A Framework for Semantic Web Services Discovery

A Thesis submitted

for the award of degree of

Doctor of Philosophy

by

Shalini Batra

(90603503)

under the Guidance of

Dr. Seema Bawa

Professor, Computer Science and Engineering Department,

Dean of Student Affairs,

Thapar University, Patiala-147004, INDIA

Computer Science and Engineering Department

Thapar University, Patiala -147004, INDIA

April, 2012

Certificate

I hereby certify that the work which is being presented in this thesis entitled "**A Framework for Semantic Web Services Discovery**", in partial fulfillment of the requirement for the award of degree of "Doctor of Philosophy" submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa 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.

(**Shalini Batra**)

Regn. No. 90603503

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

(**Dr. Seema Bawa**)

Professor

Department of Computer Science Engineering

Dean of Student Affairs

Thapar University

Patiala, 147004

Punjab, INDIA.

Dedicated To

Almighty

and

My dearest **Guruji**

Sh. Vyas Dev Ji

Acknowledgements

I would like to express my sincere appreciation to my supervisor, **Prof. Seema Bawa**, for being the pillar of support and encouragement throughout my research work. Her 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 my PhD committee members, **Dr. Maninder Singh**, Associate Professor and Head, Computer Science and Engineering Department (CSED), Thapar University, Patiala, and **Dr. Rajesh Kumar**, Associate Professor, School of Mathematics and Computer Applications (SMCA), Thapar University, Patiala, for their constructive comments and regularly ensuring the progress of my research work. I am thankful to all the **faculty** and **staff** members of CSED for their support.

I am highly grateful to **Dr. R. K. Sharma**, Professor, School of Mathematics and Computer Applications (SMCA), Thapar University, Patiala and **Dr. R. K. Bawa**, Associate Professor, Department of Computer Science and Engineering, Punjabi University, Patiala, for sparing their precious time and providing me constructive comments.

I also acknowledge the co-operation and encouragement extended to me by my students at Thapar University, especially **Mr. Ravi Maggon**, **Mr. Dharya Arora** and **Mr. Sanjay Madan**.

I offer my deepest gratitude to my mother, **Smt. Sarla Devi Thakur** and my sisters especially **Kiran Didi** and **Nidhi Didi**, for their love, encouragement and mo

tivation and for their confidence in me.

I am thankful to my husband **Mr. Vivek Batra**, and my loving son **Prikshit Batra** for their unfailing motivation, as without their support, I won't have started this work.

The chain of gratitude would be definitely incomplete without thanking my **Guruji Sh. Viyas Dev Ji** and the **Almighty**, the prime movers, for inspiring and guiding me (the humble being) to complete this task successfully.

Patiala

September, 2011

(Shalini Batra)

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Web services in their current form do not express what a service actually does, *i.e.* they are limited in their ability to express the capabilities of a service. This limitation may be relaxed by providing a rich set of semantic annotations that augment the services description. Semantic descriptions of Web services are necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains.

In this thesis, we present a novel approach for the automatic discovery of Web services by categorizing the service using some semantic and statistical measures. A framework named *'ADWebS'* (Automatic Discovery of Web Services Semantically) has been proposed and designed for automatic discovery and categorization of Web services. Categorization process in *ADWebS* starts with incorporation of semantic description in the form of annotations by services publisher or developer, which are extracted from WSDL of annotated services. Semantic relatedness of each annotation or term is found with every candidate category using Normalized Google Distance (NGD). Next a non-parametric test Kruskal Wallis is applied and based on the median value achieved for each category, a service is permanently assigned to one or more pre-defined category(s).

ADWebS provides a 'Google-like' query interface to the service discoverer with a list of available candidate categories as a dropdown list. Performance of the *ADWebS* has been tested for service retrieval time, precision-at-n and scalability.

Chapter 1

Introduction

Service Oriented Architecture (SOA) is a technology architectural model for service-oriented solutions, with distinct characteristics in support of realizing service-orientation and the strategic goals associated with service-oriented computing. It is an architecture for the development of loosely coupled distributed applications. Basically, it is collection of many services in the network which communicate with each other and the communications involves data exchange and even service coordination.

Earlier SOA was based on the DCOM or Object Request Brokers (ORBs). Now a days SOA is based on the Web Services. Broadly SOA can be classified into two terms: Services and Connections.

Services:

A service is a function or some processing logic or business processing that is well-defined, self-contained, and does not depend on the context or state of other services. Example of services are Loan Processing Services, which can be self-contained unit for process the loan applications or a Weather Services, which can be used to get the weather information. Any application on the network can use the service of the weather service to get the weather information.

Connections:

Connections means the link connecting these self-contained distributed services with each other, it enable clients to services communications.

There are three common actions associated with a services in SOA :-

- Discovery,
- Request,
- Response

Discovery is the process of finding the services that provide the required functionality. A *Request* provides the input to the service. The *Response* yields the output from the services.

Technologies Used:

SOA is much different from point-to-point architectures. SOA comprise loosely coupled, highly interoperable application services. These services can be developed in different development technologies (such as Java, .NET, C++, PERL, PHP), the software components become very reusable i.e. the same C (C Sharp) service may be used by a Java application and / or any other programming language. WSDL defines an standard, which encapsulates / hides the vendor / language specific implementation from the calling client / service.

A service in SOA does not necessarily mean a Web service. However, the Web services technology is a well-known implementation of services in SOA. An overview of Web services, its lifecycle, features and stack is provided in the coming sections.

1.1 Web Services

The English word 'service' is overloaded in its meaning. In the business world, a 'service' normally denotes the provision of a general business activity which provides a certain value to the customer [205]. In computer science, in contrast, the term 'service' is often used synonymously with 'Web service', *i.e.* a software component accessible over the Internet via a standardized interface.

According to Preist [34]:

Service: A service is the provision of a concrete product or abstract value in some domain. The provisions of the service as such, and the contractual issues around service provision, are independent of how the supplier and the provider interact.

Web services: Web services are computational entities accessible over the Internet (using Web services standards and protocols) via platform- and programming-language independent interfaces.

Web services are the amalgamation of eXtensible Markup Language (XML) and HyperText Transfer Protocol HTTP that can convert your application into a Web-application, which publish its function or message to the rest of the world.

According to W3C

"Web Services are the message-based design frequently found on the Web and in enterprise software. The Web of Services is based on technologies such as HTTP, XML, SOAP, WSDL, SPARQL, and others."

In other words, we can say, Web services are just Internet Application Programming Interfaces (API) that can be accessed over a network, such as Internet and intranet, and executed on a remote system hosting the requested services.

Web services constitute a distributed computer architecture made up of many different computers trying to communicate over the network to form one system. They

consist of a set of standards that allow developers to implement distributed applications - using radically different tools provided by many different vendors - to create applications that use a combination of software modules called from systems in disparate departments or from other companies.

A Web service contains some number of classes, interfaces, enumerations and structures that provide black box functionality to remote clients. Web services typically define business objects that execute a unit of work (e.g., perform a calculation, read a data source, etc.) for the consumer and wait for the next request. Web service consumer does not necessarily need to be a browser-based client. Console-based and Windows Forms-based clients can consume a Web service. In each case, the client indirectly interacts with the Web service through an intervening proxy. The proxy looks and feels like the real remote type and exposes the same set of methods. Under the hood, the proxy code really forwards the request to the Web service using standard HTTP or optionally SOAP messages.

Web Services are built on open standards and any platform can invoke them from any other platform. They are not only available in public domain, but can also exist on an internal network for internal applications. Thus it can be said that Web services are available 'over intranets, extranets, and the Internet'.

1.1.1 Features of Web Services

Some of the important features of Web Services [45] are:

i) Loose coupling: Every service is atomic, self-describing, accessible, declarative, stateless and composite.

ii) Contracted: Services in an SOA are represented by a contract that describes

its inputs, outputs, access policies, quality-of-service requirements, and error-handling procedures.

iii) Discoverable: Services can be easily discovered via registries at the time of execution.

iv) Addressable: Services are uniquely identifiable in a network.

v) Distributed: Services in SOA are separated by geographic and machine boundaries.

Web Service Reference Model

The common usage scenario for Web services as seen in Figure 1.1 can be defined by three phases; Publish, Find, and Bind; and three entities: the service requester, which invokes services; the service provider which responds to requests; and the registry where services can be published or advertised [124].

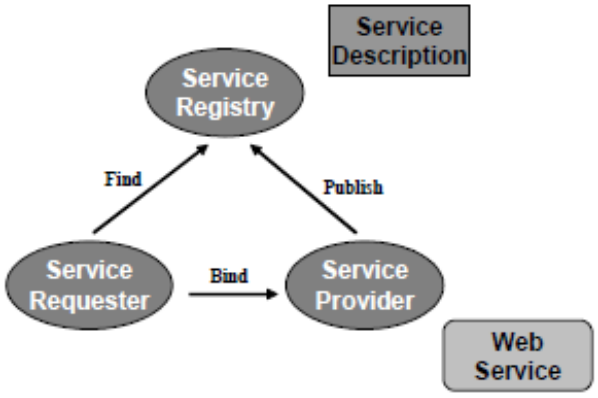

Figure 1.1: Web Services Reference Model [124]

A service provider publishes a description of a service it provides to a service registry. This description (or advertisement) includes a profile on the provider of the service (e.g.

company name and address); a profile about the service itself (e.g. name, category); and the URL of its service interface definition (i.e. WSDL description).

When a developer realizes a need for a new service, he finds the desired service either by constructing a query, or browsing the registry. The developer then interprets the meaning of the interface description (typically through the use of meaningful label or variable names, comments, or additional documentation) and binds to (i.e. includes a call to invoke) the discovered service within the application they are developing. This application is known as the service requester. At this point, the service requester can automatically invoke the discovered service (provided by the service provider) using Web service communication protocols [124].

1.1.2 Web services Stack

Figure 1.2 provides an overview of role of all the layers in the Web services stack.

Figure 1.2: Web Services Stack

Service Negotiation

The business logic process starts at the Services Negotiation layer (the top) with, say, two trading partners negotiating and agreeing on the protocols used to aggregate Web Services. This layer is also referred to as the Process Definition Layer covering document, workflow, transactions, and process flow.

Workflow, Discovery, Registries

The stack then moves to the next layer to establish workflow processes using WSDL or Web Services Flow Language (WSFL), and MS XLANG, an XML-based language to describe workflow processes and spawn them. WSFL specifies how a Web Service is interfaced with another. With it, one can determine whether the Web Services should be treated as an activity in one workflow or as a series of activities. While WSFL complements WSDL (Web Services Definition Language) and is transition-based, XLANG is an extension of WSDL and block structured-based.

Web Services that can be exposed may, for example, get information on credit validation activities from a public directory or registry, such as Universal Description, Discovery and Integration (UDDI). The ebXML, E-Services Village, BizTalk.org, and xml.org registries, and Bowstreet's (a stock service brokerage) Java-based UDDI (jUDDI) are other directories that could be used with UDDI in conjunction with Web Services for business-to-business (B2B) transactions in a complex EAI infrastructure under certain conditions [256].

Hewlett Packard Company, IBM, Microsoft, and SAP launched beta implementations of their UDDI sites that have conformed to the latest specification (UDDI v2), including enhanced support for deploying public and private Web Service registries, and the interface (SOAP/HTTP API) that the client could use to interact with the registry server. In addition to the public UDDI Business Registry sites, enterprizes

can also deploy private registries on their intranet to manage internal Web Services using the UDDI specification. Access to internal Web Service information may also be extended to a private network of business partners.

Service Description Language

As one moves down the stack, one needs WSDL to specify how to connect to a Web Service. This language is an XML format for describing network services. With it, service requesters can search for and find the information on services via UDDI, which, in turn, returns the WSDL reference that can be used to bind to the Web Service.

Web Service Conversational Language (WSCL) helps developers use the XML Schema to better describe the structure of data in a common format (say, with new data types) the customers, Web browsers, or indeed any XML enabled software programs can recognize. This protocol can be used to specify a Web Service interface and to describe service interactions [256].

Messaging

Next is the Messaging Layer in the stack where SOAP acts as the envelope for XML-based messages, covering message packaging, routing, guaranteed delivery and security. Messages are sent back and forth regarding the status of various Web Services as the work progresses.

Transport Protocols

When a series of messages completes its rounds, the stack goes to its last layer: the transport layer, using Hypertext Transfer Protocol (HTTP), Secure HTTP (HTTPS), Reliable HTTP (HTTPR), File Transfer Protocol (FTP), or Standard Mail Transfer Protocol (SMTP). Then, each Web Service takes a ride over the Internet to provide a service requester with services or give a status to a service provider or broker.

Business Issues

Finally, the Business Issues row in the table lists other key areas of importance to the use and growth of Web Services.

Advantages of using Web Services

i) Web services, for the first time have broad support in joint standardization efforts by the global players instead of previous vendor-specific solutions (DCOM, **etc**.).

ii) They are not an all-in-one framework to solve it all (like CORBA), rather these are a family of (a) simple, (b) modular, (c) interrelated, and (d) extensible standards.

iii) They rely on and are built upon Web protocols such as HTTP, thus allowing the use of widely supported technologies with a large community of developers and users behind them.

iv) All Web services technologies are based fundamentally on XML as the exchange and message format of choice.

1.1.3 Web Service Lifecycle

Web Services includes the following main steps:

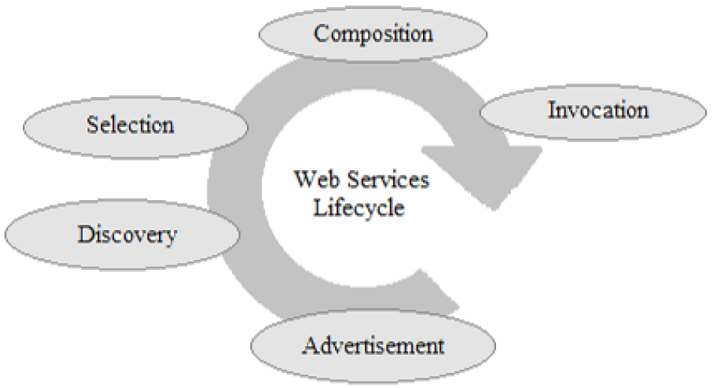

Figure 1.3: Web Services Lifecycle [92]

Advertisement: Service advertisement is a process of publishing the description and endpoints of Web services in a service registry. Services providers describe their Web Services and advertise them in a universal registry. This enables services requestors to search the registry and find services in registries that are accessible over the Web.

Discovery: It a process of locating all Web services that match the requestor's functional requirements. If the requester entity wishes to initiate an interaction with a provider entity and it does not already know what provider agent it wishes to engage, then the requester entity may need to 'discover' a suitable candidate. Given the URL to a discovery document residing on a Web server, a developer of a client application can learn where a Web service exists, what its capabilities are, and how to properly interact with it.

Selection: Services selection is a process of selecting the most suitable Web Service out of the discovered ones, usually based on application-dependent metrics (e.g., QoS). Once the list of services offering the required functionality is available, set of steps (instructions, rules) for Web Services selection should be followed to determine the component to handle the client's request.

Compositions: As the name suggests, this particular term does not apply to a single Web service, but to a composite relationship between a collection of services. Any service can enlist one or more additional services to complete a given task. Further, any of the enlisted service can call other services to complete a given sub-task. Therefore, each service that participates in a composition assumes an individual role of service composition member. It is a process of integrating selected Web Services into a complex process.

Invocation: Invoking a Web service refers to the actions that a client application

performs to use the Web service. It is a process of invoking a single Web service or complex process, by providing it with all the necessary inputs for its execution. Client applications that invoke Web services can be written using any technology for *e.g.* Java, Microsoft .NET,*etc.*

Considering all the five steps in Web services lifecycle, Web Services discovery is the most important step because if a well designed Web service is not discovered by a client or user, what ever functionalities it may provide, it is of no use.

W3C clearly state that

"Discovery is the act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions".

In this thesis we focuses on development of a framework for semantic discovery of Web services.

1.2 Web Services Discovery

Three factors mainly affecting the service discovery are [92]:

i) Ability of service providers to describe their services,

ii) Ability of service requestors to describe their requirements, and

iii) 'Intelligence' of the service matchmaking algorithm.

1.2.1 Registering and Discovering Web Services

A directory service serves as information point for components and participants (e.g. applications, agents, web services, people, objects, requestors and procedures) which

enables them to locate each other. Directories organize and provide Web services location and description details and publish them to any requestor. There are two kinds of directories available: name servers referred to as "white pages" in which entries are defined and found by their name as well as so called "yellow pages", where entries are defined and found by their characteristics and functions [149].

Two main standards for directories have been defined: ebXML (Electronic Business using XML) [56] registries and UDDI (Universal Description, Discovery and Integration) registries.

Electronic Business Extensible Markup Language (ebXML)

ebXML is used as a global electronic market place where enterprises of any size, anywhere can determine each other electronically and clear semantics by exchanging XML messages. This exchange is based on mutual trading protocol agreements. ebXML Registry's goal is to provide generic, extensible, secure and federated information management. ebXML architecture enables to define business processes and their related messages. It gives a way to register and discover business process sequences. Moreover, ebXML allows defining company profiles and a uniform message transport layer [56].

Universal Description, Discovery and Integration (UDDI)

Singh [149] defines UDDI as "a platform-independent, Extensible Markup Language (XML)-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet". UDDI supports three kinds of services descriptions: white-pages, yellow-pages, and green-pages services. White pages contains the following information fields:

- Business name.
- Text description: a list of Multilanguage text strings.
- Contact information: names, phone numbers, and web sites.

Yellow pages provide the business categories organized as the following major taxonomies:

- Industry, a six-digit code for classifying companies.
- Products and services.
- Geographical location for countries and region code.

Green pages contain the information business used to describe how other businesses can conduct electronic commerce with them. It is a nested model comprising business processes, service descriptions, and binding information.

Data structures describe details about organizations (for example their services, implementation technologies, relationships to other businesses).

UDDI defines five principal data structures [149]:

- A businessEntity represents the business or organization that provides the Web services.
- A businessService represents a Web service.
- A BindingTemplate represents the technical binding of a Web services to its access point, its URL and to tModels.
- A tModel represents a specific kind of technology, such as SOAP or WSDL.
- A publisher Assertion represents a relationship between two business entities.

There are two kinds of APIs that allow programmatic access to a UDDI registry: firstly the inquiry API which is used to retrieve information from a registry and, secondly, the publish API which is used to store information in a registry. In contrast with the inquiry API, the publish API needs authenticated access.

ebXML vs UDDI

Table 1.1 provides comparative analysis of the two standards.

Table 1.1: Comparative analysis of UDDI and ebXML Registry [149]

UDDI Registry	**ebXML Registry and Repository**
Contains no repository. Unable to store content. Capable only of storing metadata about (or pointers to) content.	Has an integrated Registry and Repository. Able to store content as well as metadata.
Design center lists businesses and services, similar to yellow/white pages	Design center provides secure, federated information management of any type of artifact.
Protocols and information model is focused and specific.	Protocols and information model is generic and extensible.
Supports multi-registry topologies using replication of every transaction to all participating registries	Supports multi-registry topologies using loosely coupled federations with optional selective replication.

1.2.2 RESTful Web Services

REST the abbreviation of 'Representational State Transfer' designates an architecture style used to create networked applications. The terms "representational state transfer" and 'REST' were introduced in 2000 in the doctoral dissertation of Roy Fielding [201], one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification. REST uses a stateless, client-server, cacheable communications protocol which is almost always the HTTP protocol. Its original feature is to work by using mere HTTP to make calls between machines instead of choosing complex mechanisms such as CORBA, RPC or SOAP [137].

In many ways, the World Wide Web itself, based on HTTP, can be viewed as a REST-based architecture. Many aspects of the Internet correspond to the characteristics of a REST-based architecture. Restful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations [137].

REST does not offer security features, encryption, session management, QoS guarantees, etc. But these can be added by building on top of HTTP, for example username/password tokens are often used and for encryption, REST can be used on top of HTTPS (secure sockets).

RESTful Resources

Characteristic elements of RESTful technology are sources of specific information which are referred to as resources. Each of them is linked to a global identifier, for example a URI in HTTP. These resources are accessed by components of the network (user agents and servers) which communicate through a standardized protocol (e.g., HTTP) and exchange content (representations) of these resources. In order to interoperate with a resource, an application must possess both the resource's identifier and the required method.

There is no need to know the services implementation and system configuration, i.e. whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between the application and the server which hosts the resources. However, the application must be capable of interpreting the data format (representation) returned from the resource, which is often an HTML or XML document, though it may also be an image, plain text, or any other content [258].

RESTful Messages types

Restful allows the realization of applications without requiring any new format.

Resources can be represented by any format (i.e. HTML, GIF and PDF files). XML can be used to transmit structured data.

RESTful Principles

The REST architectural style is based on four principles [33]

- Resource identification through URI. Resources are identified by URIs which provide a service discovery mechanism.
- Uniform interface. A fixed set of four create, read, update, delete operations is responsible for the manipulation of resources.
- Self-descriptive messages. Resources' content can be presented in several formats (e.g. HTML, XML, plain text, PDF or JPEG). Metadata about the resource can be used to detect transmission errors and perform authentication or access control.
- Stateful interactions through hyperlinks. Stateful interaction can be achieved using several techniques, for example rewriting URL, cookies or hidden form fields.

Documenting REST Services: WSDL and WADL

WSDL, a W3C recommendation, is commonly used to spell out in detail the services offered by a SOAP server. While WSDL is flexible in service binding options (for example, services can be offered via SMTP mail servers), it did not originally support HTTP operations other than GET and POST. Since REST services often use other HTTP verbs, such as PUT and DELETE, WSDL was a poor choice for documenting REST services.

With version 2.0, WSDL supports all HTTP verbs and it is now considered to be an acceptable method of documenting REST services.

The second alternative is the Web Application Description Language (WADL). Like the rest of REST, WADL is lightweight, easier to understand and easier to write than WSDL. In some respects, it is not as flexible as WSDL (no binding to SMTP servers), but it is sufficient for any REST service and much less verbose.

The main advantages of REST web services are:

- Lightweight - not a lot of extra XML markup
- Human Readable Results
- Easy to build - no toolkits required

SOAP also has some advantages:

- Easy to consume- many times
- Rigid - type checking, adheres to a contract
- Development tools-Easily Available

The main advantage of SOAP-based SOA over REST is the more mature tool support. Another SOAP advantages include the type-safety of XML requests (for responses, ROA can also use XML if the developers desire it).

For consuming Web services, its sometimes a toss up between which is easier. For instance Google's AdWords Web service is really hard to consume, it uses SOAP headers, and a number of other things that make it kind of difficult. On the converse, Amazon's REST web service can sometimes be tricky to parse because it can be highly nested, and the result schema can vary quite a bit based on what you search for.

1.3 Problem Areas

Web services allow easy adaption and integration by deploying machinery of XML, providing a standards-based framework for exchanging information dynamically between applications. Industry efforts to standardize Web services description, discovery and invocation have led to standards such as WSDL or WADL, UDDI, SOAP, *etc.* In this section, we focus on two major problems of Web services technology that limit the extraction of functional capabilities of the services.

1.3.1 Web Service description

Web services in their current form, are designed to represent information about the interfaces of services, how they are deployed, and how to invoke them. They do not express what a service actually does, *i.e.* they are limited in their ability to express what the capabilities and requirements of services are. For instance, if services use different XML tags to annotate their operations/ messages, then it is not possible to find out that two operations offered by different services in different WSDLs actually perform the same function.

It is difficult to find a direct mapping from this kind of query to the service capability attributes addressed in a service description based on the current Web service description standards.

Ideally, a Web service should be completely autonomous so that it can be discovered automatically by software agents or other services. To be completely autonomous, a better service description solution is required. So, an extra step is required to gather sufficient information from the service requester in order to perform autonomous matchmaking.

Need is to provide machine-interpretable and inferenceable meaning within the

services which can lead to easy accessability of functional capabilities of the available Web services.

1.3.2 Web Service Discovery

The service discovery approaches in their current form suffer from some shortcomings, due to the service description technologies applied and the matching algorithms used. As there is no standard convention or rule to define a service, each service provider has different method of providing detail for the service. This increases the difficulty in service discovery and matching.

Current Web service discovery mechanisms in their existing form are limited in their search capabilities due to their inability to extend beyond the keyword-based matches. Since they do not capture the relationships between entities in its directory they are not capable of making use of the semantic information to infer relationships during search.

Such methods support search based only on the high-level information specified about businesses and services and do not get to the specifics of the capabilities of services during matching. The current search facilities support only direct matches. In cases where no direct matches are available but a set of services can be composed to fulfill a request, registries like UDDI fails to provide any search results because it does not look beyond direct matches. The demand for automation is increasing, which requires a service registry supporting automatic service discovery.

As the number of services is getting large, manually searching services is becoming more and more time consuming and inefficient. One of the possible solution is to categorize all services according to specific domains according to their functional description.

Categorizing a service will require creation of a suitable semantic similarity calculation method. The semantic similarity calculation between a service query and a service description is again a very important component which needs to be studied thoroughly. The complexity of the service semantics requires a dedicated method to precisely and effectively compute the semantic similarity.

Another important aspect which should be considered is how to build a mapping between service queries and service descriptions. According to Du *et al.* [260] in real business scenarios, service users usually propose general service queries to describe what they need because they are not aware of the technical detail of services.

1.4 Possible Solutions

In the previous section, we present an overview of the two major problem areas of Web services. In this thesis, we will discuss in details how we address these issues and provide a research solution. This section outlines in brief the directions provided by research community to overcome the problems discussed above. A detailed survey of the approaches used for service discovery has been provided in Chapter 2.

Adding semantics to represent the requirements and capabilities of Web services is essential for achieving unambiguous and machine-interpretable services. Semantic models provide agreement on the meaning and intended use of terms, and may provide formal and informal definitions of the entities, and there will be less ambiguity in the intended semantics of the provider. It has been clearly stated by Paolucci [151] that first step towards solving the services location problem is to 'rise above these superficial differences in the representation of interfaces of services and to identify the semantic similarities between them'.

1.4.1 Semantics in service description

Actual problem is that Web services lack semantic descriptions in their current format. This lack of semantic representation leads to disparities in service specifications by service providers and requesters for similar service in a given industry. This definitely leaves the promise of automatic integration of applications written to Web services standards unfulfilled. If no semantics are specified, a service requester may not be able to find services provider due to the superficial differences in interface specifications even if it was a suitable match.

It has been clearly stated that there is a relationship between a developer's effort and the benefit user's obtain from a discovery system. The *more effort developers put on describing their services and describing their service needs, the more accurate discovery results they would obtain.*

Many efforts have been made to extract the information available in the Web service description file in the form of comment, documentations, etc. Additional information in the form of annotations can be added during development stage as the services providers can explicate the intended semantics by annotating the appropriate parts of the Web services with concepts from a richer semantic model.

There are two ways of creating semantic annotated Web services. One way is to create an independent semantic Web service description framework and link it to the current Web services standards. The leading research efforts in this direction are OWL-S [48] and WSMO [91]. The other way is to add semantic annotations into the current Web services standards. The major research work in this way is WSDL-S [184].

There have been many proposals for Web services discovery based on OWL ontologies. Many researchers have proposed the usage of ontology [12], [61], [118], [127], [151], to annotate the elements in Web services.

Though ontology-based approaches can result in accurate semantic matching capabilities, there are certain drawbacks as well. Major problem is that ontology creation, integrating, mapping, matching, maintenance, *etc.* involve a huge amount of human effort. Further ontologies are limited to specific domains while Web services discovery mechanism should be domain independent.

1.4.2 Semantic Service Discovery

As Web Services discovery is an important and difficult task in the development cycle of service oriented application, many algorithms, tools and mechanisms are proposed to solve this problem.

Various proposals include initiatives, projects and languages such as WSMO [91], METEOR-S [122], OWL-S [48]and SWSA/SWSL have come up for adding semantics in Web services. It is observed that all these approaches use ontology as a base for mapping and matching different type of services.

Researchers have proposed a number of ideas to enhance the accuracy of Web services discovery by applying data mining approaches [196], [195], singular vector decomposition [17], graph based methods, various ontology based discovery frameworks, agent- based, logic based methods and others [113], [55], [3].

Service categorization for easy service discovery

As the number of services are growing manifold, one of the possible option is to categorize the services into functional domains. Current techniques for filtering based on a taxonomic organization of service categories require discoverers to manually determine the proper category of a service and then browse services within that category.

Yellow pages in UDDI provide a classification of the services or business, based on standard taxonomies. These include the Standard Industrial Classification (SIC), the

North American Industry Classification System (NAICS) [166] or the United Nations Standard Products and Services Code (UNSPSC) [245] which help in classifying businesses and services in a standard, universally-known way. The UDDI Type Model, or tModel, was devised to categorize Web services and is an abstraction for a technical specification of a service type that organizes its information and makes it accessible [232]. Unfortunately, it is not always defined by creators of Web services when published in UDDI and is thus rarely used. The API for UDDI registries thus supports only simple keyword-based matching, usually resulting in unsatisfactory precision and recall.

1.5 Objectives

Based on problems in the current Web services discovery scenarios (Section 1.3) and the current approaches (Section 1.4) following objectives have been laid down for our thesis work

(i) To study, analyze and explore the current applications and utility of Web services.

(ii) To design and develop a framework for Semantic Web Services Discovery.

(iii) To develop and deploy Semantic Web Services.

(iv) To test and validate the Semantic Web Services on the framework using testing parameters like scalability and service retrieval time.

1.6 Proposed Method

1.6.1 Service description through annotation

To provide discovery of Web services without the addition of ontologies, a framework named 'ADWebS' (Automatic Discovery of Web Services Semantically) has been proposed. A new tag called '<annotation/ >' has been introduced in the existing WSDL file of every Web service where semantic metadata is added to this tag by the service provider (developer). Thus, before publishing the Web services in a registry, it is mandatory for service developer to add a set of terms (n $\geq$ 1) which provide the functional description of the available services.

1.6.2 Service discovery through categorization

One of the option for finding semantic relatedness between services is corpus based matchmaking where WordNet, Wikipedia or search engines like Google or Yahoo can be used as knowledge corpus.

ADWebS exploits the metadata provided by the service publisher and add a Web service into one or more candidate category(s). To deduce the category of a given service, semantic relatedness of each term in found with all the candidate categories using Normalized Google Distance (NGD), one of the most prevalent Measure of Semantic Relatedness (MSR) and a term category matrix is generated. Next, a statistical measure, Kruskal Wallis, a non-parametric test is applied and based on the results achieved a particular Web service is allocated to one of the predefined category(s).

A Google like interface is provided to the user that allows him to ask for a service by providing a list of categories and text box for query. "Google-like" query interface, relieves user from learning another query language. Thus the entire problem of finding

relevant services is reduced to looking for similar services within a category. Basically, we aim at allowing a developer to state a query using simple English language. The goal is to exempt service user from making an additional effort for searching services by transparently pulling out annotations from their service-oriented application source code. Thus, ADWebS allows service discoverer to ask for a third-party service by providing a natural language description, or a handful set of keywords, of its expected functionality.

