical Learning Theory', *John Wiley and Sons*.

[83] Varguez-Moo M., Moo-Mena F. and Uc-Cetina V. (2013) 'Use of classification algorithms for semantic web services discovery', *Journal of Computers*, vol. 8, no. 7, pp. 1810-1814.

[84] Voorhees E. (1994) 'Query Expansion Using Lexical-Semantic Relations', in *Proc.of ACMSIGIR, ACM/Springer, Dublin, Ireland*, pp. 61-69.

[85] Wang H., Shi Y., Zhouy X. and Zhou Q. (2010) 'Web service classification using support vector machine' in *Proc. of IEEE 22nd International Conference on Tools with Artificial Intelligence*, vol. 1, pp. 3-6.

[86] Wang X., Vitvar T., Kerrigan M. and Toma I. (2006) 'A QoS-Aware Selection Model for Semantic Web Services', in *Proc. of ICSOC 2006*, pp. 390-401.

[87] Wang Y.Q. (2001) 'Theory and method of artificial intelligence', *Press of Xi'an Jiao Tong University*.

[88] Web service List on http://seekda.com

[89] Web Service List, http://www.webservicelist.com.

[90] Weider D.U., Rachana B.R., Pingali S. and Kolluri V. (2007) 'Modeling the Measurements of QoS Requirements in Web Service Systems', *Simulation*, vol. 83, no. 1, pp.75-91.

[91] Witten H. and Frank E. 2005 'Data Mining Practical Machine Learning Tools and Techniques', *Second Edition, Morgan Kaufmann Publishers, San Francisco*.

[92] Wongpun S. and Srivihok A. (2008) 'Comparison of Attribute Selection Techniques and Algorithms in Classifying Bad Behaviours of Vocational Education Students', in *Proc. of 2nd IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST), Australia*, pp. 526-531.

[93] Wu C., Chang E. and Aitken A. (2008) 'An empirical approach for semantic web services discovery', in *Proc. of 19th Australian Conference on Software Engineering (aswec 2008)*, pp. 412-42.

[94] www.andreashess.info/projects/annotator/ws2003.html

[95] XMethods, http://www.xmethods.net, accessed February 2012.

[96] Yan J. and Piao J. (2009) 'Towards QoS-based Web Service Discovery', in *Proc. of the International Conference on Service Oriented Computing, Sydney*, pp. 200-210.

[97] Yang Y. and Pedersen J. (1997) 'A comparative study on feature selection in text categorization', in *Proc. of* 14th *International Conference on Machine Learning (ICML), 08 – 12 July 1997, Morgan Kaufmann Publishers Inc. San Francisco, CA*, pp.412-420.

[98] Yang Y. (1997) 'An evaluation of statistical approaches to text categorization', *Technical Report CMU-CS-97-127, Carnegie Mellon University*.

[99] Ying L. (2010) 'Algorithm for semantic web services clustering and discovery', in *Proc. of International Conference on Communications and Mobile Computing*, vol. 1, pp. 532-536.

[100] Yuan-jie L. and Jian C. (2012) 'Web service classification based on automatic semantic annotation and ensemble learning', in *Proc. of IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pp. 2274-79.

[101] Zeng L. Z. and Benatallah B. (2004) 'QoS-aware middleware for Web services composition', *IEEE Transaction on Software Engineering*, vol. 30, no. 5, pp. 311-327.

[102] Hou J., Zhang T., Hui M., Xiao L., Chen G. and Li D. (2008) 'Web services discovery based on keyword clustering and ontology', in *Proc. of IEEE International Conference on Granular Computing*, vol. 1, pp. 844-848.