1.7 Thesis Contribution

Major contributions of the thesis are:

i) ADWebS (Automatic Discovery of Web Services Semantically) propose semantic discovery of Web services by providing semantic at two stages:

   a) Providing annotations has been made mandatory for Web service publisher by introducing 'annotations' tag in the WSDL of the Web service.

   b) Normalized Google Distance (NGD), a Measure of Semantic Relatedness is used to find semantic association between user defined tags and pre defined categories.

ii) It does not require any language dependent pre-processing steps such as part-of-speech tagging or dependency parsing, which can be time consuming or even infeasible at very large scale.

iii) Most frequently used and most reliable Web search engine 'Google' has been used to measure the semantic relatedness.

iv) Most important contribution of ADWebS is that it follows incremental approach, i.e. it proposes an enhancement in the existing Web services standards used for Web service discovery.

v) A Google like interface is provided to the user that allows him to search for a service by providing a list of categories and text box for query.

1.8 Thesis Organization

The thesis is organized as follows:-

Chapter 2: Approaches and Frameworks for Web Services Discovery: This chapter provides the comprehensive survey of work done in the area of Web Service Discovery along with a set of dimensions which are used as benchmark to judge the performance of services mechanisms. A comparative study and analysis of the existing Web services frameworks has also been provided in this chapter.

Chapter 3: Annotations in '*ADWebS*': The proposed framework 'Automatic Discovery of Web services' named '*ADWebS*' is broadly divided into five steps. This chapter discusses first two steps of the proposed framework in detail. This chapter also provides the algorithms and analysis of the algorithms for comparative analysis of three approaches: i) Simple word matching approach, ii) The WordNet approach and iii) The Normalized Semantic Similarity Approach (The proposed approach)

Chapter 4: Semantic Categorization and Discovery in '*ADWebS*': This chapter provides the details of last three steps of the proposed framework. Approach used to classify a Web service into pre-defined candidate category(s) using statistical measures and ranking of service according to user's query are discussed in this chapter.

Chapter 5: Experiments and Testing: This chapter provides the implementation

details and discusses various evaluation parameters used for measuring the performance of the proposed framework. The front end provided to the Web service discoverer or the end user is also presented in this chapter.

Chapter 6: Conclusion and Future Scope: Thesis concludes with this chapter by providing a brief overview of the proposed framework and providing a insight into the future scope of the work.

Chapter 2

Approaches and Frameworks for Web Services Discovery

The Web is a distributed, dynamic, and large information repository. It has now evolved to encompass various information resources accessible worldwide. Organizations across all spectra have already moved their main operations to the Web, which has brought about a fast growth of various Web applications. This has dramatically increased the need to build a fundamental infrastructure for efficient deployment and access of the exponentially growing plethora of Web applications. The development of enabling technologies for such an infrastructure is expected to change the business paradigm on the Web [113].

2.1 Web Services

Web service is a piece of software application whose interface and binding can be defined, described, and discovered as XML artifacts [76]. It supports direct interactions with other software agents using XML-based messages exchanged via Internet-based

protocols.

Web services have *de facto* become the most significant technological by-product of World Wide Web. Different definitions about Web services are given by different industry leaders, research groups, and Web services consortia. For example,

The W3C consortium defines a Web service as '*a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards*' [254].

IBM defines Web services as *'self-describing, self-contained, modular applications that can be mixed and matched with other Web services to create innovative products, processes, and value chains. Web services are Internet applications that fulfill a specific task or a set of tasks that work with many other Web services in a manner to carry out their part of a complex workflow or a business transaction'.*

According to Microsoft, *'A Web Service is a unit of application logic providing data and services to other applications. Applications access Web Services via ubiquitous Web protocols and data formats, such as HTTP, XML, and SOAP, with no need to worry about how each Web Service is implemented'.*

HP defines Web services as *'modular and reusable software components that are created by wrapping a business application inside a Web service interface. Web services communicate directly with other Web services via standards-based technologies'*

SUN perceives a Web service as '*an application functionality made available on the World Wide Web. A Web service consists of a network- accessible service, plus a formal description of how to connect to and use the service.*'

The aforementioned definitions give a high-level description of the major objective and supporting technologies of Web services. Interoperation among machines is the major design goal of Web services. By providing a standards-based framework for exchanging information dynamically on demand between applications, Web Services show promise to address the information integration needs of enterprise application integration [183].

Tsalgatidou *et al.* present the basic Web services model and sketch the Web services lifecycle in their work [242]. They present an overview of the key Web services standards, including SOAP, WSDL, UDDI, and ebXML. Medjahed *et al.* reviewed Web services technologies from a business-to business (B2B) application perspective [25]. Their work outlines the major features of Web services and examines how they could fit into the B2B interaction environment. Services composition is presented as a powerful tool to combine inter-enterprise applications and enhance the B2B interactions. Papazoglou *et al.* provide a review of the Web services technologies as an application of service-oriented computing [150].

2.1.1 Benefits of using Web Services

Exposing the function on to network:

A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows one to expose the functionality of existing code over the network. Once it is exposed on the network, other application can use the functionality of the program.

Connecting Different Applications:

Web Services allows different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the Web

services. For example VB or .NET application can talk to Java Web services and vice versa. So, Web services is used to make the application platform and technology independent.

Standardized Protocol:

Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) uses the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

Low Cost of communication: Web Services uses SOAP over HTTP protocol for the communication, so one can use existing low cost internet for implementing Web Services. This solution is much less costly compared to proprietary solutions like EDI/B2B.

Support for Other communication means:

Beside SOAP over HTTP, Web Services can also be implemented on other reliable transport mechanisms. So, it gives flexibility to use the communication means of one's requirement and choice. For example Web Services can also be implemented using FTP protocol (Web services over FTP).

Loosely Coupled Applications:

Web Services are self-describing software modules which encapsulates discrete functionality. Web Services are accessible via standard Internet communication protocols like XML and SOAP. These Web Services can be developed in any technologies (like c++, Java, .NET, PHP, Perl etc.) and any application or Web Services can access these services. So, the Web Services are loosely coupled application and can be used by applications developed in any technologies.

Web Services Sharing:

These days due to complexness of the business, organizations are using different technologies like EAI, EDI, B2B, Portals etc. for distributing computing. Web Services supports all these technologies, thus helping the business to use existing investments in other technologies.

Web Services are Self Describing:

Web Services are self describing applications, which reduces the software development time. This helps the other business partners to quickly develop application and start doing business. This helps business to save time and money by cutting development time.

Automatic Discovery:

Web Services automatic discovery mechanism helps the business to easily find the Service Providers. This also helps customer to find services easily. With the help of Web Services one's business can also increase revenue by exposing their own Web Services available to others.

Business Opportunity:

Web Services has opened the door to new business opportunities by making it easy to connect with partners.

2.2 Dimensions for Web service Discovery approaches

Before starting with the approaches used for Web services discovery we define the dimensions for assessing the approaches. These dimensions are used as a benchmark for evaluating the proposed solutions in the area of Web services discovery.

Table 2.1: Dimensions

Sr.No.	Dimension	Definition
1.	Coverage	Explores the use of domain-independent and domain-specific sources to find matching services descriptions
2.	Method of determining semantic association	Determines how the distance between concepts in the taxonomy is calculated
3.	Scalability	Scope for expansion without any decrease in performance
4.	Prevalent Frameworks	Determines current frameworks using a particular approach

Coverage:

This measure explore the use of domain-independent and domain-specific sources to find matching services descriptions. It is assumed that, in the context of Web applications with no predefined domain, maximum coverage of possible interpretations of the words must be warranted. If one is limited to a particular knowledge source, such as WordNet, or a certain set of ontologies, one is constraining the scope of applications.

Method of determining semantic association:

This measure determines how the distance between concepts in the taxonomy is calculated i.e. how semantic similarity calculation between a service query and a service description is done. In order to enable semantic matchmaking, it is necessary that possible communication partners, say service providers and requesters, agree on a certain specification of a conceptualization of the domain.

Scalability:

It refers to scope of expansion of a approach as the number of services increase and when the proposed approach is applied to real time operative services on WWW.

Prevalent Frameworks using this approach:

Web service discovery is still in its developing stages and no standard benchmarks have been set so, all proposed approaches have been studied on the basis of prevalent frameworks following such approaches.

2.3 Approaches to Web Services Discovery

Web services discovery is normally defined as a matching process in which available services' capabilities can satisfy a service requester's requirements. The capability of a Web service is often implicitly indicated through a service's name, a method's name and some descriptions included in the service and it can be described as an abstract interface by using standard service description languages. With the help of the standard descriptions of Web services, various approaches can be used to find services on the Web, such as using Web search engines like Google, Yahoo, *etc*, services portals [263] and services registries like UDDI [232], ebXML, *etc*.

According to Kokash [164] information matching can be accomplished on two levels:

- In syntactic matching the similarity of data is found using syntax driven techniques. Usually, the similarity of two concepts is a relation with values between 0 (completely dissimilar) and 1 (completely similar).

- In semantic matching the key intuition is the mapping of meanings. There are several semantic relations of two concepts: equivalence more general, less general, mismatch and overlapping. Nevertheless, they can be mapped into a relation with values between 0 and 1.

An important aspect of the Web Services discovery concerns the issue of the way the services themselves are modeled. The Web services architecture has two broad classifications: Centralized and Decentralized. A review of approaches followed in

both the architectural classifications are discussed in detail in the coming sections.

2.3.1 Centralized Approaches

Keyword Based Discovery

Booth *et al.* define the Web services Discovery mechanism in a broader sense as 'the act of locating a machine-processable description of a Web Services that may have been previously unknown and that meets certain functional criteria' [41] .

The simplest data model is the Catalogue/Keyword Based which follows the legacy UDDI standard and the discovery mechanism it supports. The textual description that accompanies each Web Service is stored in the UDDI catalogue along with the tModel that provides the service functionality. The retrieval stage comprises a user or a search program, entering a query to the catalogue and the matched Web services are returned as a candidate answer set and the user browses them in order to find which one of them really suits his/her needs. First description of discovery mechanisms for services providers appears as the *match-making process*. It is the process of finding an appropriate services provider for a service requester through a middle agent [116]. Since services discovery is perceived as a matchmaking process various algorithms were proposed for key word based discovery. Charlie Abela developed an algorithm that uses two filter operations to carry out the matching [28]. The first filter limits the search space to the restrictions that the user defined in the request (search by ServiceProvider, ServiceCategory or ServiceName). The second filter is an output matching that the user defines in the request, which allows using only one filter or both. In one matching result there can be different services and, thus, a range needs to be created so as to specify which one fits the client's needs. Sometimes, the client's intervention will be needed so as to choose the one that he considers as most appropriate

Disadvantages of Keywords-based mechanism

Despite the success of the UDDI specification and its rapid uptake by the industry, the capabilities of services discovery facilities offered by it are rather limited . The lack of machine understandable semantics in the technical specifications and classification schemes used for retrieving services, prevent UDDI registries from supporting fully automated and thus truly effective service discovery. Although the WSDL documents are machine process able, they still lack the formal rigor and machine-understandable semantics that would make them amenable to logic-based reasoning and automated processing. As a result, UDDI registries cannot support fine-grained matchmaking based on the actual definitions of technical specifications or classification systems, and effectively, cannot support truly automated services discovery. In a typical service discovery scenario, a developer still needs to retrieve the WSDL document and additional specification documents referenced by a UDDI service advertisement and inspect them manually, in order to assert that the advertised service is fully interoperable with other services assembled in a service composition [53].

It has been clearly cited by Jiangang [108] that although the keyword-based discovering mechanism are supported by UDDI and most existing service search engines [79], [269] still they suffer from some key problems. Firstly, it is difficult for a user to obtain the desired services because the number of the retrieved services with respect to the keywords may be huge. Secondly, keywords are insufficient in expressing semantic concepts. This is partially due to the fact that keywords are often described by natural language, being much richer in terms of diversity. For example, syntactically different words may have similar semantics (synonyms) which results in low recall. In addition, semantically different concepts could possess identical representation (homonyms), leading to low precision. As a result, the retrieved services might be

totally irrelevant to the need of their consumers.

After analyzing the shortcomings of the present syntactic discovery mechanism of UDDI, the research community shifted towards the semantic Web services discovery mechanism but when it comes to adding semantics to the present Web services wide range of proposals have come up: some prefer information retrieval approaches, majority favors ontologies as the basic requirement for semantic match making while some consider Description logic based approaches as the best suitable candidates for semantic discovery. Within Information Retrieval approaches there are several viewpoints, ranging from applying machine learning concepts of classification, clustering, *etc.* to corpus based match making where Wordnet, Wikipedia and even search engines like Yahoo and Google are being used as knowledge bases. If ontologies are considered as the domain knowledge base then definitely there is a need for matchmaking engines for performing mapping between different ontologies. A review of all the prevalent semantic Web services discovery approaches is provided in the next section starting from the first semantic Web services approach of DAML-S.

Semantic Approaches to Services Discovery

The Semantic Web is an effort to extend the current World Wide Web by representing data on the Web in a meaningful and machine-interpretable form to better enable computers and people to work in cooperation [218]. Semantic Web Services are generally self-sufficient, reusable software components which can be used to fulfill a particular task. They have modular structure and can be invoked through the Web. This development is increasingly significant since it seems to be able to tackle some of the UDDI catalogue inadequacies. It is a vision for a Web of applications (public or private) whose ‘properties, capabilities, interfaces, and effects are encoded in an unambiguous,

and machine-interpretable form' [231].

Web services are key elements in the fields of software engineering and code reuse. They also play a very important role on the Semantic Web by providing data to semantic software agents, as described by Tim Berners-Lee:

"The real power of the Semantic Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services become available. The Semantic Web promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data come with semantics."

Despite the ever-growing number of available Web services, they are rarely described formally, which keeps them from joining the third phase of the Web. Service descriptions, if any, are typically informal, written in natural language and therefore available to human consumption only. Tim Berners-Lee highlights the need for semantically described Web services as follows:

"Many automated Web-based services already exist without semantics, but other programs such as agents have no way to locate one that will perform a specific function. This process, called service discovery, can happen only when there is a common language to describe a service in a way that lets other agents "understand" both the function offered and how to take advantage of it."

Considering the above vision, it is reasonable to conclude that any given service should be offered the opportunity to join the Semantic Web, regardless of the technologies and protocols it relies on. There are basically two groups of services, namely RPC and RESTful. They represent completely different approaches to implementing

Web services, but Semantic Web researchers should be able to address both equally.

Semantic Matchmaking

The main foundation of the investigation related to semantic matching of services is in the studies of Software Engineering that Zaremski and Wing carried out in [13], [14]. Basing their work on two of the important existing technologies for developing the Semantic Web, namely XML [69] and RDF [189], a team of researcher's with DARPA's funding developed agent markup languages such as DAML and DAML-S [52] for semantic markup of software agents and Web services respectively. Supported by an automatically inferenceable language namely DAML+OIL, DAML family of semantic markup languages together pave the way for the realization of Semantic Web Services. Specifically, DAML-S provided a semantically based view of the Web Services [218] thereby complementing the existing interface specification capabilities of WSDL. These models lay the foundation for automatic services discovery, service composition and execution in application integration [183].

One of the earliest works on matchmaking engines put in the context of semantic Web services is LARKS by Sycara *et al.* in which the services are seen as frames, and the input, output, inConstraints and outConstraints can be used to describe the essential attributes of a service [119]. The matching process in LARKS makes both syntactic and semantic matching and is formed of a group of filters (which are independent from each other) that restrict progressively the number of candidate advertisements to be matched. In addition to utilizing a capability-based semantic match, this engine uses various Information Retrieval (IR)-based filters such as namespace comparison, word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching, thereby reducing the number of false positives. In their follow-

up work to DAML-S specification, Paolucci *et al.* tie the semantic representation of Web services work with Web service directories/registries by arguing that Web services discovery should be based on the semantic match between a declarative description of the services being sought, and a description of the services being offered [151]. They present a sample semantic matching algorithm that matches the inputs, outputs, preconditions and effects of service request with those of service advertisements. They also argue that this semantic matching is outside the capabilities of registries such as UDDI and languages such as WSDL.

The earliest approach to the problem of semi-automatic translation of a WSDL-based Web services to DAML-S- based semantic Web Services is described in [152]. Their WSDL2DAMLS converter automatically generates grounding ontology of the services (Execution semantics) and partially creates the Process Model (Functional Semantics) as a set of atomic processes. The designer of a SWS should manually construct the complete Service profile based on process control and data flow extracted from text or other files accompanying the WSDL description of the service. The converter also perform a preliminary step to the addition of data semantics by converting existing complex XSD types into corresponding DAML-S concepts, which are not related to ontologies available in the Semantic Web.

Akkiraju *et al.* provide a design mechanism for a tighter integration of semantic matching with UDDI registry by directly extending UDDI's inquiry application programming interface and its implementation [183]. This approach incorporates semantic matching directly in UDDI registry by altering the API that users of UDDI registry are familiar with. While this is a workable solution, it proposes embedding matching capability into UDDI registry implementation. This makes UDDI matching specific to a particular service description language - in their case, DAML-S. Colgrave *et al.*

present an approach in which a UDDI registry can select a matchmaking engine dynamically based on the services requirements and find suitable services advertisements whose capabilities are described in the language that the requester can understand [94]. In this approach, the matchmaking engines themselves are advertised as Web services in the registry.

Overhage propose implementation of semantic descriptions as an extension of the UDDI protocol, termed as E-UDDI. E-UDDI introduces 'blue pages', sections that contain semantic descriptions of Web Services, where descriptions are implemented in DAML-S [221]. The model provides extensions to the green page section of the UDDI, by adding the capability to define constraints in the Web services execution sequence. However E-UDDI seems to be more a vision than an implementable system.

The approximation used for services matching by Castillo *et al.* is an algorithm based on the subsumption tree given by a description logic (DL) reasoner [98]. The services descriptions have been specified in DAM+OIL and since DAM+OIL is based on DL, the DL reasoner is the heart of the matching algorithm.

Carman and Serafini designed an algorithm for semantic matching of complex types [134] where structure information is used to infer equal, more general or less general relations between type schemas. Alternate approaches propose services process-model matching [2], recursive tree matching [207], P2P discovery [70], automated selection of WSMO services [247] and METEOR-S for WSDL-S services [122]. Stroulia *et al.* developed a suite of algorithms for similarity assessment of service specifications [66]. Agre *et al.* describe the architecture of the INFRAWEBS SWS Designer-a tool for semiautomatic converting of non-semantic Web services to semantic ones [75]. A general conceptual architecture of the Designer, which does not depend on a specific framework used for describing semantic Web services, is specified.

According to Sandeep *et al.* [216] in some case, the semantic Web based systems can not satisfy the client requirements using only the single service components. In those cases, discovery and selection is used for selecting the most appropriate service component followed by services composition [216]. Umardand *et al.* designed a dynamic Web services selection framework that makes use of a semantic matcher to support matching and composition of software services [248]. It uses a DAML document to specify the functional and the non functional requirements, agreements, contracts. It has some memory where it stores these DAML descriptions of Web services. To describe an interface of a service (technical description of service i.e input and output parameters to service, location of service), WSDL document is used. It matches the DAML description of user requirements with the DAML description of a Web services to find a matching services.

Agents based approaches

The Information Management Group at the University of Manchester [127] developed a prototype description of the matching algorithm based on the descriptions of DAML-S. It uses the reasoner Racer Descriptions Logics for making semantic matching between services and uses JADE as the agent platform. In the DAML-S Matchmaker design, the services requests in DAML-S are matched with those of the services advertisement. In such approach, the profiles are treated as complete entities and their components are not treated separately. A certain degree of similitude is associated to the matching result.

The Intelligent Software Agents Group team at the Carnegie Mellon University (CMU) designed an architecture based on ATLAS (Agent Transaction Language for Advertising Services), a language based on DAML, for matching of services. It uses

two filters: matching of functional attributes of services for determining the advertisement applicability and matching of services functionalities, to determine if the advertised services match the required services [239]. Kawamura *et al.* provide another deployed system, in which a series of four filters are used to refine the set of search results [237]. These filters cover the usage of services namespaces, textual information, inputs/outputs of services, and the subsumption relationship of the I/O constraints.

InfoSleuth an agent-based information discovery and retrieval system, adopts 'broker agents' to perform the syntactic and semantic matching [141]. The broker agent matches agents that require services with other agents that can provide those services. By maintaining a repository containing up-to-date information about the operational agents and their services, the broker enables the querying agent to locate all available agents that provide appropriate services. In InfoSleuth, the services capability information is written in LDL++ [43], a logical deduction language. RETSINA is a multi agent infrastructure [120]. It distinguished three general agent categories: services provider, services requester, and middle agent. To describe these agents' capabilities in the matching process, it has defined and implemented an Agent Capability Description Language, called Larks. Larks offers the option to use application domain knowledge in any advertisement or request by using a local ontology, written in a specific concept language ITL, to describe the meaning in a Larks specification [192].

Moreau *et al.* perform semantic matching where agents perform Grid computations as Web services [128]. They transform agent ontologies into XML and semantic matching is performed by validating structurally expressed queries against agent description schemas. An approach of combining semantic annotation of Web services and agent-based publishing and discovery is followed by Abela [28].

Logic-based approaches

The objective of hybrid semantic Web services matching is to improve semantic services retrieval performance by appropriately exploiting means of both logic based and approximate matching where each of them alone would fail. Prominent examples of such *logic-based approaches* to semantic services discovery are provided by the OWLS-UDDI matchmaker [142], RACER [127], MAMA [211] and the WSMO services discovery approach [91]. These approaches do not exploit semantics that are implicit, for example, in patterns or relative frequencies of terms in services descriptions as computed by techniques from data mining, linguistics, or content-based information retrieval (IR). According to Chukmol despite the interesting advantages of Description Logics, expressing a query in this formalism is still not an intuitive and easy task for an ordinary user or even for a software developer [246]. Moreover, Description Logics based Web services discovery tools are not available to large public .

Ontology based Approaches

Ontology is a formal, explicit specification of a shared concept [238]. An ontology models all the entities and relations inside a domain necessary for knowledge representation. The key of ontologies is that they can be shared and, therefore, they increase efficiency and interoperability. Currently, ontology has emerged as a very important discipline as its usefulness has been demonstrated in varying types of applications which include information organization and extraction, personalization, natural language processing, artificial intelligence, knowledge representation and acquisition. Ontology management tools provide the facilities and environments to build a new ontology from scratch, modify existing ontologies, reuse other ontologies and also provide a visual interface for viewing ontology [171]. A number of frameworks like OWL-S [48] and WSDL-S [184]

supporting ontologies have been proposed to address the semantic heterogeneity among Web resources and services.

In order to enable semantic matchmaking, it is necessary that possible communication partners, say service providers and requesters, agree on a certain specification of a conceptualization [238] of the domain, *i.e.* a shared, formal ontology. Distance between concepts in the taxonomy is the main parameter used by structural approaches. This measure makes sense under the assumption of equally distributed instances over concepts; otherwise pairs of concepts in a fine grained part of the taxonomy would be ranked with lower similarity than concepts at the same distance in another part [7].

There have been many proposals for Web services discovery based on OWL ontologies. Many researchers have proposed the usage of ontology [12], [61], [118], [127], [151], to annotate the elements in Web services. Modeling Web services with ontologies, the semantic representation of concepts and their relations can be exploited and thus semantic matching to be performed. Semantic descriptions of Web Services can be obtained with the use of semantic languages like DAML-S [52] or OWL-S [48]. Paolucci et. al. [152] present a framework to allow WSDL and UDDI perform semantic matching where Web services are modeled as ontologies, or Service Profiles as they are called, with the use of the DAML-S [84].

Li and Horrocks describe the design of a service matchmaker that uses DAML-S based ontology [127]. It uses techniques from knowledge representation to match service capabilities. In particular, it defines a description logic (DL) reasoner; advertisements and requests are represented in DL notations.

The WSMO framework provides ontology translation to support automatic interoperation between Web services [91]. Specifically, in the WSMO architecture various mediators (e.g., OO-Mediators) address the interoperability problems that arise when

various Web services work together. To incorporate QoS attributes with services discovery, Zhou *et al.* propose a DAML-QoS ontology for specifying various QoS properties and metrics [37]. However, their framework assumes the existence of a single QoS ontology for the service providers and requesters, and hence does not take into consideration the specification of semantic correspondences. Further, there is no provision for the users to specify ranking criteria (based on non-functional attributes) for services selection [114].

The first hybrid OWL-S services matchmaker called OWLS-MX was presented by Klusch *et al.* that exploits means of both crisp logic based and Information Retrieval (IR) based approximate matching [142]. Their experimental evaluation indicated that under certain constraints their way of matching can indeed outperform logic.

Syeda-Mahmood *et al.* explore the use of domain-independent and domain-specific ontologies to find matching services descriptions [241]. The domain-independent relationships are derived using an English thesaurus after tokenization and part-of-speech tagging. The domain-specific ontological similarity is derived by inferencing the semantic annotations associated with Web services descriptions. Matches due to the two cues are combined to determine an overall semantic similarity score. They demonstrate that by combining multiple cues better relevancy results can be obtained for services matches from a large repository, than could be obtained using any one cue alone [92].

An implementation framework for semantic matching based on ontology matching is proposed by Hu [273]. The author provides a framework that uses ontologies to discover ('bind' in that case) the Web services that best match a specific operation domain (desired set of operations). The available data are represented with domain ontologies and the available operations, with operation ontologies. Binding is performed by a binding ontology that decides what fits where according to binding relations.

Pathak *et al.* allow the users to specify context-specific semantic correspondences between multiple ontologies to resolve semantic differences between them [114]. These correspondences are used for selecting services based on the user's functional and non-functional requirements, which are then ranked based on user-specified criteria.

Bose *et al.* have rightly stated that despite the popularity ontology approach, a majority of the Web services available over Internet are not annotated using any of these ontologies [3]. This is mainly because WSDL is the standard language to express a service and any semantically enhanced language is not yet a standard that must be followed. Secondly, a number of ontologies and frameworks have been proposed each having its own advantages and disadvantages. This makes hard to select one as a standard. Moreover, there exist a large number of Web services on the Internet that have been already created long before any of these annotations were proposed.

Machine Learning based Approaches

Machine learning techniques can be used to build classifiers for a category by observing the characteristics of a set of documents or corpus. Laura *et al.* [202] propose grouping of results of traditional search engines into various categories using semantics techniques and ontologies available on Web. Subramaniam *et al.* [240] study various classifiers like Nave Bayes [155], SVM [250], Neural Net, *etc.* for categorization of emails. Similar techniques can be applied to classify Web services into different domains or categories.

Some *clustering and classification techniques* of machine learning are applied to the problem of Web services matching and classification at either the whole Web services level [12] or at the operation level [259]. Hess and Kushmerick [12] treat all terms from portTypes, operations and messages in a WSDL document as a bag of words and multidimensional vectors are created from these bag of words which are further

used for Web services classification. According to Zhuang [275] although this type of classification retrieves matches with higher precision than full-text indexed search, the overall matches produced, however, do not guarantee a match of operations to operations, messages to messages, *etc* . Madhavan *et al.* focus on matching of operations in Web services [109] by clustering parameters present in inputs and outputs of operations (*i.e.*, messages) based on their co-occurrence into parameter concept clusters. This information is exploited at the parameter, the inputs and output, and operation levels to determine similarity of operations in Web services.

Arbramowicz [251] proposes an architecture for Web services filtering and clustering where services filtering is based on the profiles representing users and application information, which are further described through OWL-S. In order to improve the effectiveness of the filtering process, a clustering analysis is applied to the filtering process by comparing services with related clusters. Another similar approach followed by Nayak el al. [196] concentrates on Web services discovery with OWL-S and clustering technology, which consists of three main steps: the OWL-S is first combined with WSDL to represent services semantics before a clustering algorithm is used to group the collections of heterogeneous services together. Finally, a user query is matched against the clusters, in order to return the suitable services.

A Web services discovery method *combining semantic and statistical association* is discussed in [259] where Dong describes Woogle, the system to offer more refined Web services similarity search capability which first clusters the parameter names into semantically meaningful concepts and then uses them to decide the similarity between inputs/outputs sets and operations. Hyperclique pattern discovery methodology is another machine learning approach which combines semantic relationship ranking, for establishing semantic relevance, and a hyperclique pattern discovery approach that

groups Web services parameters into meaningful associations has been proposed in [20]. These associations combined by the semantic relevance are then leveraged to discover and rank Web services.

Algorithms using Singular Vector Decomposition (SVD) [17] and probabilistic latent semantic analysis [106] have been proposed to find the similarity between the Web services to enhance the accuracy of services discovery. Approach used by Bose *et al.* is an extension of SVD to support-based latent semantic kernel to further increase the accuracy of Web services discovery [3]. The constructed semantic kernel, which represents each document as a set of topics, discovers the hidden topics and their relationships to the term and document set. In this approach the dimensionality reduction of the term-document matrix by binning and merging the training documents has been proposed to create the semantic kernel. The similarity between the user query and Web services is calculated using the support based latent semantic kernel.

A common problem with the SVD based approaches is that the computation of the high dimensional matrix representing the training documents is expensive. There have been some attempts to reduce the dimensionality of matrix prior to applying SVD. One such solution is using random projection [31]. In random projection, the initial corpus is projected to 'L' dimensions, for some L > k, where 'k' is the dimension of the semantic kernel, to obtain a smaller representation which is close to the original corpus and then perform SVD on the reduced dimension matrix.

Jiangang [107] propose extention of the SVD of matrix approach of matching Web services with a different methodology called Probabilistic Latent Semantics Analysis (PLSA) [234], which has sound probabilistic interpretation and better performance. The probabilistic semantic approach is based on the PLSA model called aspect model [235]. PLSA utilizes the Bayesian Network to model an observed event of two random

objects with a set of probabilistic distributions. In their extended work to [107] authors proposed CPLSA [108] where Web services whose contents are not compatible with a user's query are filtered out via a clustering algorithm to acquire an initial working dataset. As a next step, PLSA is applied to capture semantic concepts hidden behind the words in a query and the advertisements in services, so that services matching is implemented at an advanced concept level.

In [85] Constantinescu *et al.* focus on services discovery based on a directory where Web services are clustered into the predefined hierarchical business categories. In this situation, the performance of reasonable services discovery relies on both services providers and services requesters having prior knowledge on the services organization schemes [108]. Bruno *et al.* experimented with automated classification of WSDL descriptions using support vector machines [132].

In Information Retrieval approaches to services discovery a query consists of keywords, which are matched against the stored descriptions in a UDDI registry. An elegant approach to tackle the inadequacy of keyword-based Web Services Discovery was proposed by Sajjanhar [17]. The key concept in their approach is to represent services descriptions as document vectors, a dominant approach in the IR field. A description text, corresponds to a vector 'v' in the vector space spanned by all the terms used in all services description texts and represent all the document vectors as columns in a term-document matrix A. Another IR technique is afterwards applied, which transforms the matrix A achieving a representation of the document collection by its more significant semantic concepts called Latent Semantic Indexing (LSI) [159]. This method is observed to return documents of the modeled text collection, which are more closely related to the semantics of the expressed query, regardless of exact matching or not with the query terms. When applying LSI to the discovery of Web

Services it was observed that description vectors resulting from the transformation of the original matrix, were mapped more closely to the vector space representation of the query, than the respective representations of plain keyword-based descriptions.

The Semantic Web is a medium where the data can be shared and processed both by automated tools and humans. The key point is in the automation and integration of the processes through machine readable languages. In order to use and connect all the information available on the Web, the software agents should be able to understand the information *i.e.* the data should be written with semantics. Therefore, in XML or HTML documents, additional semantics or metadata should be added so that the software programs can fix the meaning of the documents labels. Any SWS offer discovery mechanism, together with any necessary semantic annotations mechanisms, would be different and novel because SWS offer discovery aims to be generic, independent of the domain of the services offers. Indeed, the semantic annotations should make the offer discovery algorithm adapt to any available negotiation or query protocol [103].

Annotations based Approaches

Annotation are considered as a promising technology to add and manage the knowledge associated with a set of resources. Annotating specific domains with accuracy from an automatic or semiautomatic viewpoint has raised a challenge for the current state of the art of semantic technologies [105]. Some recent works have proposed annotating Web services manually with additional semantic information, and then using these annotations to compose services. Hess and Kushmerick suggest the use of machine learning to generate suggestions for annotating Web services [12].

METEOR-S discovery framework addresses the problem of discovering services in a scenario where services providers and requesters may use terms from different on-

tologies [122]. Their approach relies on annotating services registries (for a particular domain) and exploiting such annotations during discovery. The element level matching is based on a combination of Porter Stemmer [139] for root word selection, WordNet dictionary [30] for synonyms, abbreviation dictionary to handle acronyms and NGram algorithm for linguistic similarity of the names of the two concepts. The schema matching examines the structural similarity between two concepts. Both element match score and schema match score are then used to determine the final match score.

In a related effort, Patil and colleagues have developed MWSAF, a Web service annotation framework [122]. In their work, they generate recommendations for automatically annotating WSDL documents. To accomplish this they match XML schema used by the WSDL files with ontologies by creating canonical schema graphs. A tool called ASSAM (Automated Semantic Service Annotation with Machine Learning) was developed by Andreas *et al.* [10] that assists a user in creating semantic metadata for Web Services. ASSAM consists of two parts, a WSDL annotator application, and OATS, a data aggregation algorithm. In ASSAM authors developed an iterative relational classification algorithm for semantically classifying Web Services, their operations, and input and output messages. Bai and Liu propose matching of semantic annotations of a Web service using fuzzy set theory [123]. A survey of semantic annotations platforms is presented by Reeve and Han [129].

Chukmol *et al.* consider a Web service as a resource (e.g. similarly to photo on Flickr [73] or URL on del.icio.us [44]) on which tags and annotations are added (a Web service can be tagged by many users) [246]. Considering various tagging styles perceived in Web 2.0 environment [255] three types of tagging / annotation are proposed to the users. Users can use: a) keyword tags (a set of simple or composed words separated by a delimiter), b) free text tags allowing users to comment freely on a Web service by a free

text in the form of sentence or paragraph and c) structural guided tags allowing users to tag a service using free text to fill in different information fields (organized according to a predefined structure) proposed by the discovery system. Users will be guided by the system to complete the tagging information inside a form. In order to deal with these three forms of tags efficiently, a matching module is proposed which embeds relatively information retrieval techniques (for the fact that it works with natural language tags) capable of providing interesting result via textual search.

Kourtesis *et al.* use a semantically-enhanced registry that builds on the UDDI specification and augments its services publication and discovery facilities [53]. The proposed solution combines the use of SAWSDL for creating semantically annotated descriptions of services interfaces and the use of OWL-DL for modelling services capabilities and for performing matchmaking via DL reasoning.

Using Knowledge corpus for services discovery

An alternative solution for ontology-based semantic Web services is usage of some lightweight semantic Web services discovery prototypes directly targeting the WSDL files with WordNet. This is an effort to enhance the precision of Web services discovery without involving an additional level of semantic markup. Web services similarity is defined using a WordNet-based distance metric by Wu and Wu [112]. Stroulia and Wang [66] employ WordNet to expand the query and WSDL files with the synonyms, direct hypernyms, hyponyms, and siblings senses, and then to calculate the syntactic similarity between the WordNet-powered VSM feature vectors and the semantic distances between the identifiers of the WSDL files.

Zhung *et al.* use a corpus-based method to facilitate matchmaking in WSDL files [275]. Here an algorithm is designed for identifying the semantic similarity between the two WSDL files. Using the Web as an effective corpus, the Web services matcher looks

at all possible pairs of elements from the two WSDL files it is matching and assigns a similarity score to each pair; based upon such scores, it later generates the overall similarity score for the two WSDL files. A corpus of Web documents belonging to the functional domain of the WSDL files being matched is used as the context for the word relator. The word relator calculates word-similarity scores based on the similarity of the context in which the words appear in the documents.

Kokash [164] present a similar method to join syntactic, semantic and structural similarities of different elements in a single measure to ameliorate retrieval performance. They construct three sub-vectors for each document and query: stems of the original words in the document (*i.e.* the services description and the query), stems of words' synonyms for all word senses, and stems of words' direct hypernyms, hyponyms and siblings for all word senses. Different weights are assigned to the different sub-vector matching scores. The overall matching score between a service description and a query is the average of their three sub-vector matching scores. Ding and Buyya proposes a guided meta-search engine, called Guided Google that serves as an advanced interface to the actual Google.com search engine, developed using the Google Web Services. It guides and allows the user to view the search results 'combinatorial keywords' and 'search by hosts' [42]. Many authors have proposed the use of external resources such as thesaurus, dictionaries or more complicated ontologies to perform the task [227], [270], [271], [71]. Gligorov proposed the idea of approximate ontology mappings by using Google based similarity measure *i.e.* Google distance as a weighting heuristic [191].

All of these approaches discussed so far assume a *centralized repository* of Web services that offers search functions. Work done to investigate how semantic matching could be done in a decentralized services environment is discussed in the coming section.

The table 2.2 provides a short summary of approaches for Web service discovery vis-

Table 2.2: Approaches to Web Service Discovery vs Dimensions

Sr.No.	Approaches to Web Service Discovery	Cover age	Method of determining semantic association	Scal ability	Prevalent Frameworks using this approach
1.	Logic Based Approaches	Domain Dependent	Use Formal logic for measuring similarity	Yes	OWLS-UDDI matchmaker, MAMA, RACER
2.	Agent Based Approaches	Domain Dependent	Use Agent description language	Yes	InfoSleuth, ATLAS
3.	Ontology Based Approaches.	Domain Dependent	The semantic distance between two concepts in an ontology	Yes	WSMO, OWL-S
4.	ML Based Approaches	Domain Independent	Clustering, Classification, SVD, LSA, PLSA	Yes	Woogle
5.	Annotations Based Approaches	Domain Independent	Measure term similarity based on association between terms	Yes	WSDL-S, ASSAM, METEOR-S
6.	Corpus Based Approaches	Domain Independent	Wordnet, Wikipedia, NGD, PMI	Yes	Guided Google

a-vis and the dimensions used for measuring the performance of a particular approach.

2.3.2 Decentralized Approaches

Decentralized approaches to Web services discovery includes Peer to Peer (P2P) platforms which provide a good arena for the Web Services Discovery mechanisms' implementations. A P2P overlay network provides an infrastructure for routing and data location in a decentralized, self-organized environment in which each peer acts not only as a node providing routing and data location services, but also as a server providing

services access. P2P can be considered a complete distributed computing model. Recently proposed P2P systems include CAN [222], Pastry [16] and Chord [88]. These systems arrange the network of peers to a ring and nodes are assigned IDs drawn from a global address space. Peers are assigned a range of keys from the global address space that they are responsible for and each peer stores auxiliary information in order to appropriately route key lookups. Usually a key lookup is initiated at a peer. In this case the peer consults its look-up table in order to successfully route the query to the peer that stores the queried key. In the case of Chord routing with the aid of look-up table, simulates binary search on the address space of all peers. Chord mainly has been adopted as the overlay P2P network distributed Web services architectures. The hosts in the P2P network publish their services descriptions to the overlay, and the users access the up-to-date Web services.

A discussion on Semantic Web and peer-to-peer issues is presented by works of Armugam *et al.* [131] and Schlosser *et al.* [156]. Verma *et al.* present a scalable peer-to-peer infrastructure of registries for semantic publication and discovery of Web services [122]. In an architecture called P2P-based Web services Discovery (PWSD) authors use Chord P2P protocol as overlay, consisting of Service Peers (SP) [266]. Each SP is mapped to several Logical Machines (Different Machines corresponding to the same hardware). Each Logical Machine maintains the necessary interfaces to map and search Web Service in the P2P network. Service Descriptions as well as queries are hashed and routed in the Chord network. Keyword matching is combined with P2P storage presenting a system that also maps the XML Service Descriptions over a P2P Network, using distributed hashing. In that sense peers act both as services providers and request generators. The XML Service Descriptions are parsed in order to extract services keywords and keywords are hashed with the MD5 hash function. The underlying P2P

Network Protocol is also Chord and the modified system is called XChord. Chord's distribution policy is enforced to route the generated hash descriptions to nodes. A Web Service query starting at a peer node and decomposed into keywords which are subsequently sought for using the Chord searching principle. XChord is proved more stable, load balanced and less space consuming according to the conducted experiments [97].

Banaei-Kashani *et al.* developed the WSPDS system, a peer-to-peer discovery services with semantic-level matching capability [70]. Their framework is guided by the principle that a decentralized design for Web services discovery is more scalable, fault tolerant and efficient as opposed to a centralized approach (e.g., UDDI). WSPDS also semantically- annotates the WSDL files using the WSDL-S framework described in [232]. According to Pathak [114] one advantage of this approach is that it makes the WSDL-S file agnostic to any ontology representation language (e.g., OWL ,WSMO) but adopting such a framework means that WSDL files for the existing Web services would have to re-written, which is an additional overhead .

The Speed-R system is a Web services storage and retrieval system that uses ontologies and a P2P infrastructure [118]. Some nodes in the P2P subsystem are assigned registries, which in turn partitioned according to their specific domain. An ontology is assigned to each domain and the P2P system is based on JXTA [125] implementation. Its architecture is based on role assignment to peers (for example some nodes have undertaken the role of controlling updates and propagating them), thus their system may suffer from single point failure.

In the system of Schmidt and Parashar, a unique ID is generated for each Web Service, through the Hilbert curve mapping [36]. The IDs are then stored in a Chord of Web Service Peers. Thus, the storage and retrieval of WS inherits the load balancing

capability and the dynamic nature of Chord. The main advantage of the model followed by [36] is that it can efficiently support partial match queries. These queries are realized efficiently mainly due to the clustering properties of Hilbert curve. The querying procedure is further enhanced by some query optimization heuristics.

After reviewing various approaches to Web services discovery, next section provides a brief overview of prevalent semantic Web services frameworks.

2.4 Prevalent Semantic Web Services Frameworks

Web services enhance current Web functionality by altering its nature from document to service-oriented and transforming from a provider of static pages to a provider of interactive, automated and intelligent services that interact via the Internet. Multiple Web services may interoperate to perform tasks, provide information, transact business and generally take action for users, dynamically and on demand. [8].

The Semantic Web encompasses efforts to build a new WWW architecture that enhances content with formal semantics. Content should be suitable for machine consumption, as opposed to content that is only intended for human consumption. This will enable automated agents to reason about Web content, and produce an intelligent response to unforeseen situations [8].

Semantic Web Services (SWS) aims at an integrated technology for the next generation of the Web by combining Semantic Web technologies and Web services, thereby turning the Internet from a information repository for human consumption into a world-wide system for distributed Web computing. Therefore, appropriate frameworks for Semantic Web services need to integrate the basic Web design principles, those defined for the Semantic Web, as well as design principles for distributed, service-orientated

computing of the Web.

Semantic descriptions of Web services are necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains. Semantic Web Services constitute one of the most promising research directions to improve the integration of applications within and across enterprise boundaries. The activities required for running an application using SWS include: publishing, discovery, selection, composition, invocation, deployment and ontology management.

Some important SWS frameworks and related technologies proposed are OWL-S [48], WSDL-S [184], WSMO [91], SWSF [208], *etc.* An overview of the prevalent SWS frameworks along with the main concepts that they define is provided in the comming sections.

2.4.1 WSMO

The WSMO initiative1, part of the SDK Cluster2, is the major initiative in the area of SWS in Europe and has the aim of standardizing a unifying framework for SWS which provides support for conceptual modeling and formally representing services, as well as for automatic execution of services [91].

The Conceptual Model - The Web Services Modeling Ontology (WSMO)

WSMO [91] provides ontological specifications for the core elements of Semantic Web services. WSMO is based on the following design principles :

Web Compliance: WSMO inherits the concept of IRIs (Internationalized Re- source Identifier) for unique identification of resources as the essential design principle of the Word-Wide Web. It adopts the concept of Namespaces for denoting consistent information spaces, supports XML and other W3C Web technology recommendations,

as well as the decentralization of resources.

Ontology Based: Ontologies are used as the data model throughout WSMO, *i.e.*, all resource descriptions as well as all data interchanged during services usage are based on ontologies, allowing semantically enhanced information processing as well as support for semantic interoperability.

Strict Decoupling: Decoupling denotes that WSMO resources are defined in isolation, meaning that each resource is specified independently without regard to possible usage or interactions with other resources.

Centrality of Mediation: As a complementary design principle to strict decoupling, mediation addresses the handling of heterogeneities that naturally arise in open environments. Heterogeneity can occur in terms of data, underlying ontology, protocol, or process. WSMO deploys Web services by making mediation a first class component of the framework.

Ontological Role Separation: WSMO differentiates between the desires of users or clients and available services.

Description versus Implementation: WSMO differentiates between the descriptions of Semantic Web services elements (description) and executable technologies (implementation). While the former requires a concise and sound description framework based on appropriate formalisms in order to provide a concise for semantic descriptions, the latter is concerned with the support of existing and emerging execution technologies for the Semantic Web and Web services.

Execution Semantics: In order to verify the WSMO specification, the formal execution semantics of reference implementations like WSMX as well as other WSMO-enabled systems provide the technical realization of WSMO. This principle precisely defines the functionality and behavior of the systems that are WSMO compliant.

Service versus Web service: WSMO provides means to describe Web services that provide access to services.

Web Service Modeling Language (WSML)

Web Service Modeling Language (WSML) [90] is a language for the description of ontologies, goals, Web services, and mediators based on the conceptual model of WSMO. WSML provides one coherent framework which brings together Web technologies with different well-known logical language paradigms. Description Logics [26], Logic Programming [111] and F-Logic [144] are taken as starting points for the development of a number of WSML language variants. WSML takes a top-down approach to Semantic Web Service description. So far, the focus has been on the use of different formalisms for describing static knowledge (ontologies) related to the Web services.

2.4.2 OWL-S

OWL-S [48], part of the DAML program10, is an OWL-based Web Service Ontology; it aims at providing building blocks for encoding rich semantic service descriptions, in a way that builds naturally upon OWL. Very often is referred to the OWL-S ontology as a language for describing services, thus reflecting the fact that it provides a vocabulary that can be used together with the other aspects of the OWL to create service descriptions.

The OWL-S ontology mainly consists of three interrelated sub-ontologies, known as the profile, process model, and grounding. The profile is used to express 'what a service does', for purposes of advertising, constructing service requests, and matchmaking; the process model describes 'how it works', to enable invocation, enactment, composition, monitoring, and recovery; and the grounding maps the constructs of the

process model onto detailed specifications of message formats, protocols, and so forth (normally expressed in WSDL). All these sub-ontologies are linked to the top-level concept Service, which serves as an organizational point of reference for declaring Web Services; whenever a service is declared, an instance of the Service concept is created.

By switching from DAML+OIL to OWL, OWL-S tried to adopt existing Semantic Web recommendations yet still maintain bindings to the world of Web services by linking OWL-S descriptions to existing WSDL descriptions. The ontology itself defines the top-level concept 'Service' and three OWLS sub-ontologies known as the 'Service Profile', 'Service Model', and 'Service Grounding'.

Service Profile

Each instance of the service class presents zero or more service profiles. A service profile expresses 'what a service does' for the purposes of advertising and serves as a template for service requests, thus enabling discovery and matchmaking. The profile, is the specification of what functionality the service provides and comprises nonfunctional aspects such as references to existing categorization schemes (e.g. UNSPC) or ontologies, provider information, and the quality rating of the service. Information transformation is represented by inputs and outputs; the change in the state of the real world caused by the execution of the service is represented by Inputs and outputs refer to OWL classes describing the types of instances to be sent to the service and the respective responses to be expected. The format of the preconditions and effects is not fixed, but the authors of OWL-S allow various formats to be embedded into the OWL ontology. Preferably, SWRL [86]; alternatively, DRS [49] or KIF [140] expressions may be allowed. A possible problem in this context is that the semantics of these conditions is not covered by the (description logics) expressivity of the OWL-S ontology itself, but

by reference to these languages. Parties, exchanging OWL-S profile descriptions or using OWL-S for discovery, need to agree on the language for expressing conditions and the notions of a 'match' which is not addressed in the standard as such [45].

Service Model

A service might be described by a service model which exposes 'how a service works'. The main use of a service model is to enable invocation, enactment, composition, monitoring and recovery. The service model views the interactions of the service as a process. A process is not necessarily a program to be executed, but rather a specification of ways in which a client may interact with a service. OWL-S distinguishes between atomic processes, simple processes and composite processes. Atomic processes are simple operations, whereas simple processes are views of complex or atomic processes. Composite processes are built up from atomic or simple processes by standard workflow constructs such as sequence, split, or join to determine the control flow, plus additional dataflow information (which outputs are routed to which inputs within the workflow). Results and outputs may depend on a condition, again expressed in a third-party logical language (SWRL, KIF, or DRS). A possible problem is that the semantics of the workflow constructs is not expressible in the description logics underlying OWL, for which reason this semantics has been externally defined [218].

Service Grounding

In order to map to the Web service world, an OWL service can support a grounding which maps the constructs of the process model to detailed specifications of message formats, protocols, *etc.* OWL-S allows one to map atomic processes to WSDL operations and their inputs and outputs to WSDL messages. These message mappings might

have XSLT transformations attached, in order to solve the problem of lifting/lowering from the ontological representation in OWL to XML Schema in WSDL.1. Although WSDL is the only completely defined grounding for OWL, OWL-S does not restrict itself to WSDL as the only underlying service technology. Rather, OWL-S is a general service description ontology, and is extensible to other grounding mechanisms [45].

2.4.3 WSDL-S

WSDL-S [184] proposes a mechanism to augment the Web service functional descriptions, as represented by WSDL [57], with semantics. This work is a refinement of an initial proposal developed by the Meteor-S group, at the LSDIS Lab15, Athens, Georgia.

Starting from the assumption that a semantic model of the Web service already exists, WSDL-S describes a mechanism to link this semantic model with the syntactical functional description captured by WSDL. Using the extensibility elements of WSDL, a set of annotations can be created to semantically describe the inputs, outputs, and the operation of a Web service. By this the semantic model is kept outside WSDL, making the approach agnostic to any ontology representation language. The advantage of such an approach is that it is an incremental approach, building on top of an already existing standard and taking advantage the already existing expertise and tool support. In addition, the user can develop in WSDL in a compatible manner both the semantic and operational level aspects of Web services.

Semantic Annotations

WSDL-S proposes five extensibility elements to be used in annotating the inputs, outputs, and operations of Web services:

modelReference: Extension element that denotes a one-to-one mapping between schema elements and concepts from the ontology;

schemaMapping: Extension attribute that can be added to XSD elements or complex types to associate them with an ontology (used for one-to-many and many-to-one mappings);

precondition: Extension element (child of the operation element) used to point to a combination of complex expressions and conditions in the ontology, that have to hold before the execution of the Web service's operation;

effects: Similar with preconditions, with the difference that the conditions in the ontology have to hold after the execution of the Web service's operation.

category: Extension attribute of the interface element that points to categorization information that can be used for example when publishing the Web service.

Each of these elements can be used to create annotations.

Annotating the Input and Output Elements

If the input or the output is a simple type it can be annotated using the extensibility of the XML Schema element: the modelReference attribute is used to associate annotations to the element. If the input or the output is a complex type two strategies can be adopted: bottom level annotation and top level annotation. In bottom level annotation all the leaf elements can be annotated with concepts from the ontology.

In the case of one-to-many or many-to-one correspondences top level annotation method are used. Although it is a more complex method, its advantage is that it allows for complex mappings to be specified between the XML element and the domain ontology. The semantic of the elements in the complex type is captured by using the schemaMapping attribute.

Service Categorization

Adding categorization information to the Web service can be helpful in the discovery process. That is, by categorizing the published Web services can narrow the range of the candidate Web services. Multiple category elements can be used to state that a Web service falls in multiple categories as one category elements specifies one categorization.

Language for WSDL-S

WSDL-S does not fix a specific formalism for semantic descriptions. WSML, OWL, and UML are mentioned as examples of formalisms that could be adopted. WSDL-S fixes the underlying service technology (referred to as 'grounding' in the other formalisms) to WSDL by definition.

2.4.4 Semantic Web Services Framework (SWSF)

Semantic Web Services framework (SWSF) [208] is another approach for Semantic Web Services, being proposed and promoted by Semantic Web Services Language Committee (SWSLC), of the Semantic Web Services Initiative (SWSI). It is based on two major components: an ontology and the corresponding conceptual model by which Web services can be described, called Semantic Web Services Ontology (SWSO) and a language used to specify formal characterizations of Web services concepts and descriptions called Semantic Web Services Language (SWSL).

The Semantic Web Services Framework (SWSF) is a relatively recent attempt towards a Semantic Web service annotation framework that greatly profits from previous work with its roots in OWL-S and the Process Specification Language (PSL) [181], standardized by ISO 18269. This framework is a joint proposal by the Semantic Web Services Language Committee and was also submitted to the W3C in September 2005.

SWSF is based on two major components: an ontology (or conceptual model) and a language used to axiomatize it.

Conceptual Model

The Semantic Web Services Ontology (SWSO) has been influenced by OWL-S and shares its three concepts of profile, model, and grounding. Thus SWSO can be seen as an extension or refinement of OWL-S. One major difference between OWL-S and SWSF is in the expressiveness of the underlying language, which is, instead of OWL, a richer language called the Semantic Web Service Language (SWSL) in SWSF. Another fundamental aspect is a richer behavioral process model based on PSL . In the SWSF approach, there are two independent formalizations of the conceptual model, both expressed in variants of the underlying SWSL.

SWSO presents a conceptual model for semantically describing Web services and an axiomatization, formal characterization of this model given in one of the two variants of SWSL: SWSL-FOL based on First- Order Logic or SWSL-Rules based on Logic programming. The resulting ontologies are called: FLOWS - First-Order Logic Ontology for Web Services, which relies on First-Order Logic semantics, and ROWS - Rule Ontology for Web Services, which relies on Logic Programming semantics. Service Descriptors are the components of FLOWS ontology which provide basic information about a service. They include information like name, textual description, version, *etc*, which are properties inherited from the OWL-S Profile.

The Semantic Web Services Language (SWSL) is a language for describing Web services concepts and descriptions of individual services in a formal way. SWSL comes in two variants based on two well-known formalisms: First-Order Logic and Logic Programming. The two sub-languages are SWSL-FOL and SWSL-Rules. The design

of both languages was driven by compliance with Web principles, like usage of URIs, integration with XML built-in types and XML-compatible namespaces, and import mechanisms. Both languages are layered languages where every layer includes a number of new concepts that enhance the modeling power of the language. SWSL-Rules is a logic programming language which includes features from Courteous logic programs [23], HiLog [252] and F-Logic [144], and can be seen as both specification and implementation language. SWSL-Rules language provides support for service-related tasks like discovery, contacting, policy specification, *etc.*

2.4.5 Comparative Analysis of the Existing Frameworks

Comparison of all the prevalent and existing frameworks can be done on various parameters which includes mediation techniques followed, languages used, semantic functionalities provided *etc.* In the coming section four type of analysis are provided, first is a Matrix of Features and Approaches in WSMO, OWL-S and WSDL-S as provided by Seth [93], second comparison is based on languages used as provided by Cardoso [92] and third on the basis of common features shared by all frameworks [45] and finally our observations and comparative analysis of all the approaches has been provided in the last section.

a) *Matrix of Features and Approaches*

This comparison is based on the following features [93]:

i) Viewpoint - provider vs. requester

ii) Mediation - handling heterogeneity between data and process models

iii) Non-functional properties - additional information about aspects that may affect services usage

iv) Grounding - how services descriptions relate to Web Services standards

v) Availability of execution environments - how do SWS get used

Table 2.3: Comparative Analysis of WSMO, OWL-S and WSDL-S [93]

Approach	**Supported Viewpoints**	**Mediation**	**Non-functional Properties**	**Grounding**	**Execution Environment**
OWL-S	Single modeling elements for both views	Does not treat heterogeneity as a modeling issue	Restricted to the Service Profile	Grounding of behavior to WSDL and data to XML	Described but details of implementation are unavailable
WSDL-S	Service provider view- same as with WSDL	Adopts the behavior of the ontology used to describe annotations	Agnostic	WSDL-S is a legal extension to WSDL and as such is directly grounded	Any WSDL compliant execution engine could be extended for WSDL-S
WSMO	-	Supports mediation of data and processes	Available to all WSMO elements	Grounding of behavior to WSDL and data to XML	Open source provides by WSMX

b) After going through all the existing frameworks, it has been observed that despite some fundamental conceptual differences, there are several *features common to these approaches* [45].

i) All frameworks seem to agree on a minimal subset of necessary aspects of semantic services annotations.

ii) General services classifications are common to all approaches.

iii) For successful services execution a notion of preconditions and post conditions is common to all approaches, on the level of either atomic operations or complex services interactions.

iv) As for the behavioral/protocol descriptions of a service, WSMO, OWLS, and SWSF all allow one to encode complex behavior and interaction patterns; in WSDL-S, this can probably be simulated to a limited extent by adding preconditions and effects to individual WSDL operations, for example by embedding WSDL-S into BPEL4WS.

v) Nonfunctional aspects (such as quality of service, cost, and availability) can be modeled in most approaches. WSMO, OWL-S, and SWSF all provide extensible sets of nonfunctional properties in their ontologies, whereas WSDL-S sees this as out of its scope and instead points to related WS-*standards.

c) *Comparative analysis of languages for providing semantics in Web services (Table 2.3)*

d) *A comparative analysis of all the frameworks is provided below:-*

i) WSMO's distinct features compared with the other approaches are mediators, a unique top-level concept in WSMO. In OWL-S and SWSF the aspect of resolving heterogeneities is not treated separately, rather it is dealt with by special kinds of services. OWL-S treating heterogeneity as an architectural issue, that is, mediators are not an element of the ontology but a part of the underlying Web service infrastructure.

ii) WSMO/WSML provides different layers of expressivity which allows rich definitions of Web services compared to OWL-S. OWL-S lags expressiveness as its approach is based on OWL which was not developed with the design rationale

Table 2.4: Comparative analysis of languages for providing semantics in Web services [45]

Proposal	**Comment**	**Semantic Language**	**Formalism**	**Relation with WSDL**
OWL-S Profile Model	An OWL based upper ontology for Web services	OWL	Description Logics	Defines connectivity to WSDL vial grounding Models but overlaps in defining inputs, outputs and operations exist.
WSDL-S	Uses extensibility elements in WSDL to annotate elements with terms in externalized ontologies	Agnostic to ontology languages (can work with OWL,WSMO, UML, XML or any other modeling language)	Agnostic to ontology language. Therefore users are at will to pick a formalism of their choice	Specific annotations directly in WSDL as extensibility elements.
WSMO	A Web service Modeling Ontology expresses using WSML	WSML	Description Logics, First-Order Logic and Logic Programming (F-Logic)	Defines connectivity to WSDL vial grounding Models but overlaps in defining inputs, outputs and operations exist.

in mind to define the semantics of processes that require rich definitions of their functionality. SWSF is an attempt to extend the work of OWL-S to incorporate a variety of capabilities not within the OWL-S goals. A difference between FLOWS - the ontology part of SWSF and OWL-S is the expressive power of the underlying language. FLOWS is based On first-Order logic, which means that it can express considerably more than can be expressed using, for example, OWL-DL. The use of First-Order logic enables a more refined approach than possible in OWL-S to representing different forms of data flow that can arise in Web services.

iii) WSMO applies the principle of separation of concerns between communication and cooperation by introducing the concepts of Web services choreography and orchestration. They make the conceptual modeling and the interaction between the inputs and outputs more clear. In OWL-S interaction between the inputs and outputs are specified as OWL classes and the logical expressions in the respective languages in OWL-S which is not clear. FLOWS, as followed in SWSF, tries to explicitly model more aspects of Web services than OWL-S. FLOWS can readily model process models using a variety of different paradigms and data flow between services, through message passing.

iv) Major drawback of SWSF is that FLOWS does not use URIs to specify their concepts. SWSF seems to tackle both conceptual modeling and language issues but it is not clear how all the paradigms work together in this appraoch.

v) WSDL-S does not provide conceptual model for the description of Web services and their related aspects, but rather being a bottom up approach (annotating existing standards with metadata) provide complete solution to the integration problem. WSDL-S can actually be used to represent a grounding mechanism for WSMO.

vi) An important feature of WSDL-S is that it can even use annotations in WSML for defining conditions or references to WSMO descriptions. WSDL-S simply adds some useful attributes to WSDL's XML tags in order to reference semantic annotations. All the referenced concepts such as message types, operation reconditions, effects, and services categories can likewise be annotated in WSML/WSMO.

After going through the comparative analysis of various frameworks for semantic

Web services, focus shifts to measures which can be used to find semantic similarity between terms in Web services. Next section discusses various Measures of Semantic Relatedness (MSRs) used to calculate the semantic association between terms in the services.

2.5 Measures of Semantic Relatedness (MSRs)

Measuring the similarity between implicit semantic relations is an important task in information retrieval and natural language processing. Measures of Semantic Relatedness (MSRs) are statistical methods for extracting word associations from text corpora. Semantic measures can also be defined between lexically expressed word senses, or between whole texts [100]. Accurate measurement of relational similarity is an important step in numerous natural language processing tasks such as identification of word analogies, classification of noun-modifier pairs, *etc.*

2.5.1 Features of Measures of Semantic Relatedness

Some characteristics that are desirable for a semantic measure to be used in current Semantic Web applications are [100] :-

i) Domain independence: Nowadays, an increasing amount of online ontologies and semantic data is available on the Web, thus, enabling a new generation of semantic applications [62]. To develop such kind of domain independent applications, one has to deal with increasing heterogeneity, without establishing in advance the ontologies to be accessed.

ii) Universality: Semantic measures, in the highly dynamic context of the Web, must be flexible and general enough to be used independently of their final purpose,

and without relying on specific lexical resources or knowledge representation languages.

iii) Maximum coverage: It is assumed that, in the context of Web applications with no predefined domain, maximum coverage of possible interpretations of the words must be warranted. If one is limited to a particular knowledge source, such as WordNet, or a certain set of ontologies, one is constraining the scope of applications.

Two varieties of MSRs are vector-based and probability-based. Vector-based MSRs, such as Latent Semantic Analysis (LSA) and Generalized Latent Semantic Analysis (GLSA) [87] have the capability to measure relatedness between multi-word terms. According to Veksler [249] vector-based MSRs have non-incremental vocabularies based on limited corpora and these measures traditionally require preprocessing steps of creating a word-by-document or a word-by-word matrix, and dimensionality reduction. For example, LSA creates a word-by-document matrix, and reduces dimensions using Singular Value Decomposition (SVD). Since SVD is computationally infeasible for sufficiently large matrices, LSA is limited by both the size of the lexicon, and the size of the corpus. GLSA avoids the corpus-size problem by using a word-by-word matrix, but it is still very computationally expensive to create a GLSA vector-space for all possible words. These techniques require complete corpus re-processing when even a single document is added to the corpus, and thus cannot be used in combination with large dynamic corpora such as the World Wide Web.

Probability-based MSRs, such as Point wise Mutual Information (PMI) [177] and Normalized Google Distance (NGD) [187] are easily implemented on top of search engines (like Google search) and thus have a virtually unlimited vocabulary.

2.5.2 Probability base MSRs

Normalized Google Distance (NGD)

Normalized Google Distance (NGD) is a MSR that uses the probabilities of search terms extracted from the text corpus in question. It analyzes all features automatically through Google searches of the most general background knowledge data base: the World-Wide-Web. Determining the NGD between two Google search terms does not involve analysis of particular features or specific background knowledge of the problem area. [187].

Every text corpus or particular user combined with a frequency extractor defines its own relative frequencies of words and pharses usage. In the World-Wide-Web and Google setting there are millions of users and text corpora, each with its own distribution. Since number of Web pages currently indexed by Google is approaching 10^{10}, any single term searched on Google returns millions of Web pages. Since number of pages returned by Google is so vast, and the number of Web authors generating Web pages is so enormous it can be considered as truly representative very large sample from humankind. The probabilities of Google search terms, conceived as the frequencies of page counts returned by Google divided by the number of pages indexed by Google, approximate the actual relative frequencies of those search terms as actually used in society. Based on this premise, the relations represented by the Normalized Google Distance (Eq. 2.1) approximately capture the assumed true semantic relations governing the search terms. Mathematically

$$NGD(x,y) = \frac{max(logf(x), f(y)) - log(f(x,y))}{logM - min(logf(x), logf(y))} \tag{2.1}$$

where M is the total number of Web pages searched by Google; f(x) and f(y) are

the number of hits for search terms 'x' and 'y', respectively; and f(x, y) is the number of Web pages on which both 'x' and 'y' occur.

Pointwise Mutual Information (PMI)

Pointwise mutual information (PMI) (or specific mutual information) is a measure of association used in information theory and statistics [176]. The PMI of a pair of outcomes x and y belonging to discrete random variables quantifies the discrepancy between the probability of their coincidence given their joint distribution versus the probability of their coincidence given only their individual distributions and assuming independence. Mathematically,

$$pmi(x;y) = log(p(x,y)/p(x)p(y)) \tag{2.2}$$

The measure is symmetric ($SI(x,y) = SI(y,x)$). It is zero if 'X' and 'Y' are independent, and equal to -log($p(x)$) if 'X' and 'Y' are perfectly associated. SI(x, y) will increase if p($x|y$) is fixed, but p(x) decreases.

2.6 Methods of Content Retrieval

Information content methods, also known as corpus based methods measure the difference in information content of two words as a function of their probability of occurrence in a corpus. The method first proposed by Resnik says that similarity of two words is equal to information content (IC) of the least common subsumer [173]. Jiang [102] and Conrath and Lin [104] have developed measures that scale the information content of the subsuming concept by the information content of the individual concepts. Lin does this via a ratio, and Jiang and Conrath with a difference. Similar edge-counting

based methods were also applied on existing knowledge repositories such as Roget's Thesaurus [143] or WordNet [30] to compute the semantic relatedness.

Gloss based methods define the relatedness between two words as a function of gloss overlap. Banerjee and Pedersen have proposed the method that computes the overlap score by extending the glosses of the words under consideration to include the glosses of related works in a hierarchy [206]. Many of these measures were initially defined using the context of the WordNet ontology [77].

2.6.1 Various Knowledge bases used as Corpus

Many semantic measures have been proposed in the past to compute degrees of relatedness among words, texts or concepts. Degree of relatedness can always be measured in terms relation which exists between two words in a corpora. The knowledge base used for measuring the degree of relatedness can be a simple dictionary, a pre defined text corpora, Word net or Word Wide Web. Most of traditional methods to compute semantic measures exploit particular lexical resources: corpus, dictionaries, or well structured taxonomies such as WordNet. Some of them explore path lengths among nodes in taxonomies. Others exploit glosses (textual descriptions of concepts) in dictionaries, while some groups rely on annotated corpora to compute information content. Various corpora have been used as knowledge base. A brief review of some corpora is provided in the coming section.

Using Thesauri and Other Lexical Resources as knowledge base

Some very preliminary approaches calculated the similarity between two words on the basis of the number of edges in the term hierarchy created by indexing of articles. Rada *et al.* suggest that the assessment of similarity in semantic networks can in fact be thought of as involving just taxonomic (IS-A) links, to the exclusion of other link

types [199] .

Some groups have proposed to use the description of words present in dictionaries [148] and techniques such as LSA [212] to compute semantic relatedness. Kedad and Mietais propose using metadata, including a linguistic dictionary, in a hierarchical structure with no distance set by the user [274]. In their model, values are considered close if they belong to the same class. The semantic distance is fixed by the dictionary definition. They employ examples showing how classes of colours can be classified in terms of the various terms for shades of primary colours.

Li *et al.* combined structural semantic information from a lexical taxonomy and information content from a corpus in a nonlinear model [127]. They proposed a similarity measure that uses shortest path length, depth and local density in a taxonomy. Their experiments reported a Pearson correlation coefficient of 0:8914 on the Miller and Charles [78] benchmark dataset. In other work Lin [104] defined the similarity between two concepts as the information that is in common to both concepts and the information contained in each individual concept.

Weinstein and Birmingham measure syntactic correspondence between definitions of pairs of terms. Their work deals with artificial ontologies and not real world complexities as in the context of real-world applications, it is not possible to calculate the meaning of a term [180].

Manually compiled taxonomies such as WordNet [30] and large text corpora have been used in previous works on semantic similarity [102], [104], [174]. WordNet is a lexical reference system that was created by a team of linguists and psycholinguists at Princeton University. WordNet may be distinguished from traditional lexicons in that lexical information is organized according to word meanings, and not according to word forms. As a result of the shift of emphasis toward word meanings, the core

unit in WordNet is something called a synset. Synsets are sets of words that have the same meaning, that is, synonyms. A synset represents one concept, to which different word forms refer. For example, the set 'car, auto, automobile, machine, motorcar' is a synset in WordNet and forms one basic unit of the WordNet lexicon. Although there are subtle differences in the meanings of synonyms, these are ignored in WordNet.

Resnik proposed a similarity measure using information content. He defined the similarity between two concepts C1 and C2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C1 and C2 [174]. Then the similarity between two words is defined as the maximum of the similarity between any concepts that the words belong to. He used WordNet as the taxonomy; information content is calculated using the Brown corpus. Richardson and Smeaton combine the lexical database WordNet with Resnik's measure of similarity to give a semantic similarity measure that can be used as an alternative to pattern matching [198]. They use synsets (synonymous word forms), collocations (connected words) and a Hierarchical Concept Graph (HCG) with semantic pointers to hyponyms/hypernyms (is a/has a relationships) and meronyms/holonyms (part of/ has part relationships). Edges between concepts are given weights and the weight of a link is affected by the density of the HCG at that point, the depth in the HCG and the strength of connotation between the nodes. Instead of simply relying on the number of connecting edges, Leacock and Chodorow [32] have proposed to take the depth of the term hierarchy into consideration.

Veale proposed a relational similarity measure based on the taxonomic similarity in WordNet [243]. He evaluates the quality of a candidate analogy A:B::C:D (*i.e.* A to B as C to D), by comparing the paths in the WordNet, joining A to B and C to D. Relational similarity is defined as the similarity between the A:B paths and

C:D paths. Using a relational similarity measure, Turney proposed an unsupervised learning algorithm to extract patterns that express implicit semantic relations from a corpus [178]. His method produces a ranked set of lexical patterns that unambiguously describes the relation between the two words in a given word-pair. Patterns are ranked according to their expected relational similarity (i.e. pertinence).

Using Wikipedia as knowledge base

The advent of Wikipedia in 2001 has fulfilled the need for a more comprehensive knowledge base. Nowadays Wikipedia is rapidly growing in size, and it is not difficult to find new terms and named entities on it. Many techniques that use Wikipedia to compute semantic relatedness have been developed in the recent years. Some recent research efforts have focused on using Wikipedia to improve coverage with respect to traditional thesauri-based methods.

Strube *et al.* used some classic measures to use Wikipedia instead of WordNet as knowledge source, showing promising results [158]. They have used Wikipedia to determine semantic relatedness. Their results outperform those obtained using Word Net, hence showing the effectiveness of Wikipedia in determining the similarity between two words.

A further step in using Wikipedia is found in work by Gabrilovich and Markovitch where a technique called Explicit Semantic Analysis (ESA) is used to represent the meaning of words in a high dimensional space of concepts derived from Wikipedia [58]. They propose a method to represent the meaning of texts or words as weighted vectors of Wikipedia-based concepts, using machine learning techniques. According to their results, they provide even better correlation with human judgment than Wordnet. Chernov *et al.* [210] have suggested to make use of the links between categories present on Wikipedia to extract semantic information. Milne and Witten [50] have

proposed the use of links between articles of Wikipedia rather than its categories to determine semantic relatedness between words. Zesch *et al.* [244] have proposed to use Wiktionary, a comprehensive wiki-based dictionary and thesaurus for computation of semantic relatedness.

According to Kulkani [215], although Wikipedia has proven to be a better knowledge base than Word Net, many terms are still unavailable on Wikipedia, thus Wikipedia is still not comparable with the whole Web in the task of discovering and evaluation of implicit relationships. This has motivated the use of the whole Web as the knowledge base for calculating semantic relatedness.

Using Search engines as knowledge base

Regarding the Web as a live corpus has become an active research topic recently and much work has been carried out on measuring semantic similarity using Web content. Simple, unsupervised models demonstrably perform better when n-gram counts are obtained from the Web rather than from a large corpus [72]. Resnik and Smith extracted bilingual sentences from the Web to create a parallel corpora for machine translation [175]. Turney defined Point-wise Mutual Information (PMI-IR) measure using the number of hits returned by a Web search engine to recognize synonyms [178]. Matsuo et. al used a similar approach to measure the similarity between words and apply their method in a graph-based word clustering algorithm [267] by using Web hits for extracting communities on the Web. They measured the association between two personal names using the overlap (Simpson) coefficient, which is calculated based on the number of Web hits for each individual name and their conjunction (*i.e.*, AND query of the two names).

Chen *et al.* propose to exploit the text snippets returned by a Web search engine as an important measure in computing the semantic similarity between two words

[80]. Bollegala *et al.* has proposed a method that exploits page counts and text snippets returned by a Web search engine to measure semantic similarity between words [39]. Cilibrasi and Rudi developed the method that defines the relatedness between the words via Google Similarity Distance [187]. They use the World Wide Web as the database, and Google as the search engine. and proposed to compute the semantic relatedness using the Normalized Google Distance (NGD), in which they used Google^{TM} to determine how closely related two words are on the basis of their frequency of occurring together in Web documents.

Salahli *et al.* use the related terms of two words to determine the semantic relatedness between the words [130]. Some similarity measure are based on applications of fuzzy sets theory. Particularly, the new fuzzy similarity measure with better performance compared with conventional similarity methods have been proposed in [200].

After thoroughly reviewing the approaches followed by research community for semantic discovery of Web services and getting an insight into the important role of discovery in the entire lifecycle of the Web service a framework for semantic categorization of Web services has been proposed which is discussed in detail in the coming chapters.

Chapter 3

Annotation Process in 'ADWebS'

A technology framework is a collection of things. It can include one or more architectures, technologies, concepts, models, and even sub-frameworks.

3.1 Introduction

The framework named *'ADWebS'* (Automatic categorization and Discovery of Web Services Semantically) designed as the part of the thesis work, proposes the automatic discovery of Web services by semantic categorization of all available Web services into one of the predefined category(s). Discovering Web services involves three interrelated phases: i) matching, ii) assessment, and iii) selection. In first phase the service description provided by the developer is matched to that of a set of available resources. In second phase the result of matching (typically a set of ranked Web services) is assessed and filtered by a given set of criteria. In third phase, services are actually selected for subsequent customizing and combining with others [164]. Major focus of this thesis is on the first phase *i.e,* services discovery matching.

Entire framework has been divided into five steps i) Semantic Annotations addi-

tion ii) Annotations extraction iii) Finding Semantic relatedness iv) Categorization of services and v) Ranking.

To provide semantic discovery of Web services without the addition of ontologies, a new tag called '<annotation/ >' has been introduced in the existing service description file of every Web service where semantic metadata is added to this tag by the service provider (developer). Thus, before publishing the Web services in a registry, it is mandatory for service developer to add a set of terms ($n \geq 1$) which provide the functional description of the available services.

Once the semantic descriptions are published, *ADWebS* extracts these terms and add a Web service into one or more candidate category(s). Initially five candidate category *Zip Code, Country, Stock Market, Weather and Currency* have been considered and all available services have been categorized into one of these category.

Once the metadata is available and categories are determined, the degree of semantic relatedness between the terms of a service and candidate categories is determined by calculating the Normalized Google Distance (NGD), a Measure of Semantic Relatedness (MSRs) based on Google search engine. It is a semantic similarity measure derived from the number of hits returned by the Google search engine for a given set of keywords. Keywords with the same or similar meanings in a natural language sense tend to be 'close' in units of Google distance, while words with dissimilar meanings tend to be farther apart.

Addition of service to one or more candidate category has been done using a statistical measure Kruskal Wallis test, a non-parametric test. Based on the median values achieved for each category after applying the test, a service is permanently allocated to one of the predefined category(s). Service are ranked according to the user's query. A detailed description of all these steps is provided in the coming sections.

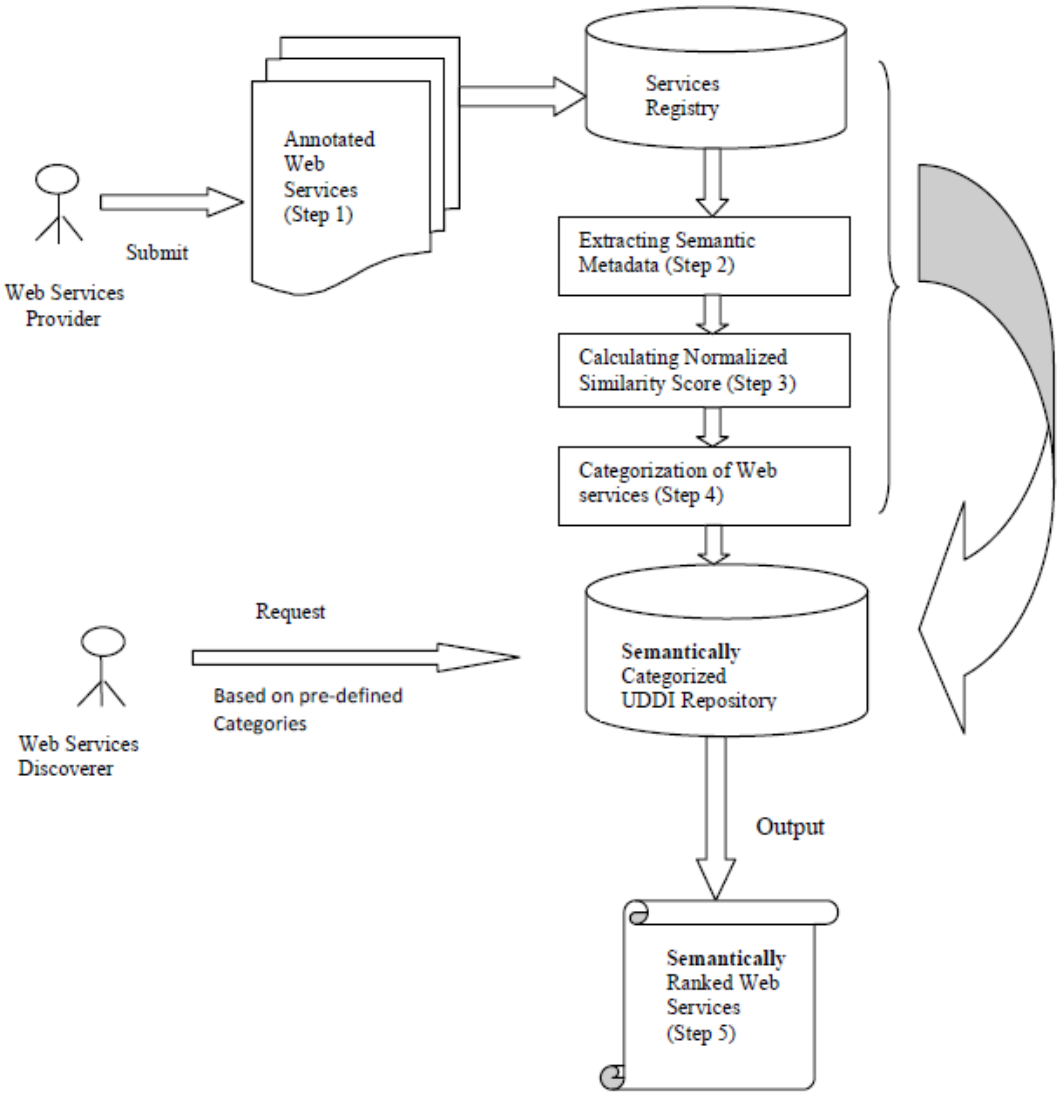

Figure 3.1: Architectural View of Proposed Framework *'ADWebS'*

3.2 Metadata and Annotations

Metadata is data that describes information about a piece of data, thereby creating a context in terms of the content and functionality of that data. Meaningful use of any data requires knowledge about its organization and content. Metadata is contextual information that establishes relationships between the data and the real world aspects it applies to, for example, a digital image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.

Broadly speaking, there are three kinds of metadata - syntactic, structural and syntactic metadata:

The simplest form of metadata is *syntactic* metadata. It describes non contextual information about content and provides very general information, such as the document's size, location, or date of creation. Syntactic metadata attaches labels or tags to data.

Structural metadata provides information about the organization and structure of some data, e.g. a Document Type Definition (DTD) is used to define the legal building blocks of an XML document. It lists the elements, attributes, and entities in a document and it defines the relationships between the different elements and attributes.

Semantic metadata adds relationships, rules, and constraints to syntactic and structural metadata. This metadata describe contextually relevant or domain-specific information about content based on a domain specific metadata model or ontology, providing a context for interpretation. In a sense, it captures a meaning associated with

the content.

Annotations

Associating metadata with resources (audio, video, structured text, unstructured text, web pages, images *etc*) is called annotation and *semantic annotation* means annotating resources with semantic metadata. Semantic annotations can be coarsely classified as being *formal or informal*. Formal semantic annotations follow representation mechanisms, drawing on conceptual models represented using well-defined knowledge representation languages. Such machine processable formal annotations on Web resources can result in vastly improved and automated search capabilities, unambiguous resource discoveries, information analytics, *etc*.

In software programming, annotations are used mainly for the purpose of expanding code documentation and comments. A markup language (XML/HTML) is a modern system for annotating a text in a way that is syntactically distinguishable from that text.

3.3 Semantic Annotations in Web Services

A Web service description specifies the properties of a Web service by metadata and semantic description does it by semantic metadata that is ontology-based or domain specific and connects a Web service property with the corresponding concept defined in an ontology or knowledge base. They are the means to bridge this gap in the form of semantic metadata describing the semantics of an item of information in a machine interpretable format.

Semantically annotating a Web service implies explicating the exact semantics of the Web service data and functionality elements that are crucial towards the use of the

Web service. The purpose of annotating Web services is to enable unambiguous and automated service discovery and composition. For example, two Web services meant for completely different functionalities may use the same data types and names for their operations, inputs and outputs, thus making the interpretation of their functionality ambiguous.

All of these Semantic Web Services research efforts try to overcome the drawback of lack of semantics of the current Web services standards and support automatic Web service discovery, invocation, and composition.

3.3.1 Aspects of Annotating Web Services

Various aspects of annotating Web services relevant in the context of Web services are [92]:

i) Which all information sources containing information relevant to the specification of Web services exist?

ii) Which descriptions are needed to manage and reuse Web services?

iii) How should the metadata be represented?

iv) How can the description be exploited?

Sources of information for annotation in Web services

Web services properties are described by various information sources of different quality. Most important source to extract the information from a service is *'documentation'* of a Web service. The documentation contains selected information about a Web service which can be analyzed to get the insight into meaning of the service. Major difficulty in extracting such information is that entire documentation is defined

in natural language and not formalized, *i.e.* it is not machine understandable. Determining actual properties of a Web service from such text becomes difficult or even impossible for a machine.

Web service's *'source code'* can serve as another information source as it contains all information about operations including all inputs and outputs but the extractable information differs between programming languages. Cardoso [92] points out that source code defines programming logic which contains information about the Web service's functionality, the Web service's preconditions that must be fulfilled before the execution of an operation, and effects of the execution of an operation of the service.

Another important source of information is *'comments'* included in the source code which can be used to determine information about operations as well as about the functionality of the whole Web service. Problem with comments is that they inherit the pros and cons of documentations and API-descriptions. If the source code is not accessible, the comments are also not accessible.

Above mentioned shortcomings clearly indicate that there are various sources where annotations can be provided in a Web services, but none of these sources serve the purpose of providing meaningful data. There are various important aspects like which data to fetch, how to use the fetched data, *etc.* which need to be considered for fetching useful information from a Web service. To overcome these bottlenecks *ADWebS* proposes annotation in WSDL files.

3.3.2 Approaches for extraction of metadata in Web services

The *word extraction approach* followed for capturing semantics in Web services is to extract all the terms from the WSDL and then pre-processing the extracted terms. In such approaches all pre-processing steps detagging (removing all the tags from the

XML document) tokenizing (breaking the stream of text into distinct meaningful units or tokens), stop word removal (discarding of certain key phrases and words that may otherwise taint the algorithms) and stemming (reduction of words into their root) have to be followed [223].

Finally, the extracted terms of every pre-processed WSDL are represented as a vector $v = (e_0, e_1..., e_n)$. Each element e_i represents the importance of a distinct term t_i for that document. This importance is calculated according to the selected word weighting method, TF-IDF, a word weighting heuristic that determines whether a word is important for a document if it occurs very often. For each term t_i of a document 'd' weight can be calculated by using the formula:

$$TermWeight(w_i) = tf_i * log[D/df_i] \tag{3.1}$$

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

df_i = document frequency or number of documents containing term 'i'

D = number of documents in the database.

Thus all the WSDL files were preprocessed using the following steps:

(i) First argument declarations and comments from each WSDL document are extracted.

(ii) Tokens are generated using the tokenizer.

(iii) Stop words are removed from the file. A text file "stopword.txt" containing list of stop words is given as a input to the toolkit. Based upon this text file all stop words are removed.

(iv) Stemming is done to bring the words to their base words. Porter stemmer is used for this process.

(v) TF-IDF is calculated. From this TF-IDF matrix the words with high importance were extracted out [223].

3.3.3 The Annotation Process in *ADWebS*

The annotation process in *ADWebS* starts with addition of semantic metadata to the existing WSDL files in a new tag called 'annotations' at the starting of a Web Service. It is mandatory for every service publisher to give a set of 'n' terms, N = $\{n_1, n_2, n_3, \ldots\}$ (semantic metadata) where n_1, n_2, n_3, *etc.* are the terms providing the functional capabilities of the Web service.

A Web service providing information about *'Weather'* conditions of a city may include the metadata related to Weather in a WSDL file as shown below:

⟨?xml version = '1.0" encoding ="utf-8"?⟩⟨wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/″

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:s="http://www.w3.org/2001/XMLSchema"

xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:tns="http://ws.fraudlabs.com/"

xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"

xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"

targetNamespace="http://ws.fraudlabs.com/"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"⟩

⟨**annotation**⟩

⟨**term** ⟩ city, temperature, pressure, humidity, precipitation, visibility, clouds, wind speed, wind direction, longitude, latitude, elevation ⟨/**term**⟩

⟨/**annotation**⟩

⟨*wsdl* : *documentation* xmlns:wsdl= "http://schemas.xmlsoap.org/wsdl/″

⟨*wsdl* : *types*⟩

Similarly a Web service providing information about 'Zip code' may include the metadata

⟨**annotation**⟩

⟨**term**⟩latitude, longitude, time zone, city, state, country⟨/**term**⟩

⟨/**annotation**⟩

A Web service providing information about 'Stock Market' may include the metadata

⟨**annotation**⟩

⟨**term** stock, stock exchange, market, ticker, exchange rate, share⟨/**term**⟩

⟨/**annotation**⟩

The purpose of adding a new tag is to ensure that the input data does not have heterogenous representation and a standard methodology is followed for efficient description of semantic metadata. Once the services publishers provides the most useful terms in the annotation tag, the next important step is to *extract* those terms and assign the Web service into one of the pre-defined category(s).

3.4 Information Extraction

Information extraction (IE) is a technology based on analyzing natural language in order to extract snippets of information. The process takes texts (and sometimes

speech) as input and produces fixed format, unambiguous data as output. This data may be used directly for display to users, or may be stored in a database or spreadsheet for later analysis, or may be used for indexing purposes in information retrieval (IR) applications such as internet search engines like Google. According to Moens [138]

'Information extraction is the identification, and consequent or concurrent classification and structuring into semantic classes, of specific information found in unstructured data sources, such as natural language text, making the information more suitable for information processing tasks.'

Information extraction pre supposes that although the semantic information and its linguistic organization is not immediately computationally transparent in a text, it can nevertheless be retrieved by taking into account surface regularities that reflect its computationally opaque internal organization. An information extraction system will use a set of extraction patterns, which are either manually constructed or automatically learned, to take information out of a text and put it in a more structured format. The use of the term *extraction* implies that the semantic target information is explicitly present in a text's linguistic organization, *i.e.*, that it is readily available in the lexical elements (words and word groups), the grammatical constructions (like phrases, sentences or temporal expressions), *etc* [138].

3.4.1 Extracting Metadata in *ADWebS*

Once service provider has provided semantic metadata in the annotation tag of Web service, *ADWebS* extracts the metadata and process of finding semantic relatedness between extracted terms starts. Terms are extracted from annotated WSDL document and term category matrix is formed (*e.g.* the terms extracted from annotation tag in example discussed above are 'city, temperature, pressure, humidity, windspeed ').

Since there is no limit on number of terms in the annotation tag, service publisher has full liberty to add all the possible terms which can best describe the functional capability of the Web service. In fact, this will increase the probability of service being selected by group of user using different terminology for same terms.

Comparison between word extraction approach and approach used in *ADWebS*

Terms retrieved from both approaches were considered and it was analyzed that terms retrieved from word extraction approach had many terms which have no specific functional relevance to the WSDL file being considered *i.e.,* many terms did not provide any contextual information about the service being considered.

For example, Consider the Web service in Appendix 1 for determining 'Zip Code'of cities in USA. Using semiautomated approach the terms available in the WSDL file of this service are extracted which contains terms like

'credits available', 'household per zipcode', 'white population', 'black population', 'day light saving', 'MSA2000', *etc.*

Such terms have no relevance to actual functional capability of the service, i.e. they are not closely associated to term Zip Code in any way. If we calculate semantic relatedness of such terms with category *Zip Code*, instead of improving the semantic score it will decrease the probability of service being added to category *Zip Code*, where it should be actually placed.

Based upon this observation, work in the entire thesis is focused on manual annotation approach only. Since the entire semantic categorization process is depend on the terms retrieved from the services, it was realized that manually annotating Web services by the service publisher outperforms the word extraction approach.

As discussed in chapter 2 there are two types of Measures of Semantic Related-

ness (MRSs) vector-based and probability-based. A comparative analysis of all three approaches *i.e.* keyword based, vector based and probability based MSRs was done to find which performs the best. Wordnet was considered as representative for vector based MSRs and Normalized Similarity Score (NSS) as measure for probability based MSRs. Three algorithms have been designed to find the comparative score of all these approaches are discussed in the coming section.

3.4.2 Correlation between Query terms and terms in information domain

Let Q_t be the set of terms in the user's query and $I = T_1, T_2, T_3, , T_r$ denote the set of all terms in information domain where T_1 represents set of terms retrieved from first file, T_2 is the set of terms extracted from second file and so on. The correlation between the Information domain's term (T_r) and Query terms (Q_t) is calculated using three different approaches.

The *first* method is to find the simple term matching between Q_t and T_r, and if the term matches exactly with query term the score is 1, called Syntactic Similarity Score (SSS) (Algorithm 1).

Time complexity of Algorithm 1(Table 3.1)

Analysis

The running time of the algorithm is: $\sum(costofstatement)\times$(No. of times statement is executed)

Let T(n) be the running time so:

$$T(n) = c1 * 1 + c2 * (n + 1) + c3 + c4 * ($$

```
   Input: Count:Total number of records in the database
   Output: SSS (Syntactic Similarity Score)
 1 r = 1;
 2 while (r ≤ Count) do
 3 |   N=T_r;
 4 |   foreach Keyword (K[i]inQuery) do
 5 |   |   j=0;
 6 |   |   foreach term(Tr[j]) do
 7 |   |   |   if K[i] = T_r[j] then
 8 |   |   |   |   SSS= 1;
 9 |   |   |   |   j = j+1;
10 |   |   |   end
11 |   |   |   i = i+1;
12 |   |   end
13 |   end
14 |   r = r+1;
15 end
```

Algorithm 1: Algorithm to find relatedness between Query terms and terms in information domain using Syntactic Similarity Score (SSS)

$$\sum_{j=0}^{n} k_j) + \text{c5}*(\sum_{j=1}^{n}(k_j-1)) + \text{c6}*(\sum_{i=1}^{n}\sum_{j=1}^{k} t_{ij}) + \text{c7}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1) + \text{c8}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1)$$

$$+ \text{c9}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1) + \text{c10}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1) + \text{c11}$$

The running time depends on the values of k_j and t_{ij}. These vary according to the input.

Best Case: If keywords present in the each corresponding row is single and term to be searched is also single

i.e. $k_j = 1$

$t_{ij} = 1$

Then

$T(n) = \text{c1} + \text{c2(n+1)} + \text{c3}*n + c4 * n + c6 * n + c11 * n$

Table 3.1: Time complexity of Algorithm 1

Sr.No.	Input	Cost	Time
1.	Set $r = 1$	C1	1
2.	While($r \leq Count$)	C2	n+1
3.	Set N=T_r	C3	n
4.	ForEachKeyword (K[i]$inQuery$)	C4	$\sum_{j=0}^{n} k_j$
5.	Set j=0	C5	$\sum_{j=1}^{n} (k_j - 1)$
6.	ForEach$term$(Tr[j])	C6	$\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij}$
7.	If $K[i] = T_r[j]$	C7	$\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} - 1$
8.	Set SSS= 1	C8	$\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} - 1$
9.	Set $j = j$+1	C9	$\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} - 1$
10.	End	0	0
11.	Set $i = i$+1	C10	$\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} - 1$
12.	End	0	0
13.	End	0	0
14.	Set $r = r$+1	C11	n
15.	End	0	0

$= (c2+c3+c4+c6+c11)*n + c1 + c2$

Expressing $T(n)$ as $a * n$ +b for constants a and b (that depend on the statement costs c_i)$\Longrightarrow$ $T(n)$ is a linear function of n.

$T(n) = \Theta(\mathbf{n})$

Worst Case: If keywords present in the each corresponding row is 'n' (which is equal to total no. of records i.e. count) and term to be searched is also 'n' i.e.

$k_j = n$

$t_{ij} = n$

Then

$T(n)$ = c1 + c2(n+1) + c3$*n + c4 * n^2 + c5 * n^2 + c6 * n^3 + c7 * n^3 + c8 * n^3 + c9 * n^3 + c10 * n^3 + c11 * n$

$= (c6 + c7 + c8 + c9 + c10) * n^3 + (c4 + c5) * n^2 + (c2 + c3) * n + (c1 + c2)$

Express T (n) as $a * n^3 + b * n^2 + c * n +$ d for constants a, b, c, d (that depend on the statement costs ci)$\Longrightarrow$ T (n) is a function of n^3.

So

$T(n) = \Theta(n^3)$

Second method is to use WordNet as a metric to calculate the distance between Q_t and T_r and this distance is represented as WordNet Similarity Score (WSS), as referred in Algorithm 2. After finding the similarity of each query term with each term of WSDL file a normalized value is obtained by dividing the total sum of all values with total terms in the annotation tag. The CALCULATE function used here is the actual value or similarity distance achieved between Q_t and T_r. The Wordnet vector measure is used to calculate similarity used in Algorithm 2.

Time complexity of Algorithm 2(Table 3.2)

Analysis

The running time of the algorithm is: $\sum$(cost of statement)$\times$(No. of times statement is executed)

Let T(n) be the running time so:

$$\text{T(n)}=c1*1+c2*(n+1)+c3*n+c4*(\sum_{j=0}^{n} k_j) + \text{c5}*(\sum_{j=1}^{n}(k_j-1)) + \text{C6}*(\sum_{j=1}^{n}(k_j-1))+$$
$$\text{c7}*(\sum_{i=1}^{n}\sum_{j=1}^{k} t_{ij}) + \text{c8}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1)+ \text{c9}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1) + \text{c10}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1) +$$

Input: Count:Total no. of records in the database; K: Total no. of terms in User's Query
Output: WSS (Wordnet Similarity Score)

```
1  r=1;
2  while (r ≤ Count) do
3      N=T_r;
4      foreach Keyword (K[i] in Query) do
5          j=0;
6          WSS=0;
7          foreach term (Tr[j]) do
8              V = CALCULATE (K[i],Tr[j]);
9              WSS= WSS+V;
10             j = j+1;
11         end
12         WSS_temp = (WSS_temp + WSS)/N;
13         i = i+1;
14     end
15     WSS = WSS_temp/K;
16     r = r+1;
17 end
```

Algorithm 2: Algorithm to find relatedness between Query terms and terms in information domain using WordNet Similarity Score (WSS)

$$\text{c11}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1)+ \text{c12}*(\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij}-1)+\text{c13}*n + c14*$$

The running time depends on the values of k_j and t_{ij}.

Best Case: If keywords present in the each corresponding row is single and term to be searched is also single, i.e.

$k_j = 1$

$t_{ij} = 1$

Then

$T(n) = \text{c1} + \text{c2(n+1)} + \text{c3}*n + c4 * n + c7 * n + c13 * n + c13 * n$

$= (\text{c2+c3+c4+c7+c13+c14})*n + c1 + c2So$

$T(n) = \Theta(n)$

Worst Case:

Table 3.2: Time complexity of Algorithm 2

Sr.No.	Input	Cost	Time
1.	Set $r = 1$	C1	1
2.	While($r \leq Count$)	C2	n+1
3.	Set N=T_r	C3	n
4.	ForEachKeyword (K[i]$inQuery$)	C4	$\sum_{j=0}^{n} k_j$
5.	Set j=0	C5	$\sum_{j=1}^{n}(k_j - 1)$
6.	Set WSS=0	C6	$\sum_{j=1}^{n}(k_j - 1)$
7.	ForEach$term$(Tr[j])	C7	$\sum_{i=1}^{n}\sum_{j=1}^{k} t_{ij}$
8.	V = CALCULATE (K[i],Tr[j])	C8	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
9.	Set WSS= WSS+V	C9	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
10.	Set $j = j$+1	C10	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
11.	End	0	0
12.	Set WSS_{temp} = (WSS_{temp} + WSS)/N	C11	$\sum_{j=1}^{n}(k_j - 1)$
13.	Set $i = i$+1	C12	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
14.	End	0	0
15	$WSS = WSS_{temp}/K$	C13	n
16.	Set $r = r$+1	C14	n
17.	End	0	0

If keywords present in the each corresponding row is 'n' (which is equal to total no. of records i.e. count) and term to be searched is also 'n' i.e.

$k_j = n$

$t_{ij} = n$

Then

$T(n) = c1 + c2(n+1) + c3 * n + c4 * n^2 + c5 * n^2 + c6 * n^2 + c7 * n^3 + c8 * n^3 + c9 * n^3 + c10 * n^3 + c11 * n^2 + c12 * n^2 + c13 * n + c14 * n$

$=$ (c7+c8+c9+c10)$*n^3$+($c4$+$c5$+$c6$+$c11$+$c12$)$*n^2$+($c2$+$c3$+$c13$+$c14$)$*n$+($c1$+$c2$)

Express T (n) as $a * n^3 + b * n^2 + c * n+$ d for constants a, b, c, d (that depend on the statement costs c_i)$\Longrightarrow$ T (n) is a function of n^3.

Then

$T(n) = \Theta(n^3)$

Third approach is to use a page-count-based similarity measure discussed in

Calculating the NSS of all terms in information domain with all terms in Qt means that semantic relatedness of each term with each query term has been found. As in algorithm 2, after finding the similarity of each query term with each term of WSDL file a normalized value is obtained by dividing the total sum of all values with total terms in the annotation tag. The CALCULATE function used here is the actual value or similarity distance achieved between Q_t and T_r. The Normalized Similarity Score (NSS) is used to calculate similarity used in Algorithm 3

Time complexity of Algorithm 3(Table 3.3)

Analysis

As discussed in Algorithm 2, best case and worst case time complexity of Algorithm are same i.e.

Best Case:

$T(n) = \Theta(n)$

Worst Case:

$T(n) = \Theta(n^3)$

All the three algorithms were run on the same data set comprising of 'n' terms and Average *precision-at-n* was calculated.

Input: Count:Total no. of records in the database; K: Total no. of terms in User's Query
Output: NSS (Normalized Similarity Score)

```
1  r=0;
2  while (r ≤ Count) do
3      N=Tr;
4      foreach Keyword (K[i]inQuery) do
5          j=0;
6          NSS=0;
7          foreach term (Tr[j]) do
8              V = CALCULATE (K[i],Tr[j]);
9              NSS= NSS+V;
10             j = j+1;
11         end
12         NSS_temp = (NSS_temp + NSS)/N;
13         i = i+1;
14     end
15     NSS = NSS_temp/K;
16     r = r+1;
17 end
```

Algorithm 3: Algorithm to find relatedness between Query terms and terms in information domain using Normalized Similarity Score (NSS)

After analyzing the precision achieved with SSS, WSS and NSS (Figure 3.2) it is clear that NSS gives the best results so semantic relatedness of extracted terms and pre defined category is determined by using Normalizes Similarity Score (NSS) through out the thesis. Next a thorough analysis was done to find that can ontology serve as a better option for finding semantic association.

3.4.3 Limitations of Ontologies and Wordnet

Existing measures of semantic relatedness rely either on ontologies and semantic networks or just raw text, hence majority of upcoming semantic Web services discovery frameworks use ontologies and some frameworks use Word net as knowledge corpus. Our initial focus was to study both ontologies and Wordnet to determining semantic relatedness between terms in a service and set of categories, but both had some major

Table 3.3: Time complexity of Algorithm 3

Sr.No.	Input	Cost	Time
1.	Set $r = 1$	C1	1
2.	While($r \leq Count$)	C2	n+1
3.	Set N=T_r	C3	n
4.	ForEachKeyword (K[i]$inQuery$)	C4	$\sum_{j=0}^{n} k_j$
5.	Set j=0	C5	$\sum_{j=1}^{n}(k_j - 1)$
6.	Set WSS=0	C6	$\sum_{j=1}^{n}(k_j - 1)$
7.	ForEach$term$(Tr[j])	C7	$\sum_{i=1}^{n}\sum_{j=1}^{k} t_{ij}$
8.	V = CALCULATE (K[i],Tr[j])	C8	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
9.	Set WSS= WSS+V	C9	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
10.	Set $j = j$+1	C10	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - 1)$
11.	End	0	0
12.	Set $WSS_{temp} = (WSS_{temp} + WSS)/N$	C11	$\sum_{j=1}^{n}(k_j - 1)$
13.	Set $i = i$+1	C12	$\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{i_j} - 1)$
14.	End	0	0
15	$WSS = WSS_{temp}/K$	C13	n
16.	Set $r = r$+1	C14	n
17.	End	0	0

drawbacks. Some of the problems encountered are:

Problems with Ontology

i) Major problem faced in ontology based systems is knowledge resources are often constructed for specific domains and general purpose ontologies do not offer adequate term coverage. So the entire concept of having providing semantic asso-

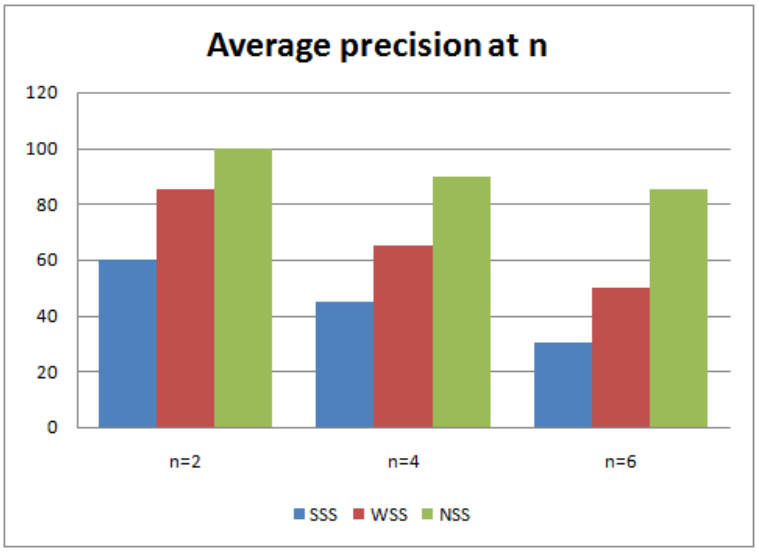

Figure 3.2: Average Precision-at-n for SSS, WSS and NSS

ciation of query term to any other term is simply given up *i.e.* they are confined to a particular domain.

ii) Hierarchy is defined by repetitive division of concepts as per their attributes in ontologies. The order in which these attributes are used to create the tree structure can result in dramatically different hierarchies. All the ontology-based measures of similarity capitalize on the property that words that occur close to each other in a hierarchy share a lot of attributes and are therefore similar. However, they usually rely on a fixed hierarchy. Word pairs that would be closer in variations of the hierarchy are not considered. Thus ontology-based measures are likely to wrongly assign a low semantic similarity value to such word pairs. As cited in [220] consider a scenario where the attributes a_1 and a_2 are used in different orders to create different hierarchies of the words w_1, w_2, w_3 and w_4. While w_1 and w_2 are closer to each other than w_1 and w_3 in hierarchy-1, it is the other way round in hierarchy-2. Thus variations in the order of use of attributes for creating the hierarchy can result in different sets of words being close to each

there. Thus an ontology based measure is likely to find them unrelated.

iii) Another significant drawback is that use and updating of resources, such as thesauri or ontologies, is a time consuming and tedious task, demanding human labor and often expert knowledge. Further, these methods cannot be applied for words or terms that are not included in the resource repository, *e.g.*, scientific terms, out of vocabulary words, neologisms.

After analyzing the problem of restricted domain usage of ontology based system, next step was to explore Wordnet as knowledge base.

The problems faced in Wordnet were:-

i) Word net have limited words set and such resources cannot serve the purpose of finding association of terms with each and every category.

ii) The major objective in our work is to compare each published annotated term to each pre defined category and since the vocabulary of Wordnet is limited this is just not possible.

According to St-Onge [77] using WordNet to compute semantic relatedness leads to various problems: missing connections of certain semantically related words, inconsistency in the semantic proximity implicit in links in WordNet, incorrect or incomplete word sense disambiguation *etc.*

Teich and Fankhauser [68] analyzed the problems with WordNet based methods in more detail: missing relations (only some relations across part-of speech are accounted for in WordNet), spurious relations (rather questionable ties are established without constraints on the length or branching factor of a transitive relation), sense proliferation (in some instances the sense-tagging appears to be overly specific, using synonymy

without repetition these senses does not form a link), and compound term problems (there might be some problem with sense-tagging with compound terms in WordNet).

At this stage it is evident that a huge database base like search engine is required to analyze semantic similarity of a term with a category since even the big domains like Wordnet cannot help in finding association of any term with any category. Next section provided a brief overview of MSRs and discusses the role of NSS in *ADWebS* for finding the semantic relatedness.

Chapter 4

Semantic Categorization and Discovery in '*ADWebS*'

One of the possible solutions for discovery of desired Web services is to categorize all the services, provided by different publishers in various registries, into their functional domains *i.e.* each Web service should be categorized into one of the pre-defined category. A federation of categories can be formed where each category may contain information about all services supported by it and list of all available categories. For example, a category related to general travel services will contain services related to air travel, rail travel, taxi services, hotel reservation, *etc.* and a search of the general travel category may return a referral to the requester pointing 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 federation will ease the discovery process as service requestor will have option to choose one of the desired category from the list of available categories.

Categorization of services into the general travel, rail travel, air travel, hotel reservation services may require sharing of a common taxonomy or ontology to avoid forwarding inappropriate queries. There are two possible solutions for this: first is that

the general travel category would discover the rail travel registry using a spider or index approach. An indexing engine could have come across a registry, and based on the information it harvested from the registry classified it as a rail travel under travel category. Second solution is that the rail travel registry publishes information to the general travel registry and using the shared taxonomy could classify itself under general travel category. However, in both cases, some machine interpretable information is required to fill the semantic gap between information available and the information required for assigning a service to a category.

Categorization is possible if the service description file contains some functional description of the service *i.e.* information about *'What a service does'*. The proposed framework accomplishes this task by making it mandatory for every service provider to add semantic metadata in the service description file of existing Web services. The set of terms which provide the contextually relevant information about the Web services must be added in newly created annotation tag in the existing WSDL file of a service. The set of categories can be fixed using some standard norms or they can be finalized by analyzing the services available in some of the most frequently used repositories. After surfing the freely available UDDI repositories like XMethods.com [263], BindingPoint [27], WebServiceX.NET [257], StrikeIron [228], , eSynaps [67], *etc.* the available Web services were downloaded and analyzed. Initially *five* predefined categories *'Weather', 'Currency', 'Stock Market', 'Country'* and *'Zip Code'* were considered as representative set of services for experimental purposes.

4.1 Measures of Semantic Relatedness (MSRs) in *ADWebS*

Measures of Semantic Relatedness (MSRs) are computational means for assessing the relative meaning of terms. More specifically, MSRs take the form of computer programs that can extract relatedness between any two terms based on large text corpora [249]. They quantify the degree in which some words or concepts are related, considering not only similarity but any possible semantic relationship among them.

Similarity measures can be broadly categorized into two types: Attributional similarity measures and Relational similarity measures. According to Turney [177]:

'Relational similarity is correspondence between relations, in contrast with attributional similarity, which is correspondence between attributes. When two words have a high degree of attributional similarity, we call them *synonyms*. When two pairs of words have a high degree of relational similarity, we say that their relations are *analogous*. For example, the word pair mason:stone is analogous to the pair carpenter:wood.'

Given a text corpus, individual words have more or less differing contexts around them. The context of a word is composed of words co-occurring with it within a certain window around it. Distributional measures use statistics acquired from a large text corpora to determine how similar the contexts of two words are. These measures are also used as proxies to measures of semantic similarity as words found in similar contexts tend to be semantically similar. This is known as the *distributional hypothesis* [110] and such measures are traditionally referred to as *measures of distributional similarity*.

According to Budanitsky and Hirst [5] if two words have many co-occurring words then similar things are being said about both of them and so they are likely to be

semantically similar. Conversely, if two words are semantically similar then they are likely to be used in a similar fashion in text and thus end up with many common co-occurrences. For example, the semantically similar 'bug' and 'insect' are expected to have a number of common co-occurring words such as 'crawl', 'squash', 'small', 'woods', *etc.* in a large text corpus.

Another measure called *measures of distributional relatedness* [82] is similar to measures of distributional similarity. These measures use raw text and co-occurrence information to determine semantic relatedness between two words. Certain word pairs have a special relation with each other. For example, 'strawberry' and 'cream', 'doctor' and 'scalpel', and so on. These words are not similar physically or in properties, but 'strawberries' are usually eaten with 'cream' and a 'doctor' uses a 'scalpel' to make an incision. The distributional measures that use simple co-occurrences capture semantic relatedness, as words that tend to occur together are likely to have large set of common co-occurring words [220].

The various senses of a word represent distinct concepts. Each of these concepts can usually be described by a number of attributes or features. These attributes may be physical descriptions like color, shape and composition or function, purpose and role. Two words are adjudged similar if they share a number of such attributes and if the strength of the shared attributes is high. By strength we mean how strongly an attribute helps define the words. The more prominent a shared feature, the more similar the two words are. Further, it is possible that words w_1 and w_2 are related as they share a certain set of attributes, while w_2 and w_3 are related because they share a different set of attributes. Thus w_1 and w_3 are likely not as related as w_1 and w_2, or w_2 and w_3. For example, the 'physical key' is closely related to the abstract 'password', as they are both means of getting access. 'Password' is closely related to 'encryption'

as they both pertain to data security. However, the 'physical key' has little to do with 'encryption' and the two are not so much related. Thus semantic relatedness is not necessarily transitive and may be a function of a subset of relevant attributes, not necessarily all [220].

4.1.1 Semantic Relatedness and Semantic Similarity

Semantic Relatedness

Semantic relatedness indicates degree to which words are associated via any type (such as synonymy, meronymy, hyponymy, hypernymy, functional, associative and other types) of semantic relationships. Determining the semantic relatedness between two words refers to computing a statistical measure of similarity between those words. The relatedness measures may use a combination of the relationships existing between words depending on the context or their importance. To illustrate difference between similarity and relatedness, Resnik [175] provides the widely used example of 'car' and 'gasoline'. These terms are not very similar; they have only few features in common. But they are more closely related in a functional context; namely that 'cars' use 'gasoline'.

Semantic Similarity

Semantic similarity is used when similar entities such as 'apples' and 'bananas' or 'table' and 'furniture' are compared [4], [5]. These entities are close to each other in an is-a hierarchy. The concept to semantic relatedness is closely related to the concept of semantic similarity. While there is some overlap in their meanings and they may be used interchangeably in certain contexts, there is a clear distinction between the two. For example, 'apples' and 'bananas' are hyponyms of 'fruit' and 'table' is a hyponym of 'furniture'. However, even dissimilar entities may be semantically related,

for example, 'door' and 'knob', 'tree' and 'shade', or 'gym' and 'weights'. In this case the two entities are not similar per se, but are related by some relationship. This relationship may be one of the classical relationships such as meronymy (is part of) as in 'door-knob' or a non-classical one as in 'tree-shade' and 'gym-weights'. Thus two entities are semantically related if they are semantically similar (close together in the is-a hierarchy) or share any other classical or non-classical relationships.

Another important concept related to semantic similarity and semantic relatedness is semantic distance. It has been traditionally used in the context of both semantic relatedness and semantic similarity. In the former context, it represents the inverse of semantic relatedness, while in the latter, it is the inverse of semantic similarity [220].

How semantic relatedness is calculated

Typically, automated systems assign a score of semantic relatedness to a given pair of words (target words) calculated from a relatedness measure. The absolute score is usually irrelevant on its own. For example, a relatedness score of 0.7 between 'a' and 'b', in a possible range of 0 to 1, does not imply that 'a' and 'b' are more related than the average word pair. However, given that the semantic relatedness of 'c' and 'd' is 0.6, the system can conclude that 'a' and 'b' are more related than 'c' and 'd'. Thus even though the absolute score given by a relatedness measure is not of much significance, it is important that the measure give a higher score to word pairs which humans think are more related and comparatively lower scores to word pairs that are less related. This ability to mimic human judgment of semantic relatedness has been used in numerous applications such as automated spelling correction, word sense disambiguation, thesaurus creation, information retrieval, text summarization, and identifying discourse structure [220].

Measures like semantic relatedness or semantic similarity are important measures

to find an association or co-relation that exists between pair of words, but the important issue is that what should be the backhand knowledge base on basis of which such calculations will be made. There are numerous knowledge sources like Wordnet, thesauri, Wikipedia, a well defined domain specific ontology, dictionary, WWW, *etc.* which can serve as background sources for comparing the semantic relatedness of two terms. Lindsay *et al.* [194] evaluated PMI and NSS on the various corpora like Project Gutenberg, Google, Enron, Wikipedia, New York Times, *etc.* NSS performed better than PMI on all but the New York Times corpus (mean NSS performance = 60.2%; mean PMI performance = 55.0%). So we have opted for NSS as MSR for finding semantic relatedness in the entire thesis. Next section gives a brief overview of how semantic association is found using Google as knowledge corpus.

4.1.2 Calculating Normalized Similarity Score in *ADWebS*

Page-count-based metrics like Google, use association ratios between words that are computed using their co-occurrence frequency in documents. The basic assumption of this approach is that high association ratios indicate a semantic relations between words [115].

Motivated by Kolmogorov complexity, Cilibrasi and Vitanyi [187] proposed a page-count-based similarity measure, called the Normalized Google Distance (NGD), defined as:

$$NGD(x,y) = \frac{max(logf(x), f(y)) - log(f(x,y))}{logM - min(logf(x), logf(y))} \tag{4.1}$$

where M is the total number of Web pages searched by Google; f(x) and f(y) are the number of hits for search terms 'x' and 'y', respectively; and f(x, y) is the number of web pages on which both x and y occur.

If the two search terms 'x' and 'y' never occur together, but do occur separately, the normalized Google distance between them is infinite, meaning they are not semantically related at all. In another extreme case where both terms always occur together, their NGD is zero, which is an indicator that they are semantically closely related. While theoretically the scope of NGD is $[0, \infty]$, experimental results show that most of the time the values fall between 0 and 1.

The main properties of NGD are

i) The range of the NGD is in between 0 and ∞ ;

   (a) If x = y or if x $\geq y$ but frequency M > f (x) = f (y) = f (x,y) > 0, then NGD(x,y) = 0. That is, the semantics of x and y in the Google sense is the same.

   (b) If frequency f (x) = 0, then for every search term y we have f(x,y) = 0, and the NGD(x,y) = log f (y)/ log (M/ f (y)).

ii) The NGD is always nonnegative and NGD(x,x) = 0 for every x. For every given pair x,y NGD(x,y) = NGD(y,x): it is symmetric.

iii) The NGD is scale-invariant. It is very important that, if the number M of pages indexed by Google grows sufficiently large, the number of pages containing given search terms goes to a fixed fraction of M, and so does the number of pages containing conjunctions of search terms. This means that if M doubles, then so do the f -frequencies. For the NGD to give us an objective semantic relation between search terms, it needs to become stable when the number M of indexed pages grows.

It is known world over that the Google search engine indexes more then ten billion pages on the Web today. Each such page can be viewed as a set of index terms. Consider an

example: If a Google search for 'horse', returned 46,700,000 hits and the number of hits for the search term 'rider' was 12,200,000. Searching for the pages where both 'horse' and 'rider' occur gave 2,630,000 hits, and Google indexed 8,058,044,651 Web pages. Using these numbers in the main formula with N = 8;058;044;651, yields a Normalized Google Distance between the terms 'horse' and 'rider' as follows: NGD(horse; rider) = (0.443)

The NGD is a normed semantic distance between the terms in question, usually (but not always, see below) in between 0 (identical) and 1 (unrelated), in the cognitive space invoked by the usage of the terms on the World-Wide-Web as filtered by Google. Because of the vastness and diversity of the Web this may be taken as related to the current use of the terms in society. Same calculation was done by Cilibrasi and Vitanyi when Google indexed only one-half of the current number of pages: 4,285,199,774. The probabilities of the used search terms didn't change significantly over this doubling of pages, with number of hits for 'horse' equal 23,700,000, for 'rider' equal 6,270,000, and for 'horse, rider' equal to 1,180,000. The NGD (horse; rider) computed in that situation was (0.460). This is in line with contention that the relative frequencies of Web pages containing search terms gives objective information about the semantic relations between the search terms. If this is the case, then the Google probabilities of search terms and the computed NGD 's should stabilize (become scale invariant) with a growing Google database [187].

Calculating Normalized Similarity Score (NSS)

NSS is an MSR that is derived from NGD. The relatedness between two words x and y is derived as:

$$NSS\,(x,y) = 1 - NGD\,(x,y) \tag{4.2}$$

where NGD is a formula derived by Cilibrasi and Vitanyi [187].

Algorithm 4 shows how term-category matrix is generated to find semantic relatedness between each term in the annotation tag and each candidate category considered in *ADWebS*.

```
Input: N:Total no. of terms in a service; LC: Total no. of categories in
       database
Output: NSS(k)
1 N=Tr;
2 foreach Category (C[i]) in LC do
3 |   j=0;
4 |   foreach term (Tr[j]) in j do
5 |   |   V[i][j] = CALCULATE(C[i],Tr[j]);
6 |   |   j=j+1;
7 |   end
8 |   i = i+1;
9 end
```

Algorithm 4: Algorithm to find Normalized Similarity Score (NSS) between terms in a service and candidate categories

Here $\mathrm{T} = T_1, T_2, T_3, \ldots T_n$ denote the set of all terms in information domain I where T_1 represents terms retrieved from first WSDL file, T_2 is the set of terms extracted from second file and so on. The semantic correlation $V_{[i][j]}$ between each term (T_i) and Candidate Categories (C_i) is calculated.

Time complexity of Algorithm 4 (Table 4.1)

So

$$\mathrm{T(n)}=c1*1 + c2 * L + c3 * (L-1) + c4*\sum_{j=1}^{L} n_j + c5*\sum_{j=1}^{L}(n_j - 1) + c6*\sum_{j=1}^{L}(n_j - 1) + c7*(L-1)$$

Best Case:

If total no. of terms in a service i.e. n_j is single i.e.

n_j=1

Then

Table 4.1: Time complexity of Algorithm 4

Sr.No.	Input	Cost	Time
1.	N=Tr	C1	1
2.	ForEachCategory ($C[i]$) in LC	C2	L
3.	j=0	C3	L-1
4.	ForEachterm (Tr[j]) *in* j	C4	$\sum_{j=1}^{L} n_j$
5.	$V_{[i][j]}$=CALCULATE($C[i]$,$Tr[j]$);	C5	$\sum_{j=1}^{L}(n_j - 1)$
6.	j=j+1	C6	$\sum_{j=1}^{L}(n_j - 1)$
7.	End	0	0
8.	i = i+1	C7	L-1
9.	End	0	0

T(n)= c1+c2$*L + c3 * (L-1) + c4 * L + c7 * (L-1)$

= (c2+c3+c4+c7)$*L + (c1 - c3 - c7)$

So **T(n)=Θ(L)**

(Where L are total no. of categories in database)

Worst Case: If total no. of terms in a service i.e. n_j is equal to total no. of categories i.e.

n_j=L

Then

T(n)=c1+c2$*L + c3 * (L-1) + c4 * L^2 + c5 * L^2 + c6 * L^2 + c7 * (L-1)$

= (c4+c5+c6)$*L^2 + (c2 + c3 + c7) * L + (c1 - c3 - c7)$

So T(n)=$\Theta(L^2)$

Calculating the NSS of all terms in a annotated Web service S_i with each candidate category means that semantic relatedness of each term with each category has been found and a term category matrix is generated. Table 3.1 shows the values achieved with five candidate categories using algorithm 4.

Table 4.2: NSS of each word with each category

Category / Words	Zip Code	Weather	Country	Stock Market	Currency
City	0.639	0.822	0.983	0.473	0.244
Temperature	0.826	0.635	0.948	0.241	0.11
Pressure	0.664	0.98	0.862	0.668	0.226
Humidity	0.924	0.992	0.891	0.533	0.547
Precipitation	0.368	0.89	0.503	0.028	0.013
Visibility	0.15	0.983	0.476	.059	0.032
Clouds	0.078	0.517	0.062	0.058	0.02
Wind speed	0.444	0.905	0.243	0.021	0.009
Wind direction	0.017	0.878	0.259	0.014	0.007
Longitude	0.861	0.988	0.934	0.217	0.54
Latitude	0.587	0.959	0.887	0.26	0.273
Elevation	0.472	0.594	0.702	0.062	0.024

Next step in *ADWebS* is to assign the service into one or more category and ranking of the services as per the user's query.

4.2 Categorization of Web services in *ADWebS*

Categorization of Web services is the next important step in the proposed framework and the cornerstone behind category determination approach in *ADWebS* is the fact that there exists a semantic associations between the category of a Web service and its semantic meta-data description. In [135] Crasso *et al.* reported several experiments for assessing the degree of dependency between the category and the metadata and their experiments clearly indicated that the proper usage of terms always increases the probability of assignment of right category to the service in question.

Once the term category matrix with semantic values is found, addition of a Web service to a respective category(s) in *ADWebS* is considered at two different levels:-

i) If the size of term category matrix is small then Non parametric test Kruskal Wallis is applied to add a service into one or more categories.

ii) If the size of term category matrix is large then dimension reduction technique Principal Component Analysis (PCA) is applied before adding a service into one or more categories.

4.2.1 Non-parametric tests

Non-parametric statistic refer to a statistic (a function on a sample) whose interpretation does not depend on the population fitting any parameterized distributions. Non-parametric statistical tests are distribution-independent tests that are used to analyze data for which an underlying distribution (such as the normal distribution) is not assumed.

Non-parametric methods are widely used for studying populations that take on a ranked order. The use of these methods may be necessary when data have a ranking but no clear numerical interpretation, such as when assessing preferences; in terms of levels of measurement, for data on an ordinal scale.

Non-parametric statistics have a number of advantages over parametric statistics. They can be quick and easy to use as they often use ranks or signs of differences rather than the values themselves. They can reduce the work of data collection because data can be ranks or simple scores rather than precise measurements. Sampling procedures do not assume homogeneity of variances, for example between locations or over time.

As non-parametric methods make fewer assumptions, their applicability is much wider than the corresponding parametric methods. In particular, they may be applied in situations where less is known about the application in question. Also, due to the reliance on fewer assumptions, non-parametric methods are more robust and above all they are simple methods. Due to the simplicity of use and to greater robustness, non-parametric methods are seen by some statisticians as more efficient as they leave

less room for improper use and misunderstanding.

Choosing Between Parametric and Nonparametric Tests

Choosing between parametric and nonparametric tests is sometimes easy. One should definitely choose a parametric test if one is sure that your data are sampled from a population that follows a Gaussian distribution (at least approximately). One should definitely select a nonparametric test in three situations:

i) The outcome is a rank or a score and the population is clearly not Gaussian. Examples include class ranking of students, the visual analogue score for pain (measured on a continuous scale where 0 is no pain and 10 is unbearable pain), *etc.*

ii) Some values are 'off the scale', that is, too high or too low to measure. Even if the population is Gaussian, it is impossible to analyze such data with a parametric test since you don't know all of the values. Using a nonparametric test with these data is simple.

iii) The data requires measurements and population is not distributed in a Gaussian manner.

Kruskal Wallis Test

The Kruskal Wallis test is one of the non parametric tests used as a generalized form of the Mann Whitney U test. It is used to test the null hypothesis which states that 'k' number of samples has been drawn from the same population or the identical population with the same or identical median. If S_j is the population median for the j_{th} group or sample in the Kruskal Wallis test, then the null hypothesis in mathematical form can

be written as $S1 = S2 = .. = S_k$. In the computation of the Kruskal Wallis test, each of the 'N' observations is replaced in the form of ranks *i.e.* all the values from the 'k' number of samples are combined together and are ranked in a single series.

The smallest in the Kruskal Wallis test is replaced by the rank 1. The next smallest in the Kruskal Wallis test is replaced by rank 2, and the largest in the Kruskal Wallis test is replaced by 'N', the total number of the observations in the 'k' number of samples. After this, the sum of ranks in each sample or column is found in the Kruskal Wallis test. From the sum of the ranks, the researcher in the Kruskal Wallis test computes the average rank for each sample or group. If the samples are from an identical population in the Kruskal Wallis test, then the average rank should be about the same. On the other hand, if the samples in the Kruskal Wallis test are from populations with different medians, then the average rank will differ.

There are certain assumptions in the Kruskal Wallis test. It is assumed that:-

i) The observations in the data set are independent of each other.

ii) The distribution of the population should not be necessarily normal and the variances should not be necessarily equal.

iii) The observations must be drawn from the population by the process of random sampling.

The sample sizes in the Kruskal Wallis test should be as equal as possible, but some differences are allowed. Procedure for Kruskal Wallis Test:

i) Arrange the data of all samples in a single series in ascending order.

ii) Assign rank to them in ascending order. In the case of a repeated value, assign ranks to them by averaging their rank position.

iii) Once this is complete, ranks or the different samples are separated and summed up as $R_1, R_2, R_3,$ *etc.*

iv) To calculate the value of Kruskal Wallis Test, apply the following formula:

$$K = \frac{12}{(n * (n+1))} \left[\Sigma \frac{R_j^2}{n_j} \right] - 3(n+1) \tag{4.3}$$

Where, n_j is the number of observations in the j_{th} sample, n is the total number of observations, and R_j is the sum of ranks for the j_{th} sample.

4.2.2 Dimension Reduction for large matrix

If the size of term-category matrix is very large the dimension reduction is done by following steps

i) Find the correlation matrix of the term-category matrix.

ii) Calculate the eigen value and eigen vector of the correlation matrix.

iii) Apply Principal Component Analysis (PCA) on the values generated in Step 2.

Correlation Matrix

A Correlation matrix describes correlation among M variables. It is a square symmetrical M × M matrix with the $(ij)_{th}$ element equal to the correlation coefficient r_{ij} between the $(i)_{th}$ and the $(j)_{th}$ variable. The diagonal elements (correlations of variables with themselves) are always equal to 1.00.

Normally a correlation between two variables is analyzed but in most studies there are more than two variables. Correlation for any pair of variables can be found by finding the value of intersecting row and column for the two variables. Let's consider there are 'n' categories ranging from C_1 to C_n where n is very large and 'm' terms

in each WSDL file then the term category matrix formed in *ADWebS* after finding semantic relationships among all terms and categories will definitely be big enough.

Eigen Values

Definition: Let A be a n × n matrix. The number λ is an eigenvalue of A if there exists a non-zero vector v such that Av=λ v. Any such vector, v is called an eigenvector of the matrix A, associated with the eigenvalue λ . An eigenvalue of a square matrix is a scalar that is usually represented by the Greek letter λ and eigenvector is a non-zero vector denoted by the small letter 'x'.

Definition: The dominant eigenvector of a matrix is an eigenvector corresponding to the eigenvalue of largest magnitude (for real numbers, largest absolute value) of that matrix.

The eigenvalues, or 'latent roots', or 'characteristic roots', of a correlation matrix are sometimes used as a means of estimating the number of factors (or components) which may underpin a test, or a scale. Each eigenvalue corresponds to a 'principal component'. The first principal component corresponds to the ellipsoid's major axis, to its longest axis. Each eigenvalue represents the relative length of one of the ellipsoid's axes. Each of these axes is said to represent, or correspond to, a principal component.

Once the correlation matrix is formed, the eigenvalues and eigenvectors are calculated and only the first p eigenvectors are chosen, based on their eigenvalues. Eigenvectors give the components in order of significance hence the eigenvector with the highest eigenvalue is the principle component of the data set.

Applying Principal Component Analysis(PCA)

Principal Component Analysis (PCA) is a linear transformation that transforms the data to a new coordinate system such that the new set of variables *i.e.*, the principal components, are linear functions of the original variables. It is a statistics technique for

simplifying a data set. In practice, this is achieved by computing the correlation matrix for the full data set. It is a form of unsupervised learning which relies entirely on the input data itself without reference to the corresponding target data (the criterion to be maximized is the variance). Eigenvectors are calculated to define principal component weights, and eigenvalues represent variances of principal components.

The principal component weights are estimated in such a way that:

i) The first principal component, ξ_1, accounts for the maximum variance in the data, the second principal component, ξ_2, accounts for the maximum variance that has not been accounted for by the first principal component, and so on.

ii) For each principal component, the sum of squares of its weights should be equal to 1.

Only first few components are considered so the size of term-category matrix is reduced considerably.

4.2.3 Soft Categorization vs. Hard Categorization

Web Service will be placed in a particular category C_i which has maximum median value after applying Kruskal Wallis test. At this point two alternates are: a service can belong to only one category (Hard categorization) or it can belong to more than one categories (Soft categorization). Soft categorization is compulsory because it is always possible that give a set of functional annotations of a Web service, some terms may be more semantically related to a particular category but they may be semantically related to other categories too. So, it would not be optimal to totally ignore all the other categories. Criteria used for soft categorization is that all categories whose median lie in the 10% limit of category having maximum median after applying Kruskal Wallis

test will also include that service in their category. For e.g., if corresponding to a service S_i if category C_i has median value 0.8 and category C_{i+1} has median value 0.74 then service S_i will fall in both the categories *i.e.* C_i and C_{i+1}, while if one category C_i has value of 0.8 and all categories have value less than 0.7 then service go to only one category C_i.

Once a service has been categorized into one or more categories all the terms in the annotation tag of the service are added in the dropdown boxes of the category.

4.3 Ranking of Services

The frontend of *ADWebS* provides service discoverer or the end user with a drop down of candidate categories and once user selects a category, user is provided with two drop downs Q_1 and Q_2 having terms related to the selected category along with a text box Q_3 for adding query terms.

After selecting a category the possible query options available to the user are:

i) User provides input only in Q_1 and nothing in Q_2 and Q_3 (e.g. term 'temperature' in category 'Weather')

ii) User provides input in both Q_1 and Q_2 and nothing in Q_3 (e.g. term 'temperature' and 'humidity' in category 'Weather')

iii) User provides input all Q_1, Q_2 and Q_3 (e.g. term 'temperature' and 'humidity' along with 'city', 'New York' in category 'Weather')

iv) User provides input only in Q_3 and nothing in Q_1 and Q_2 (e.g. only sentence 'city is New York' in category 'Weather')

4.3.1 Ranking Criteria

Ranking criteria for all the aforesaid options are:

i) URLs having all Q_1, Q_2 and Q_3 terms in their semantic annotations will be ranked first (*i.e.* case (iii), e.g. Web services having all three terms 'temperature', 'humidity' and 'city' in category 'Weather', provided they are available)

ii) URLs having both Q_1 and Q_2 in their semantic annotations will be ranked next (*i.e.* case (ii), e.g. Web services having both terms 'temperature', 'humidity' in category 'Weather', provided they are available)

iii) Next URLs having Q_1 in their semantic annotations will be ranked (*i.e.* case (i), e.g. Web services having terms 'temperature', in category 'Weather', provided they are available)

iv) URLs having words extracted from Q_3 in their semantic annotations will be ranked if only Q_3 is provided (*i.e.* case (iv), e.g. Web services having terms 'city' and 'New York', in category 'Weather', provided they are available)

The URLs of all the Web services are displayed in the pattern mentioned above and the remaining Web services of that category will be displayed giving user the entire set of semantically related Web services in category selected by the user. Next section discusses various views of the framework which provides an elaborated picture of the processes followed in the *ADWebS*.

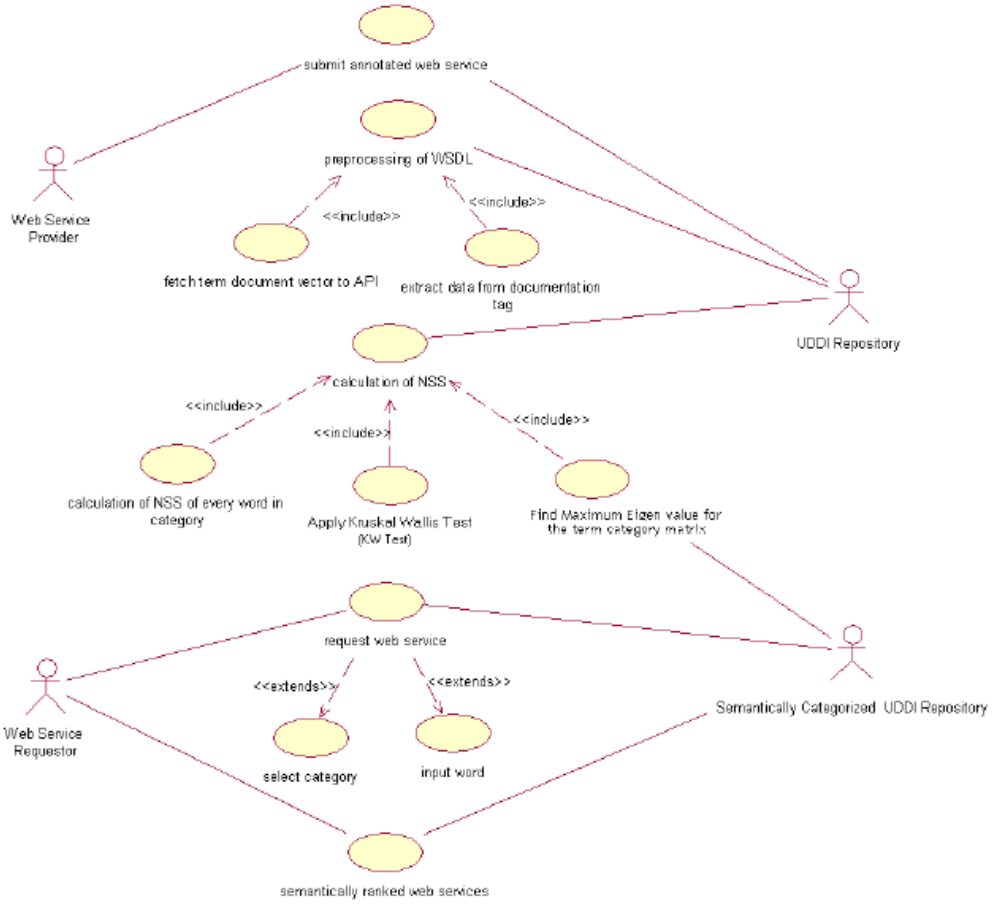

Figure 4.1: Use case diagram of '*ADWebS*'

4.4 Different Views of the Framework

4.4.1 Use case Diagram of *ADWebS*

Figure 4.1 gives the Use case diagram of *AdwebS*

4.4.2 Sequence Diagram of *ADWebS*

Figure 4.2 gives the sequence diagram of *AdwebS*

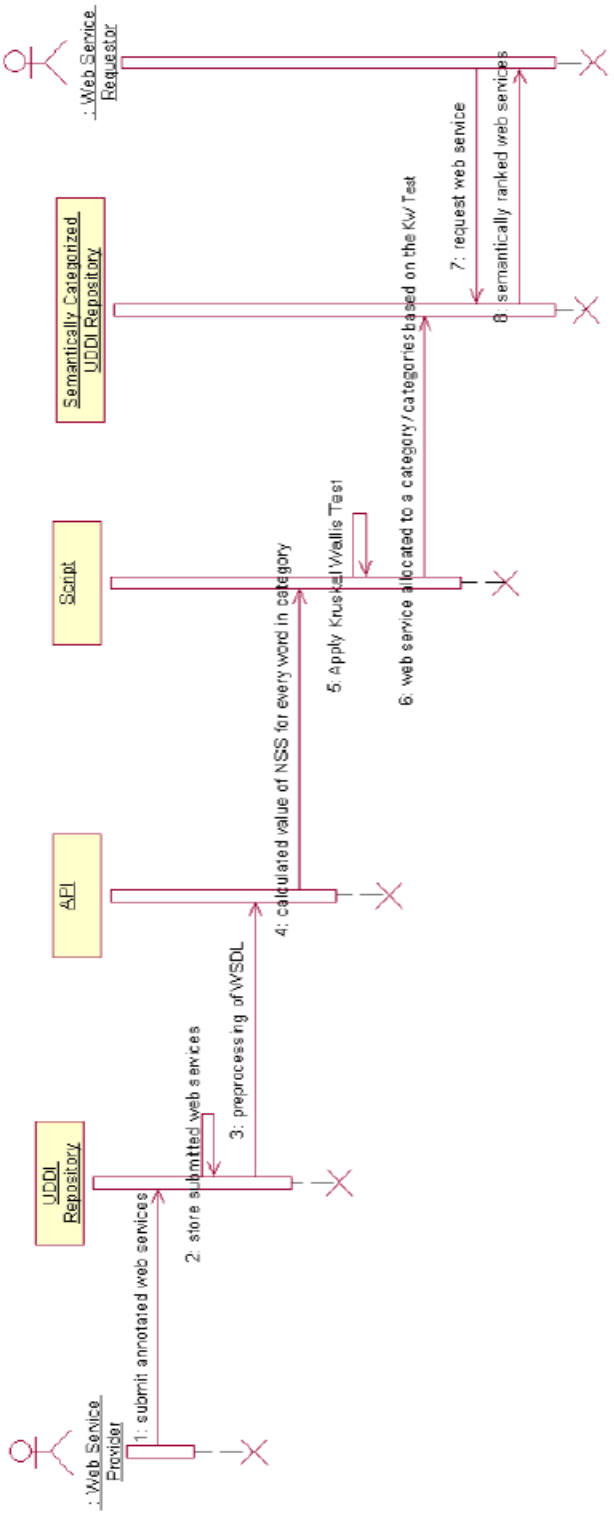

Figure 4.2: Sequence Diagram of '*ADWebS*'

4.4.3 Advantages of *ADWebS* over approaches used in prevalent Frameworks for Services Discovery

Two major benefits are:

i) No use of domain ontologies:

Usage of domain ontologies as knowledge corpus has some inherited problems:

Various proposals include initiatives, projects and languages such as WSMO, METEOR-S, OWL-S and SWSA/SWSL which have come up for adding semantics in Web services prefer usage of ontology as a base for mapping and matching different type of services. Although standard vocabularies like ontologies provide semantic descriptions they don't adapt well for heterogeneous domains like Web services as a very broad coverage vocabulary is required for such domains. Gruber [238] has clearly stated that 'It is impossible to develop a single ontology that fully covers a domain or will satisfy the needs and preferences of each user'.

Secondly, design of ontology is dependent on the representation language's expressiveness and if the language is weak then the designed ontology will also be weak.

Finally, since there is no standard methodology pre-defined for development of ontologies, many times it is difficult to understand the intended meaning of concepts and the associations existing between the interrelated concepts of the ontology.

ii) Use of World Wide Web as Knowledge Corpora:

Web services carry out operations to support real-world services, *e.g.*, the ordering of goods. Thus, Web services exist on the boundary of the world inside an information system and the external world. Lexical resources of knowledge like

Wordnet, Thesaurus, *etc.* cannot efficiently find semantic relatedness between concepts coming from millions of Web users and diverse text corpora. Web search engines can serve as an efficient interface to extract semantics since its sheer mass of users and documents with different intentions averages out to give the true semantic meaning used by people world over. Since there is no doubt that the relative page counts approximate the true societal word- and phrases usage and Google is an able extractor, Normalized Google Distance is used as a measure of semantic relatedness between terms and categories.

4.5 Comparative analysis of keyword based approaches *vs. ADWebS*

Table 4.3: Comparative analysis of Keyword approach and *ADWebS* for Web service discovery

Sr. No.	**Features**	**Keyword based Approaches**	***ADWebS* Approach**
1.	Approach Used	Syntactic Approach (Keyword based)	Semantic Approach (Semantic annotations based)
2.	Provision for additional semantics in the current service description file	No	Yes, Additional semantics included WSDL file of service

3.	Provision for integration of different domains and concepts	No	Yes, marginally covered as publisher is free to add any term which increases the relevance or clarity of the service at discovery stage
4.	Services Categorization	Yes, but semantic categorization not available	Yes, services classified into pre-defined categorizes semantically
5.	User interface for querying services	User has to identify the services and categories manually	User provided with front end having list of candidate categories
6.	Scope for defining requester's perspectives in the Services	Requester has to identify the category manually. Initial model contained categorization using USPSC an NASIC but rarely used due to difficulty of identifying it	Services have been categorized semantically and requestor is provided with list of candidate categories
7.	Scope for extracting functional capabilities of the services at discovery time	Marginally covered in some approaches by extracted terms within the WSDL or comments available in the service	Annotation tag includes all the terms describing the functional capabilities of the service
8.	Method of determining semantic association	None	Semantic of service determined using semantic relatedness measures

4.6 Conclusions

ADWebS proposes addition of semantic annotation as semantically annotating WSDL elements dissolve the ambiguities in services structures and syntax, and this aid in services discovery. Using semantic relatedness measures like NGD help in finding the semantic similarity between the annotations and candidate categories paving the way for semantic categorization of Web services.

Services categorization can greatly enhance the discovery process and reduce the discoverers burden of finding the relevant services manually. Using statistical measures like Kruskal Wallis, *ADWebS* categorizes a Web service into one of the candidate category(s). Further, soft categorization has been considered in this work *i.e.* a Web service can be fall in more than one category. Once the service is added into one or more categories end user or the services discoverer is directed to the semantically categorized UDDI repository.

ADWebS has been evaluated with a realtime and operative set of Web services, the results achieved using different data-sets has been examined and discussed in the next chapter.

Chapter 5

Implementation and Testing of *ADWebS*

The proposed framework '*ADWebS*' provides a novel approach for semantic discovery of Web service descriptions by addition of annotation tag in the current service description file and based on these annotations where semantic relatedness is calculated and search space is reduced drastically. The framework has been designed to have a 'Google' like interface which allows service discoverer to opt for a service from a list of candidate categories along with a text box for user's query. The objective of the designed framework is to relieve end user from manually tracing service category and services available within a category.

5.1 Implementing Web Services

Steps followed in implementing a Web service are:-

- A service provider creates a Web service

- The service provider uses service description language to describe the service to a registry
- The service provider registers the service in a UDDI registry and/or ebXML registry/repository.
- Another service or consumer locates and requests the registered service by querying UDDI and/or ebXML registries.
- The requesting service or user writes an application to bind the registered service using SOAP in the case of UDDI and/or ebXML
- Data and messages are exchanged as XML over HTTP

5.2 Tools used for implementing *ADWebS*

WAMP server

A WAMP Server is a Windows Machine that has Apache, MySQL, and PHP on it (WAMP - Windows, Apache, MySQL, PHP) Earlier, to install these softwares, one would have to get the installs and binaries and configure them yourself and set everything up which used to be a tedious task as well as very time consuming. Now, when used in combination they represent a solution stack of technologies that support application servers. Other such stacks include unified application development environments such as Apple Computer's WebObjects, Java/Java EE, Grails, and Microsoft's .NET architecture.

WAMP Server is a combination of following:

PHP: Hypertext Preprocessor is a scripting language that is embedded in HTML. PHP scripting code is frequently used to connect web pages to MySQL databases to

create dynamic Web sites. Some popular examples of PHP driven Web sites are blogs, message boards, and Wikis. PHP files are actually plain text files; they can be created in TextPad or any other Plain Text Editor.

MySQL: The MySQL database server is the world's most popular open source database. Its architecture makes it extremely fast and easy to customize. Extensive reuse of code within the software and a minimalistic approach to producing functionally-rich features has resulted in a database management system unmatched in speed, compactness, stability and ease of deployment. The unique separation of the core server from the table handler makes it possible to run with strict transaction control or with ultra-fast transactionless disk access, whichever is most appropriate for the situation.

phpMyAdmin is a popular PHP driven Web interface that allows users to manage your MySQL database content via a Web form. Currently it can create and drop databases, create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields. phpMyAdmin is "open source" and therefore free.

Apache is an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this server is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. It is a popular Web server that many ISP's and individuals use to host Wweb pages. When Apache is installed on your system your machine becomes a web server. Pages stored on your system in a "special folder" are accessible on the Internet via the machine's IP address. In order for pages to be viewed on the Internet, the files must be stored in a special directory; this directory is usually called htdocs, $public_html$, or www.

Windows Services are used to run these servers and modules. Once services are

installed they can be started and stopped via Start > Control Panel > Administrative Tools (switch to classic view if you do not see the Administrative tools) > Services. These Services can be configured to automatically start or to be started manually.

5.3 Implementation details of 'ADWebS'

Initially the original service description files have been modified by adding a new tag 'annotations' comprising of set of 'n' terms or semantic metadata. The set of terms selected for addition are from the document tag and functional descriptions available in the WSDL file of a Web service. Datasets used for checking the performance of *ADWebS* was taken from the public UDDI registry XMethods [263] and consist of functionally operative Web services specifications in WSDL files initially classified in *five* categories: *Zip code, Country, Stock Market, Weather and Currency.* These domains were chosen because they have larger numbers of published Web services. A PHP script has been written to extract terms from the modified WSDL files *i.e.* annotated WSDL file.

Semantic relatedness between extracted terms and candidate categories has been found by calling the API within a PHP script. Once the semantic similarity values are available (Table 4.2), they are fetches to the free down loaded tool 'Statex' and Kruskal Wallis test is applied. Based on the median value achieved for each category, service is added into one or more candidate categories.

WAMP server is used for deployment and terms are extracted and stored in database file which is queried using SQL at the backend. Both publishing and discovering evaluations were deployed on an Intel Pentium working at 2.0 GHz with 3.0 GB of RAM. Both *ADWebS* and the data-set were hosted at the same computer, to avoid the overhead introduced by the network.

The designed framework '*ADWebS*' initiates the discovery process (Figure 5.1) by asking the service discoverer to select one of the candidate category from the list of candidate categories provided as a drop down list (Figure 5.2). Once the category is selected, the discoverer has option to either query the services through terms provided in the boxes Q_1 and Q_2 and/or use the text box provided in the User's Query option Q_3 (Figure 5.3). Once user inputs the query terms, the set of URLs ranked according to user's query are displayed (Figure 5.4). If query terms in the category change, the rank of the services i.e. the order of URLs change accordingly (Figure 5.5). Each step is depicted in the Figures shown below.

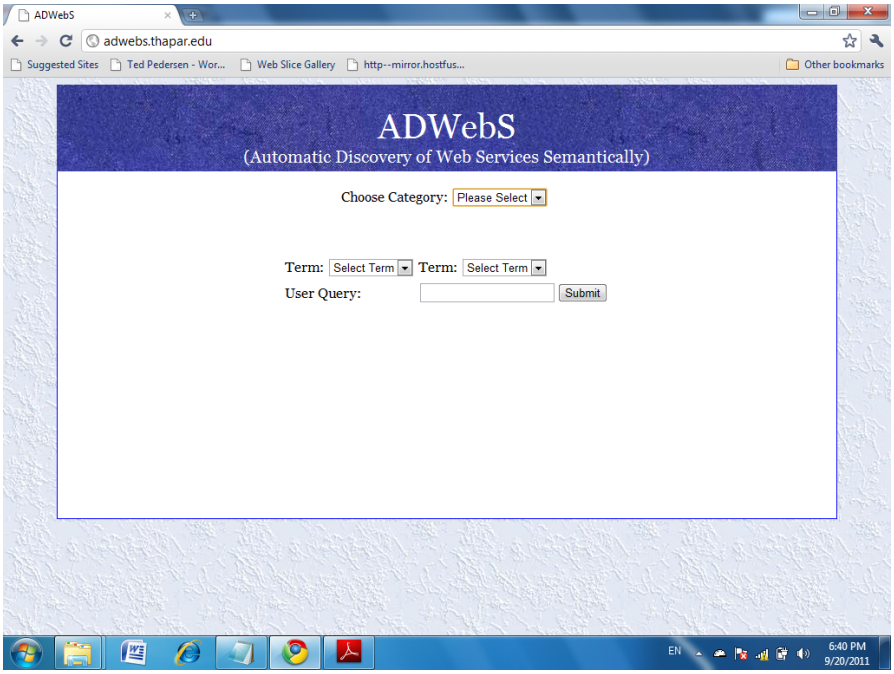

Figure 5.1: Front End of *ADWebS* provided to service discoverer

5.4 Evaluation Parameters

Framework has been evaluated on three parameters:

(i) Precision-at-n

Figure 5.2: List of categories provided to service discoverer

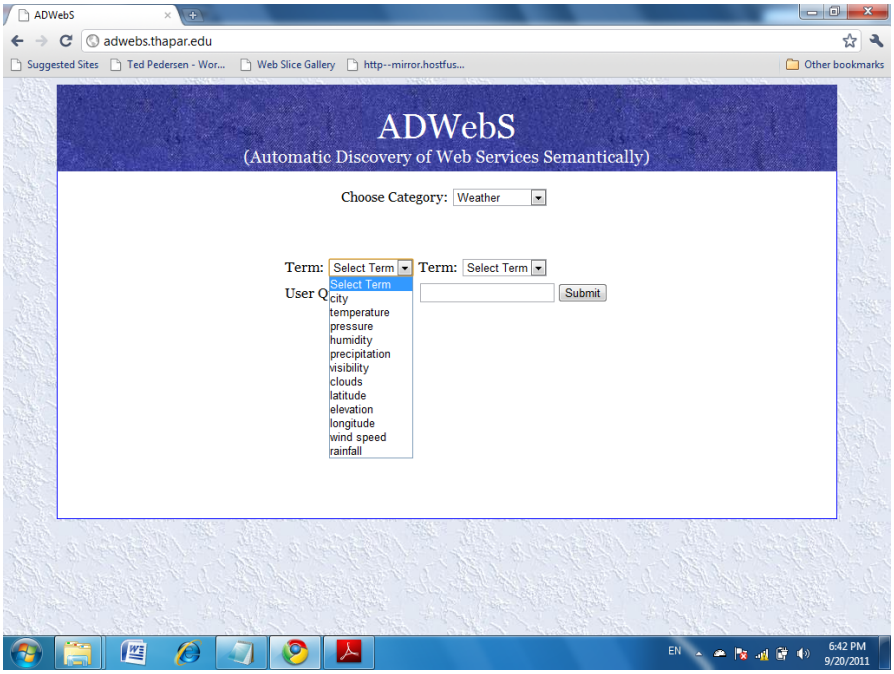

Figure 5.3: Selection of a category with terms in drop down

(i) Service retrieval time with and without categorization:

Minimum and Maximum time required for retrieving a service with categorization and without categorization are considered.

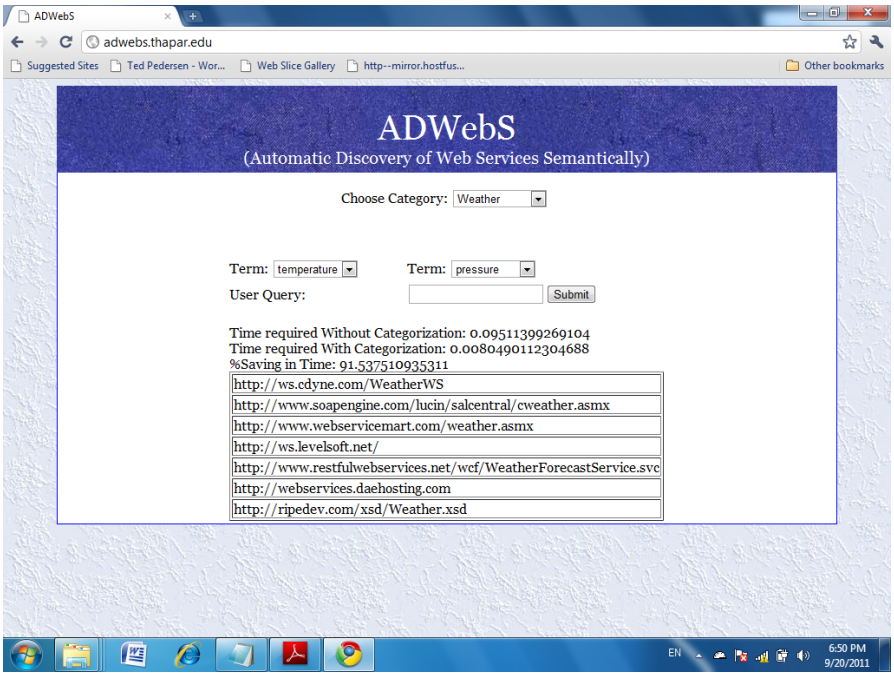

Figure 5.4: URLs displayed for selected category

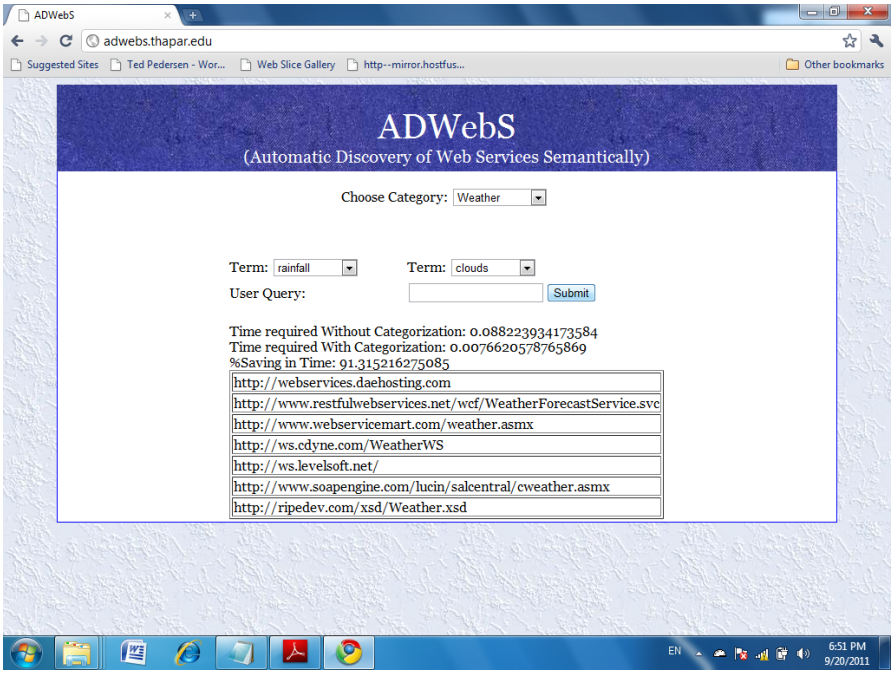

Figure 5.5: Change in the ranking with change in Query terms

(ii) Scalability of the framework:

Scalability of the framework is considered at two levels:

i) Vertical expansion of term category matrix:

Analyze the scalability of the framework by increasing the number of categorizes. In other terms, we try of find that what is the impact of increasing the number of categories in the proposed approach i.e. whether the framework will support efficient semantic categorization and discovery if the number of **categories** are increased many fold.

ii) Horizontal expansion of term category matrix:

Analyze the scalability of the framework by increasing the number of terms in a service. In other terms, we try of find that what is the impact of increasing the number of terms in a particular service i.e. whether the framework will support efficient semantic categorization and discovery if the number of **terms** in a service are increased many fold .

5.4.1 Precision-at-n measure

Precision-at-n measure allows computing precision at different cut-off points [197]. For example, if the top10 documents are all relevant to the query and the next 10 are all non-relevant, we have 100% precision at a cutoff of 10 documents but a 50% precision at a cut-off of 20 documents. Formally:

$$Precision - at - n = RetReln/n \tag{5.1}$$

where RetReln is the total number of relevant services retrieved in the top n.

We evaluated Precision-at-n for each experiment with three values of n: n = 2, n = 4 and n = 6. Figure 5.6 shows the averaged Precision-at-n for *ADWebS* discovery

approach. In *ADWebS*, average precision-at-2 was around 98 %.

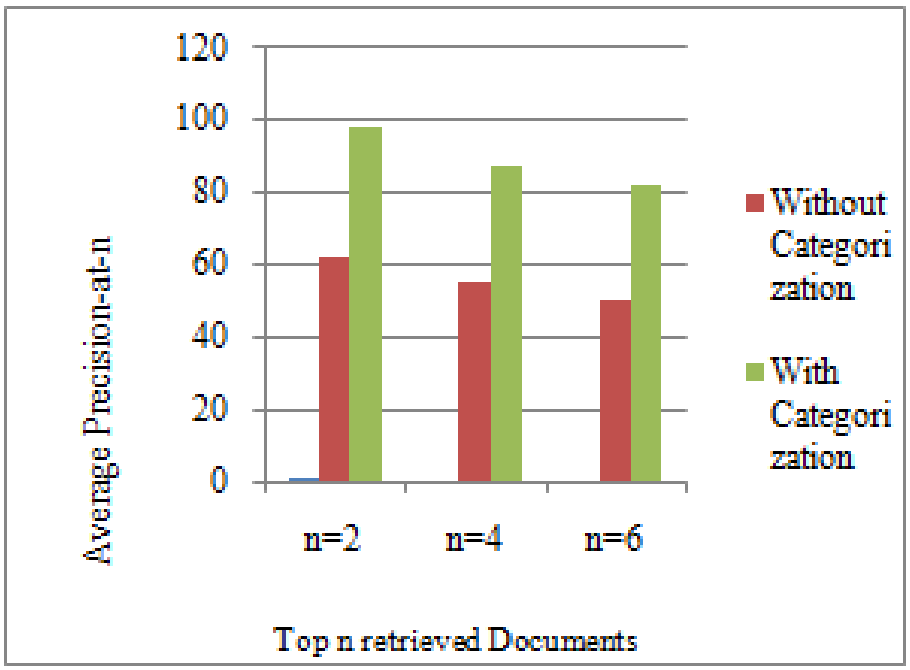

Figure 5.6: Average precision-at-n with and without categorization

5.4.2 Service retrieval time

To evaluate the performance of the proposed framework, comparison was performed between annotated Web service with categorization and services without categorization. A registry of services with and without annotations was prepared and when a user inputs the query terms, terms were searched in the entire Web service if there was no annotation tag and terms were searched only in annotation tag if the service had annotations.

Once the service category is selected, the search is limited to only finding correspondence between the query term and the set of services in that category. In consequence, a discoverer examines only those WSDL documents which are within the domain of selected category.

Service retrieval time is displayed when every user puts any query and time is shown for two cases:

i) With categorization and

ii) Without categorization

% of saving is also displayed in each case

retrieval time.png

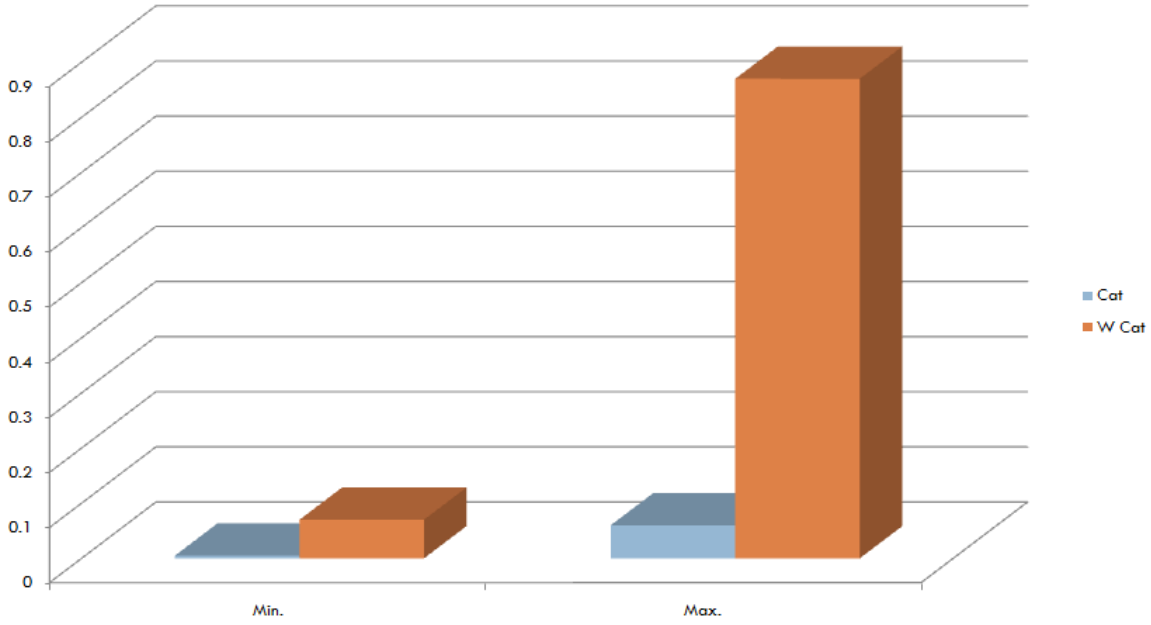

Figure 5.7: Service retrieval time with and without categorization

Search time for service with annotations was less as query terms were compared to the terms in the annotation tag only on the other hand services without annotations are searched till the end to check whether any term matches or not.

Results achieved in Figure 5.7 shows that minimum and maximum retrieval time for service with categorization is 0.06 milliseconds and 0.62 milliseconds while for service without categorization is 0.58 milliseconds and .87 milliseconds. On an average there is saving of 85% to 90% in service retrieval time.

Table 5.1: NSS of term category matrix

Category / Words	Zip Code	Weather	Country	Stock Market	Currency
Pressure	0.664	0.98	0.862	0.668	0.226
Temperature	0.826	0.635	0.948	0.241	0.11
Wind speed	0.444	0.905	0.243	0.021	0.009
Rainfall	0.155	0.302	0.	0.737	0.584
Humidity	0.924	0.992	0.891	0.533	0.547
City	0.639	0.822	0.983	0.473	0.244

5.4.3 Scalability

To check scalability by vertical expansion, the number of categories (initially five) were expanded gradually to check whether increasing the categories decreases the precision or increases the service retrieval time.

To check scalability by horizontal expansion the number of terms (initially six) were expanded gradually to check whether increasing the number of terms decreases the precision or increases the service retrieval time.

Table 5.1 was considered for testing the scalability of the proposed framework.

Vertical Expansion of Term Category Matrix

Figure 5.8 depicts NSS value of the initial set of five candidate categories *'Zip code', 'Country', 'stock market', 'Weather'* and *'Currency'* with six terms *'pressure', 'temperature', 'Wind speed', 'rainfall', 'humidity'* and *'city'*.

Figure 5.9 depicts the median value for each category after applying Kruskal Wallis test. Based on the values achieved in Figure 5.9 it is clear that service will come under category *Weather*.

Now the initial set of five candidate categorizes was expanded to seven by adding two more categories '*SMS'* and *'Corporate'* and NSS of all the terms was calculated with the new added categories. Values achieved and the service categorization is depicted in the Figure 5.10 and Figure 5.11 respectively.

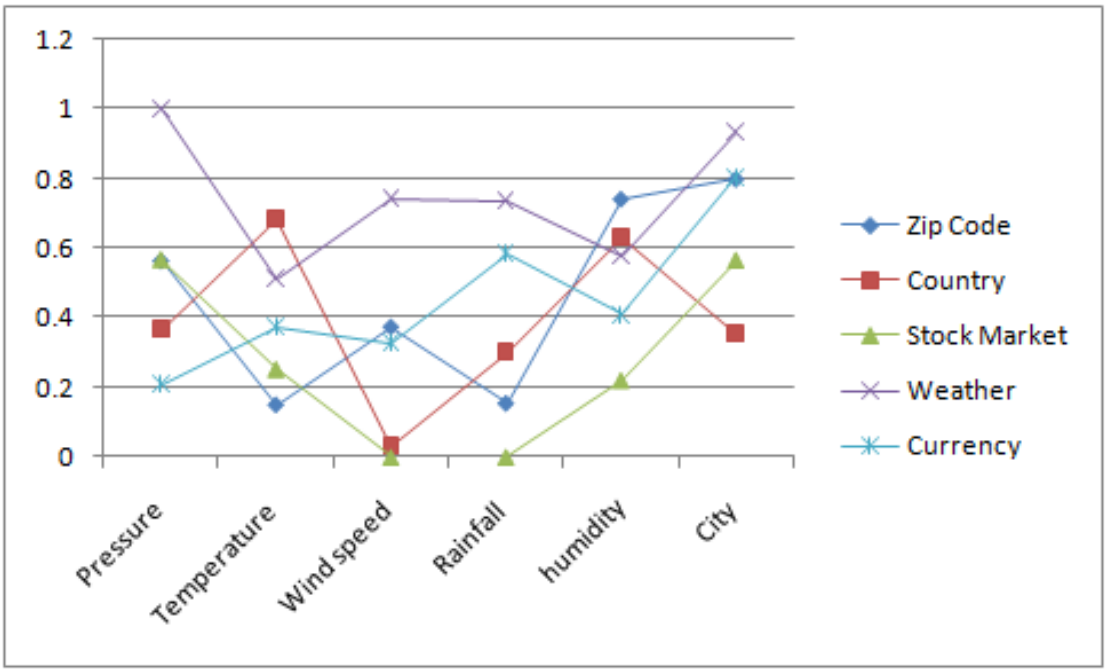

Figure 5.8: NSS values for *six* terms and *five* categories

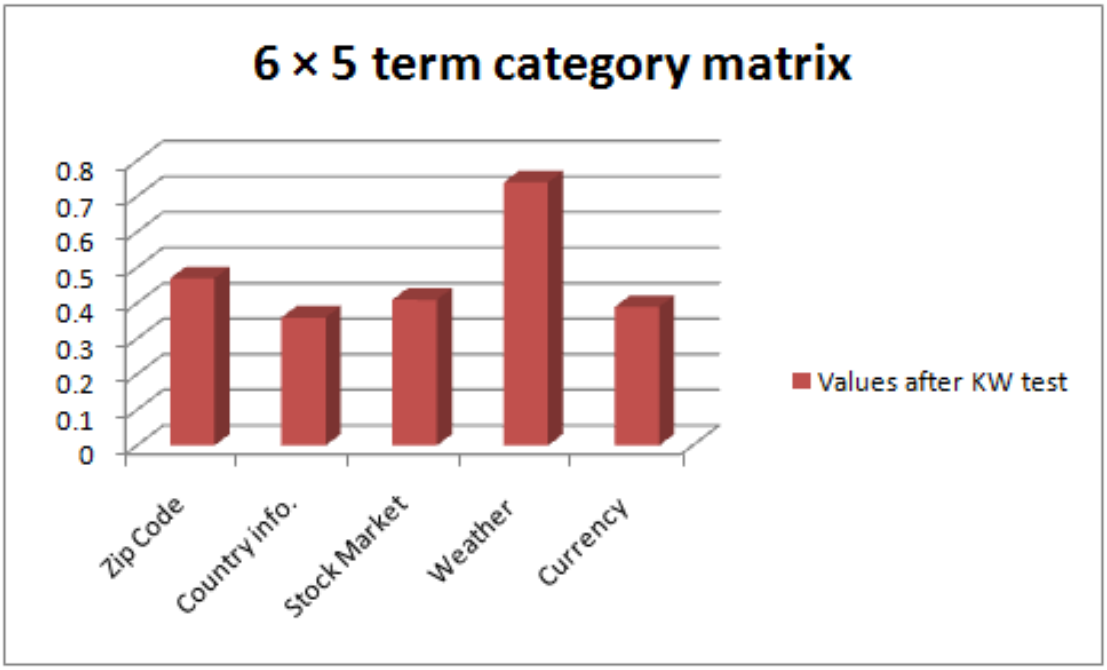

Figure 5.9: Median values for *six* terms and *five* categories after applying Kruskal Wallis test

Since the two newly added service do not have much close association to the category *Weather*, service will still come under category *Weather*. Addition of two new services

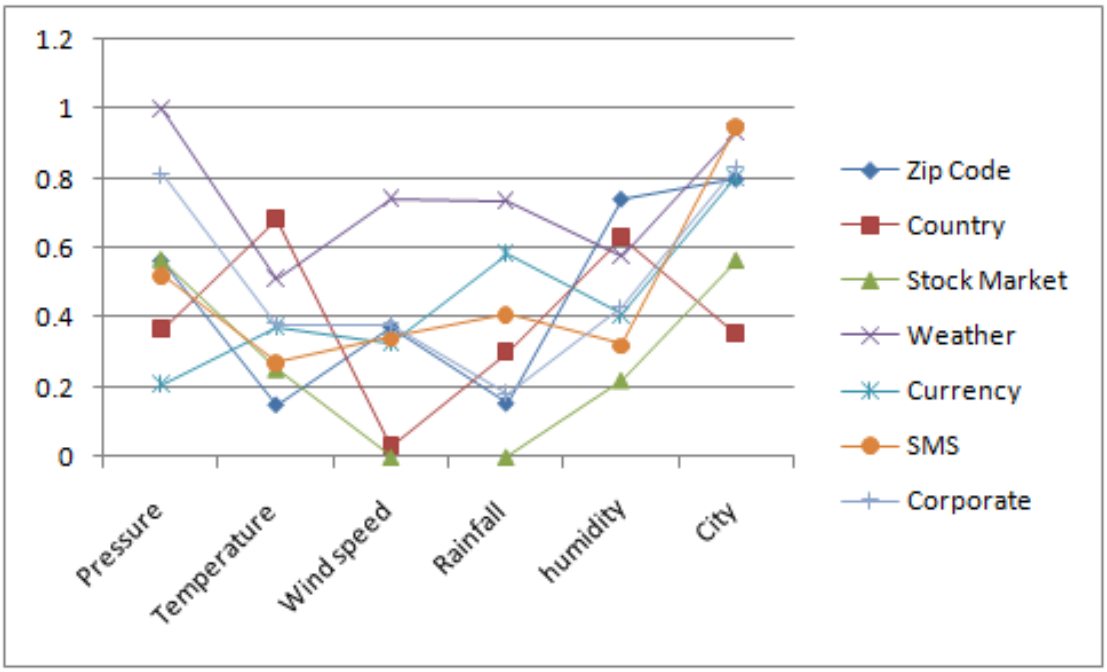

Figure 5.10: NSS values for *six* terms and *seven* categories

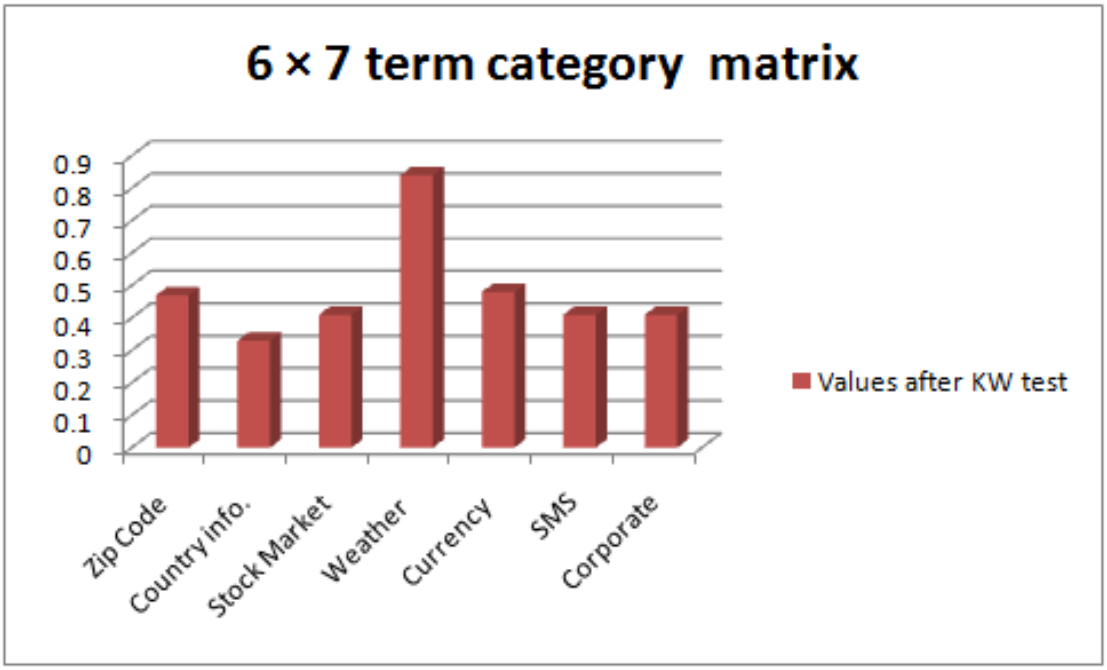

Figure 5.11: Median values for *six* terms and *seven* categories after applying Kruskal Wallis test

does not affects the performance of the framework but it increases the search scope for the service discoverer.

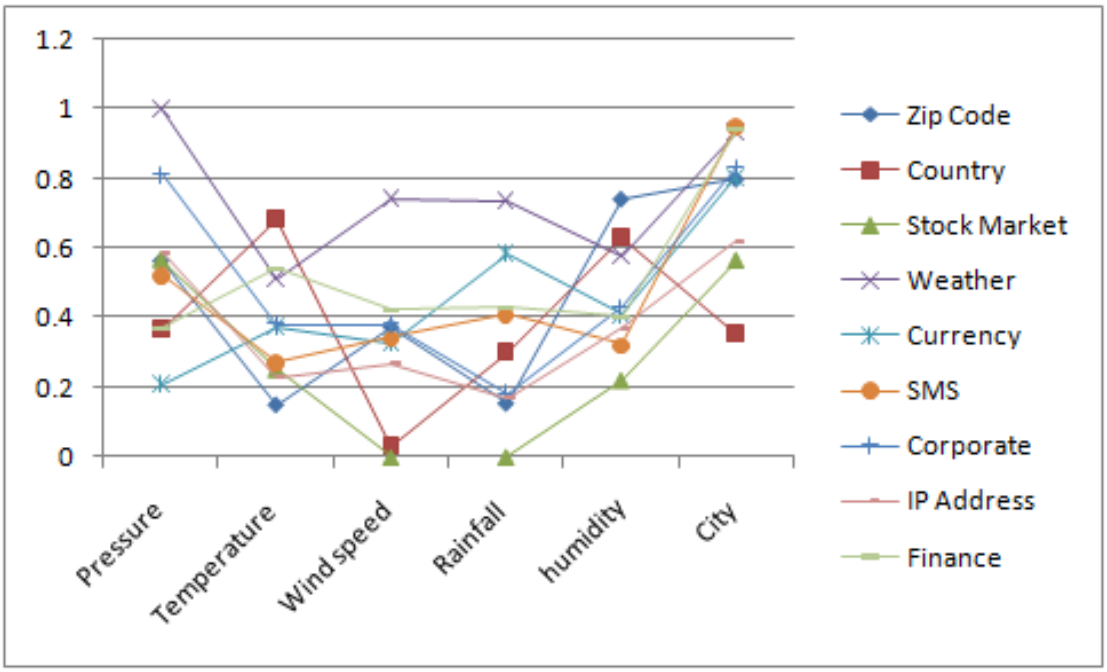

Figure 5.12: NSS values for *six* terms and *nine* categories

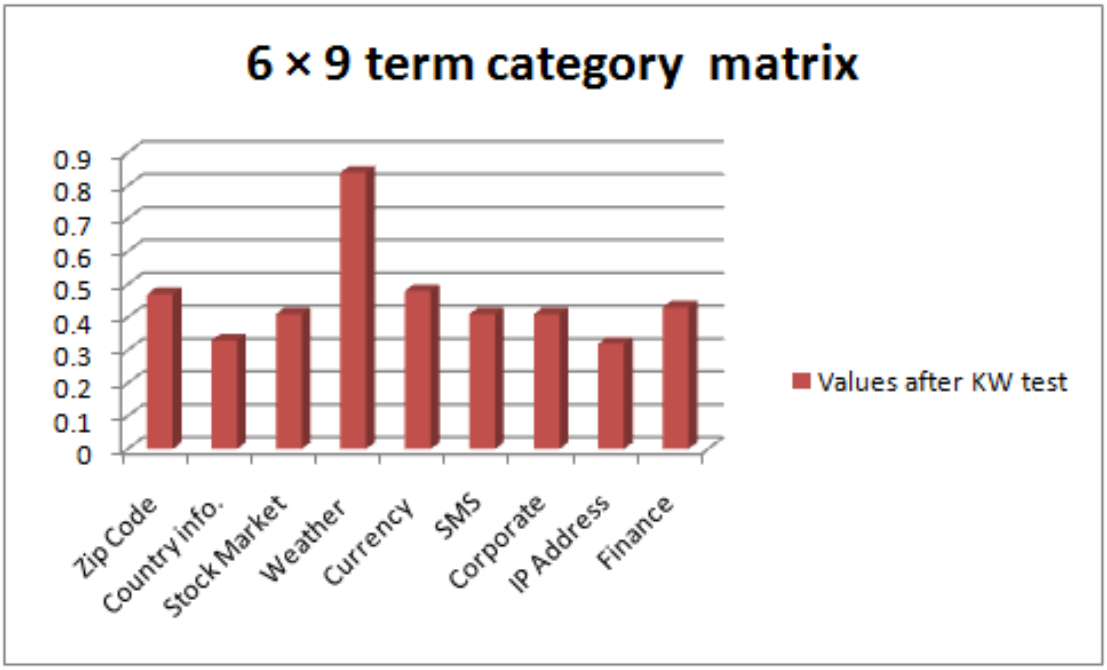

Figure 5.13: Median values for *six* terms and *nine* categories after applying Kruskal Wallis test

Now the set of seven candidate categorizes was expanded to nine by adding two more categories '*Finance*' and '*IP address*' and NSS of all the terms was calculated with

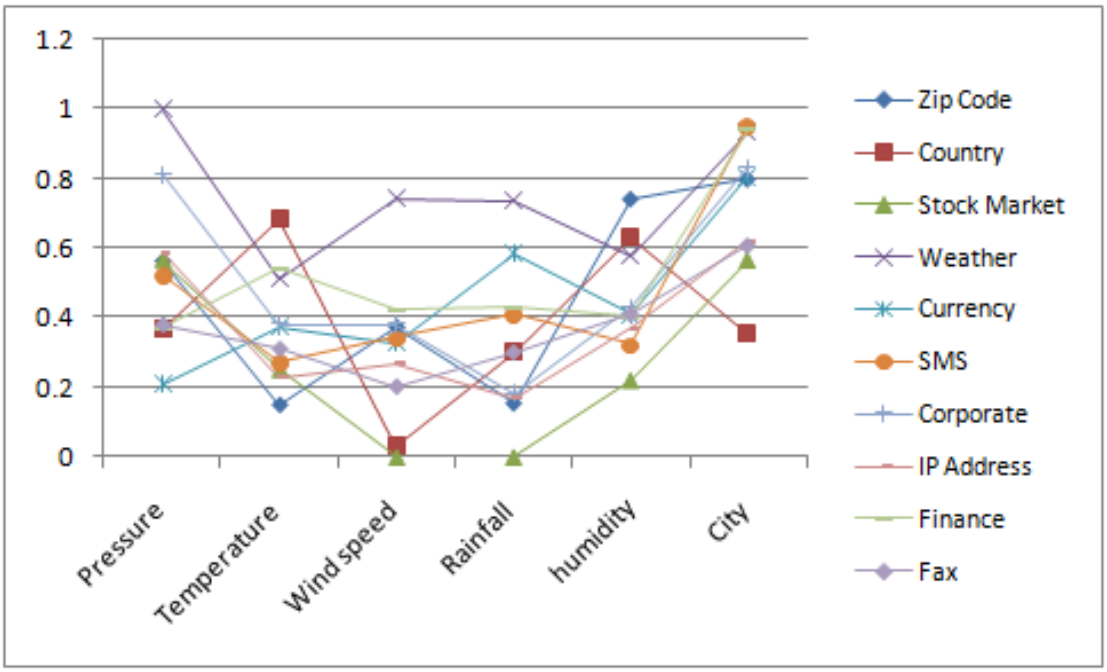

Figure 5.14: NSS values for *six* terms and *ten* categories

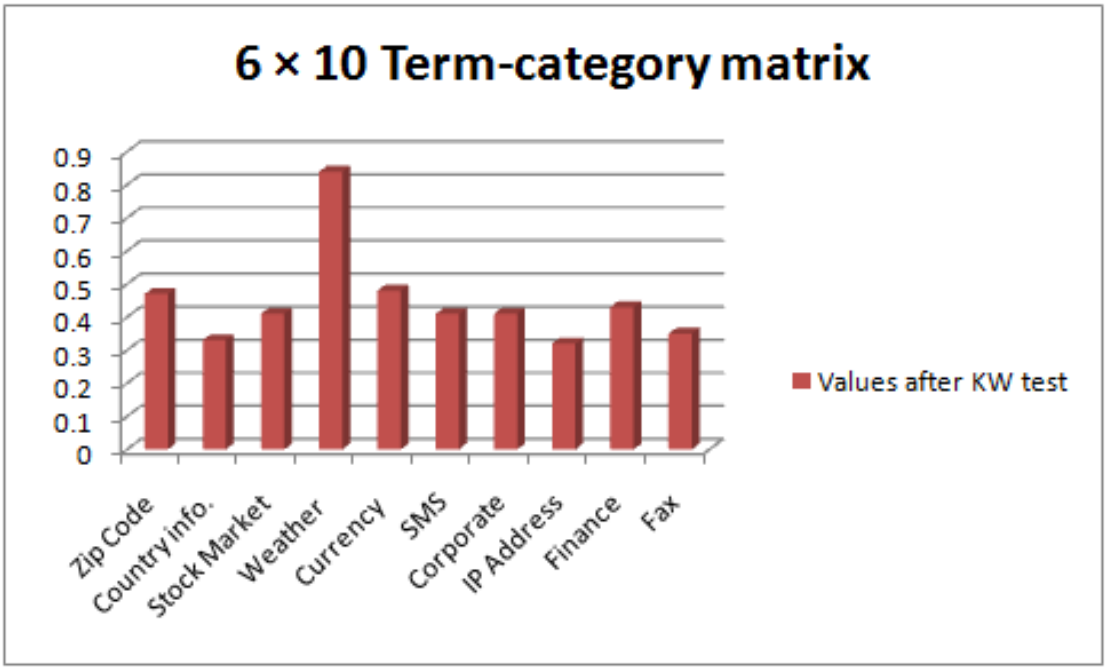

Figure 5.15: Median values for *six* terms and *ten* categories after applying Kruskal Wallis test

the new added categories. Values achieved and the service categorization is depicted in the Figure 5.12

Again the two newly added service increases the search scope for the service discoverer but service will still fall under category weather.

Now the set of candidate categorizes was further expanded to ten by adding a category *'Fax'* and NSS of all the terms was calculated with the new added category. Values achieved and the service categorization is depicted in the Figure 5.14, and 5.15

From Figure (5.14) it is clear that as number of services go on increasing search scope of the service discoverer or the end user goes on increasing. Now within financial sector service discoverer has three option *'Stock Market'*, *'Corporate'* and *'Finance'*.

Horizontal Expansion of Term Category Matrix

It was clear from Figure 5.9 that the service will be added to category *'Weather'* if 6×5 matrix is considered.

To check the horizontal scalability of the designed framework the number of terms in the annotation tag are expanded to *ten* and all terms are chosen which have much affinity to category *'Zip Code'*. Figure 5.16 displays the values achieved and Figure 5.17 displays in which category service will be placed after applying Krushkal Wallis test.

It is clear from Figure 5.17 that service will fall in category *'Weather'* and '*Zip Code*' both by applying soft categorization.

Now the number of terms in the annotation tag are expanded to *fifteen* by adding five terms closely related to category *'Weather'*. Figure 5.18 displays the values achieved and Figure 5.19 displays the medians after applying Krushkal Wallis test.

To expand the services further the number of terms in the annotation tag are expanded to *twenty* by adding five terms closely related to category *'Weather'*. Figure 5.20 displays the values achieved and Figure 5.21 displays the median values after applying Krushkal Wallis test.

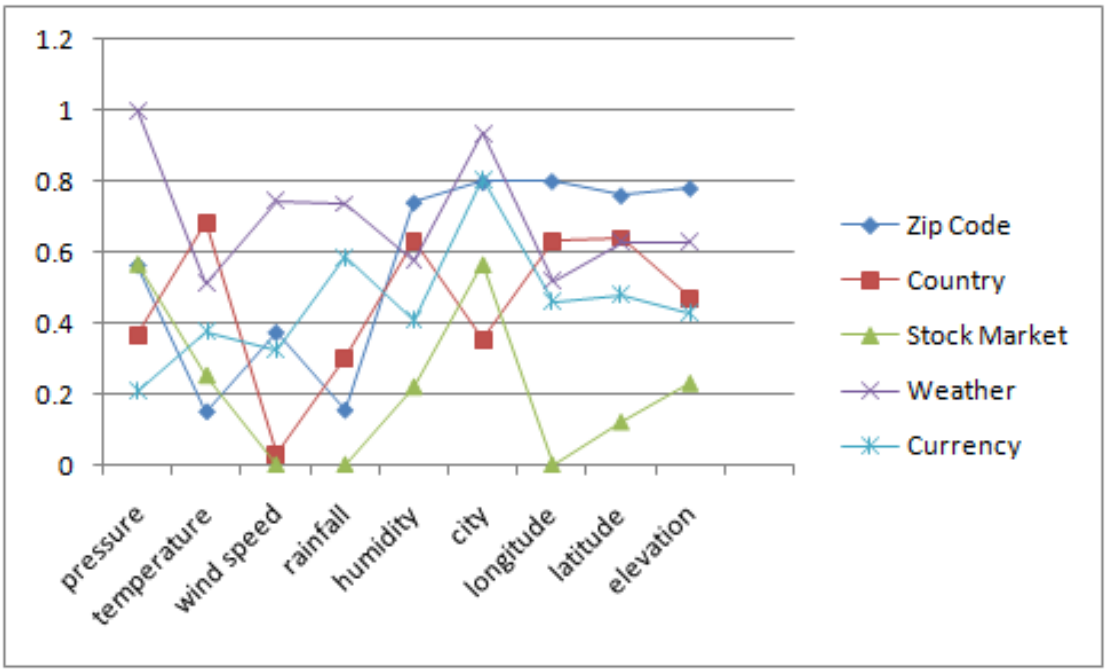

Figure 5.16: NSS values for *ten* terms and *five* categories

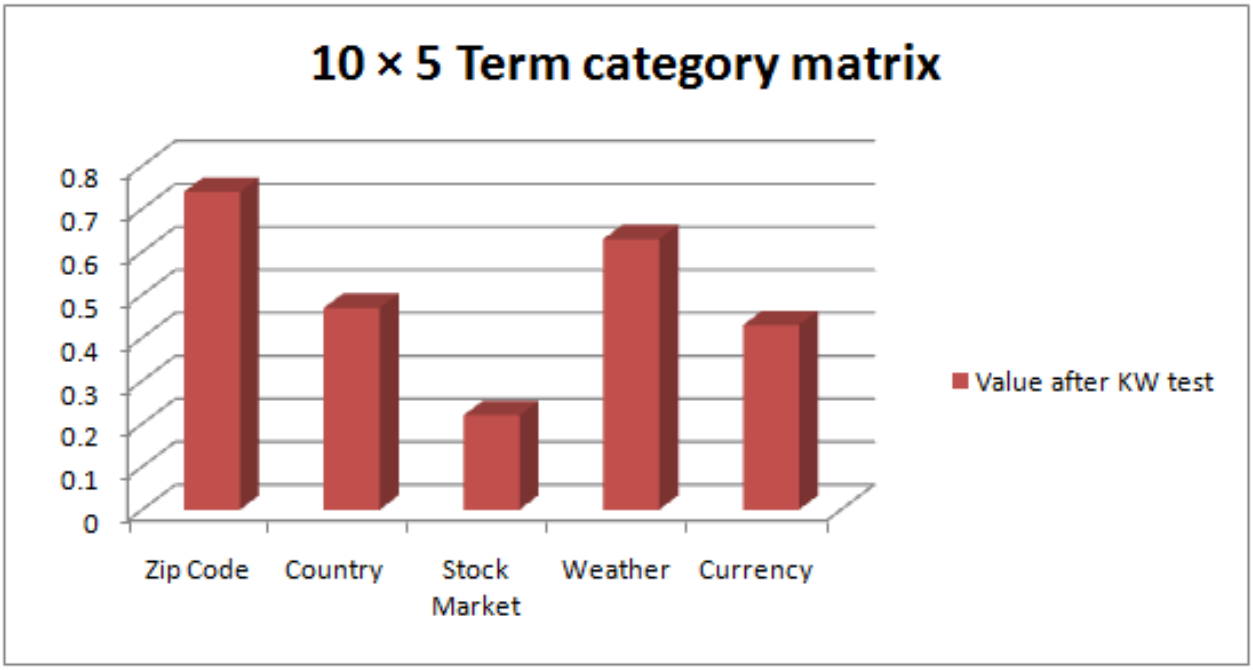

Figure 5.17: Median values for *ten* terms and *five* categories after applying Kruskal Wallis test

Next the terms are expanded from *twenty* to *twenty-five* again choosing terms closely related to category *Weather*. Figure 5.22 displays the values achieved and

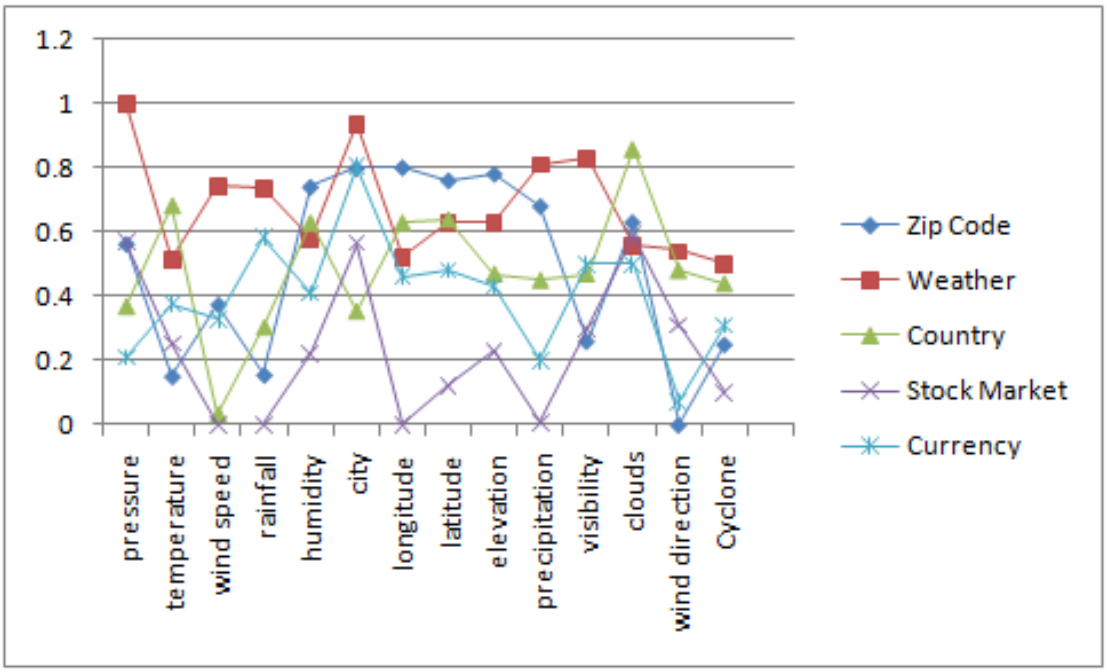

Figure 5.18: NSS values for *fifteen* terms and *five* categories

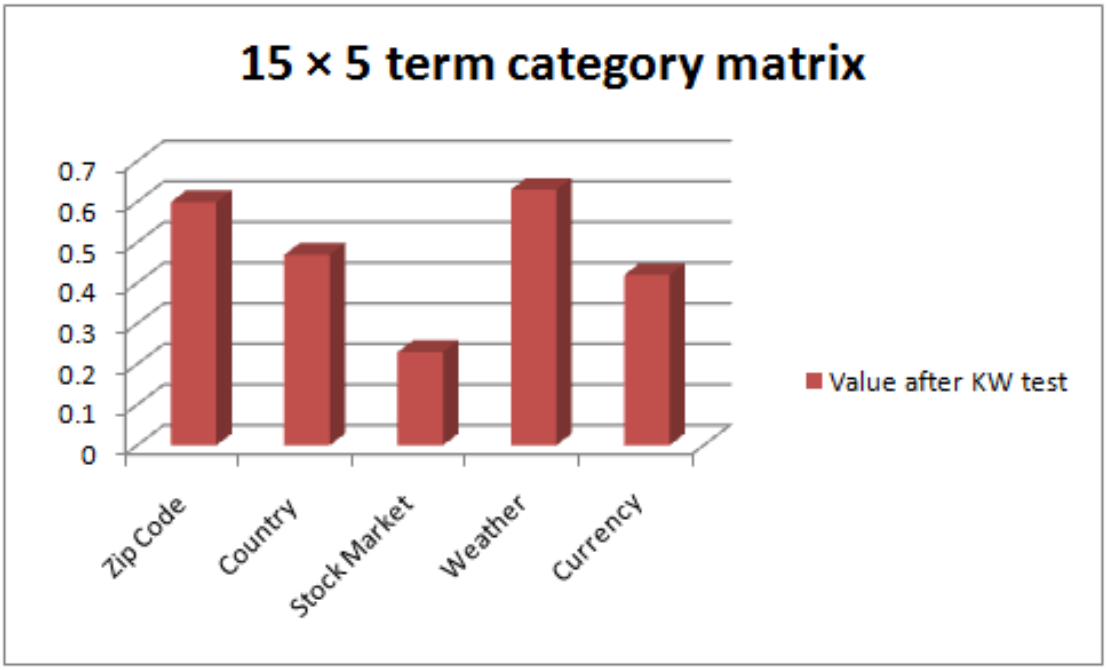

Figure 5.19: Median values for *fifteen terms* and *five* categories after applying Kruskal Wallis test

Figure 5.23 displays the categorization after applying Krushkal Wallis test.

Finally the terms in annotation tag are expanded to *thirty* and values of NSS

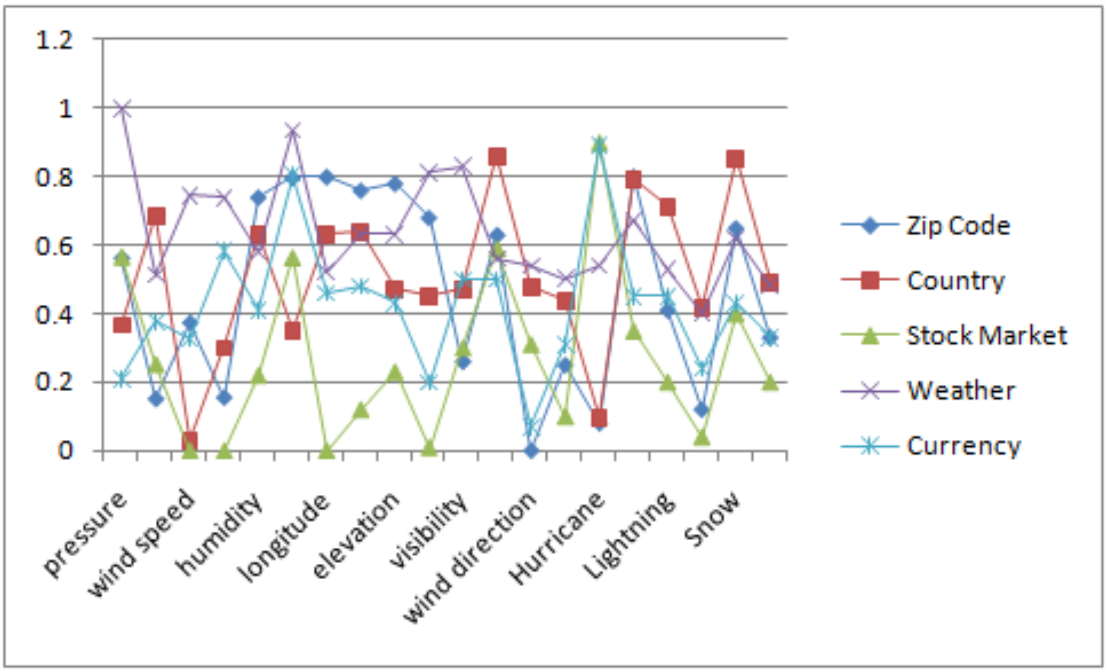

Figure 5.20: NSS values for *twenty terms* and *five* categories

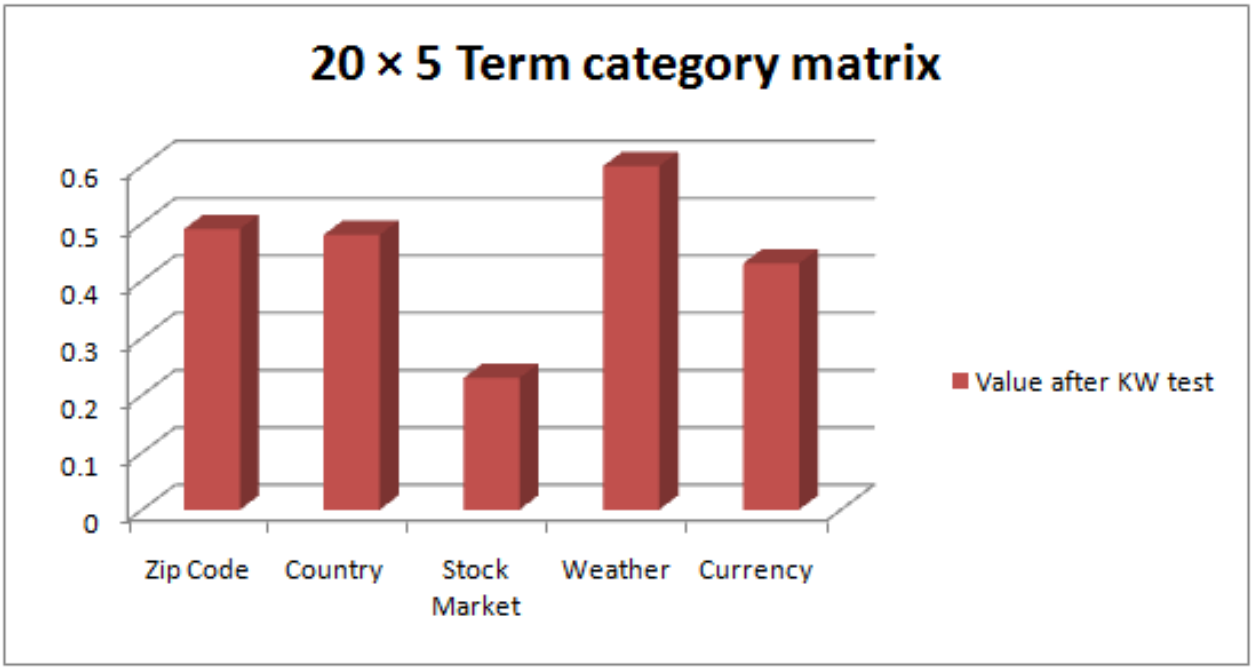

Figure 5.21: Median Values for *twenty terms* and *five* categories after applying Kruskal Wallis test

achieved for a term category-matrix of 30 × 5 along with median after applying Krushkal Wallis test is shown in Figure 5.24.

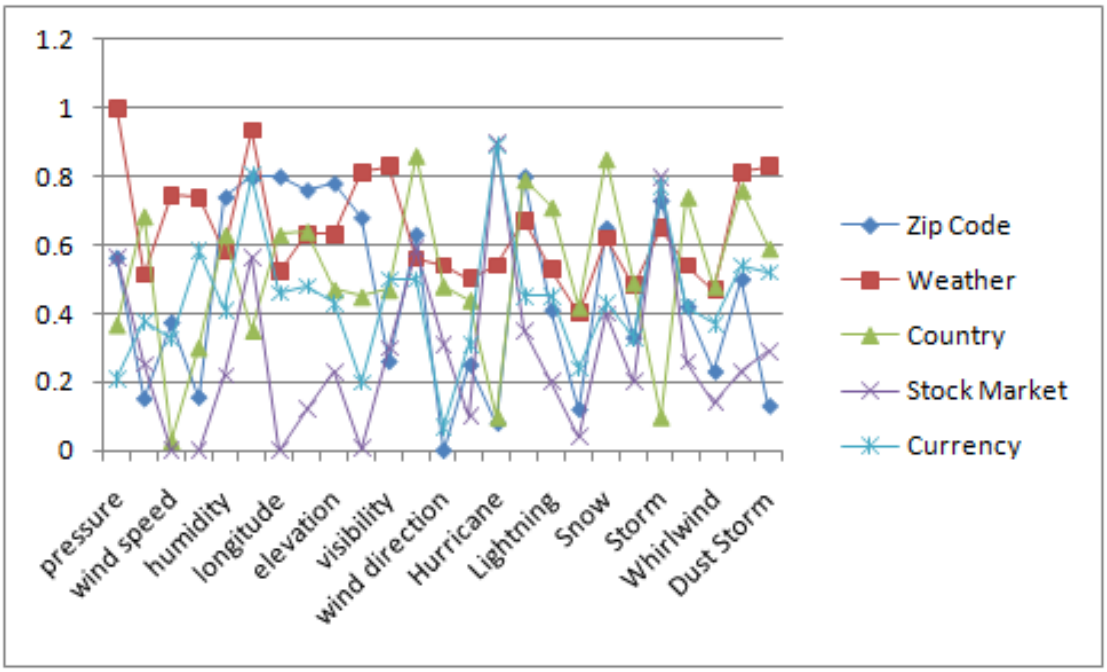

Figure 5.22: NSS values for *twenty five* terms and *five* categories

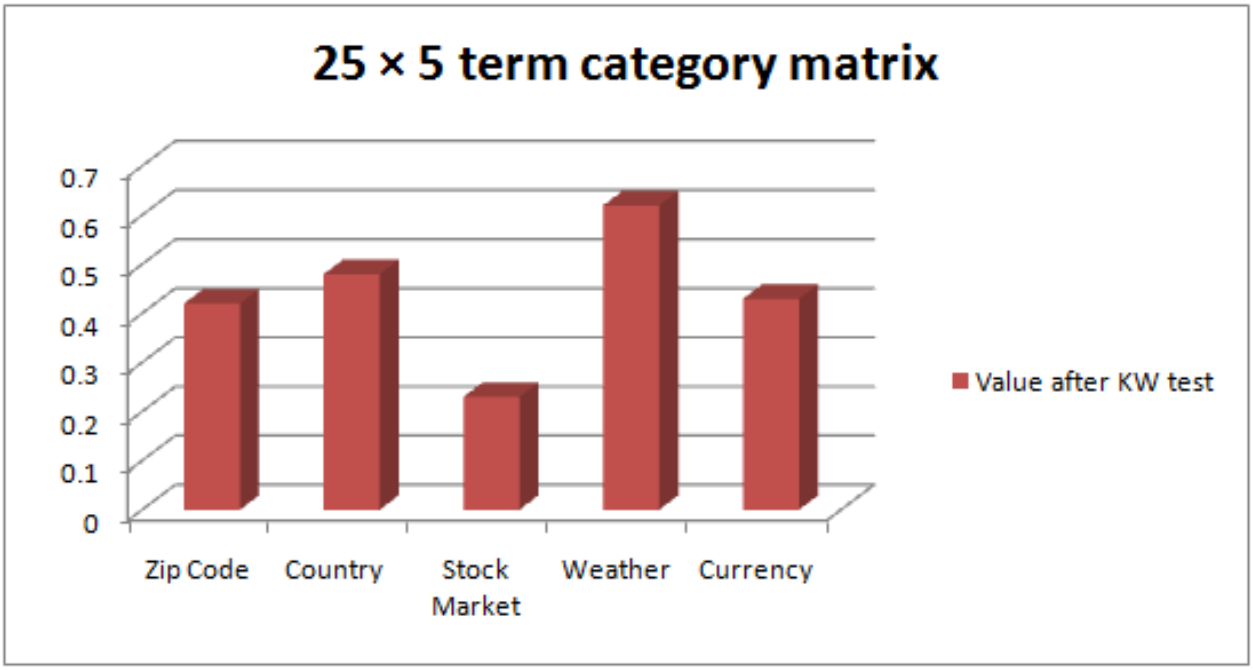

Figure 5.23: Median values for *twenty five* terms and *five* categories after applying Kruskal Wallis test

Vertical and Horizontal Expansion of term category matrix

In the end a 30 × 10 matrix i.e. a matrix of *thirty* terms compared with all the *ten*

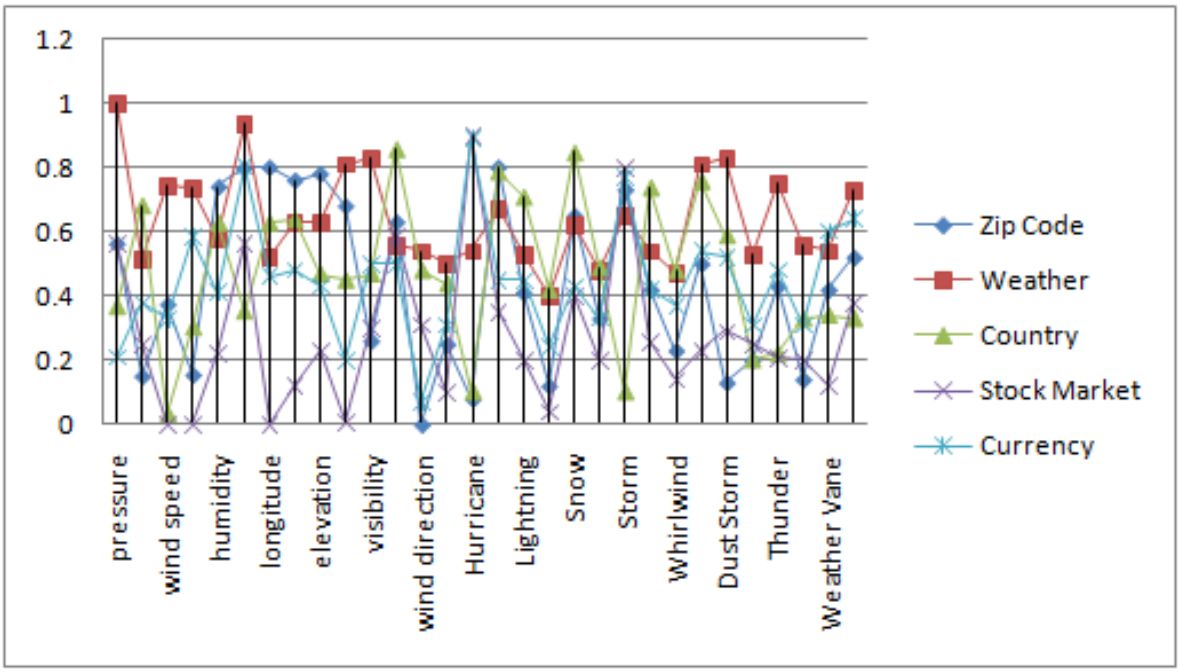

Figure 5.24: NSS values for *thirty terms* and *five* categories

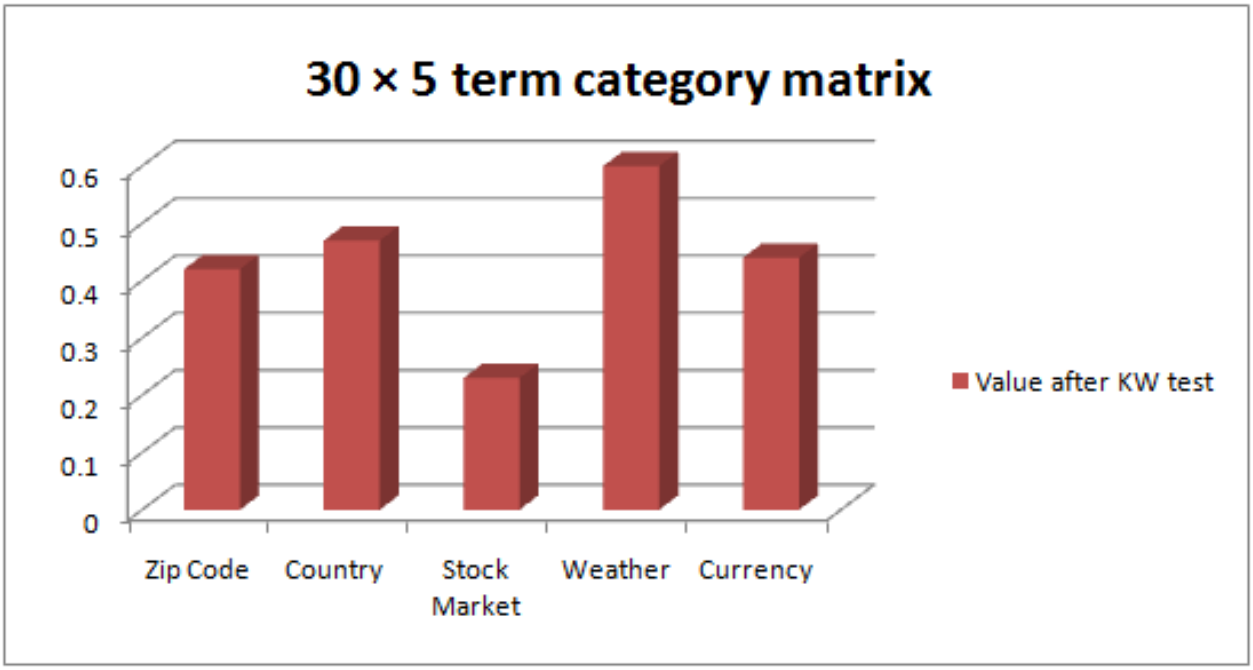

Figure 5.25: Median values for *thirty* terms and *five* categories after applying Kruskal Wallis test

categories was considered and Kruskal Wallis test was applied to this data set. Figure 5.26 depicts the data entries and Figure 5.27 graph depicts the values after applying

Kruskal Wallis test.

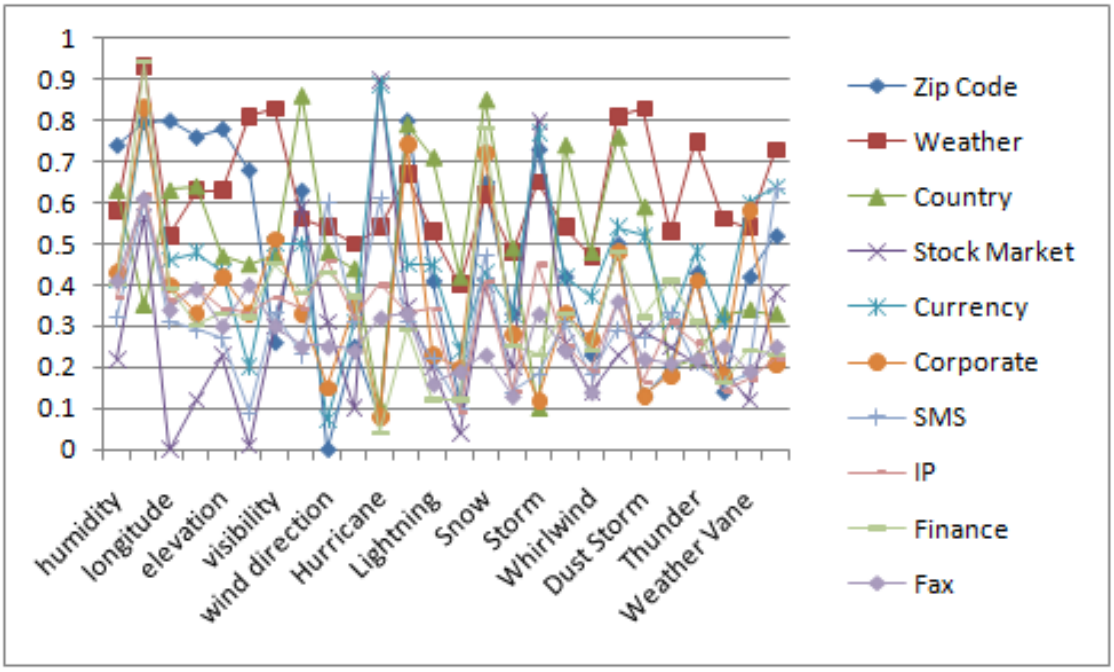

Figure 5.26: NSS values for *thirty* terms and *ten* categories

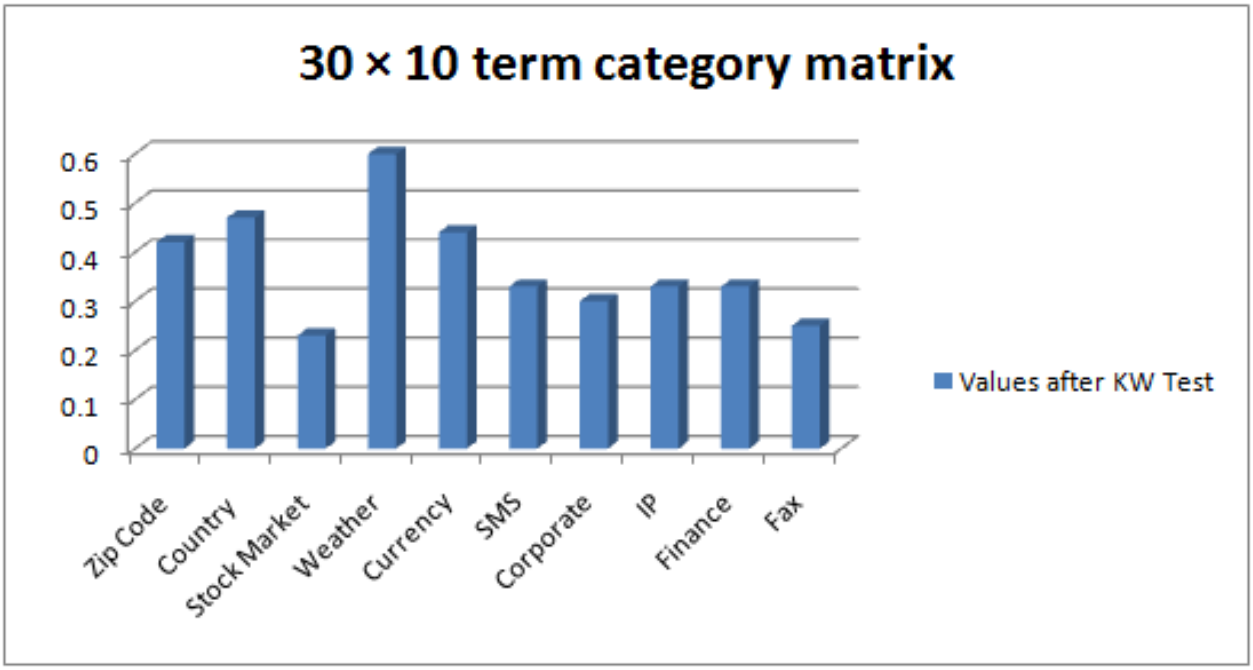

Figure 5.27: Median values for *thirty* terms and *ten* categories after applying Kruskal Wallis test

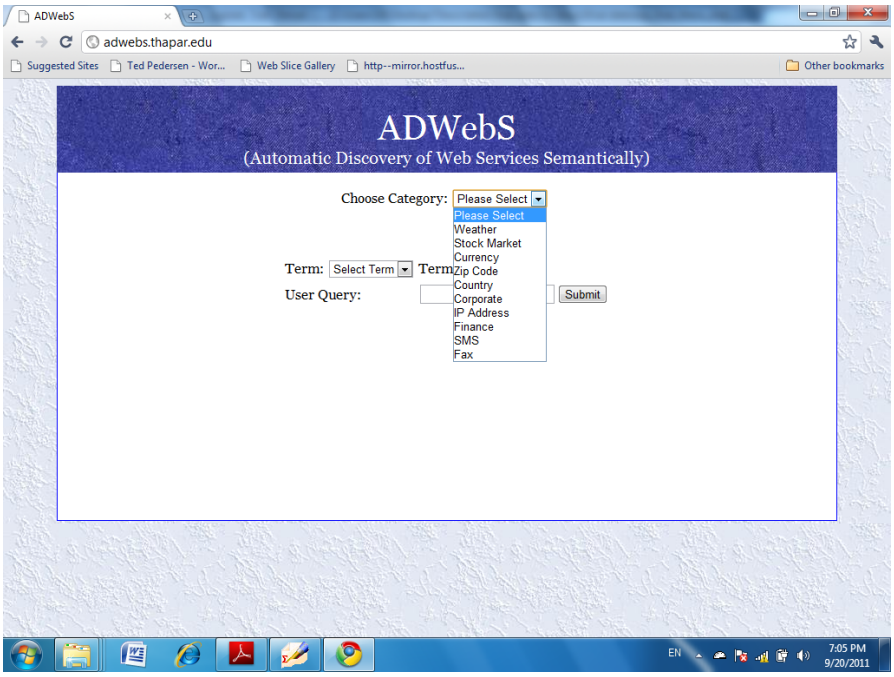

Figure 5.28: Selecting terms from *ten* categories in drop down

5.4.4 Processing time and memory usage

The only overhead introduced in following *ADWebS* approach is that when publishing a new Web service addition of semantic meta-data in the annotation tag is mandatory. Since the service publisher is best judge of the functional description of the service the time which may be required by the publisher can be comfortably ignored. The processing time required to categorize a service can be considered as an overhead but since only one time processing and categorization is required by UDDI, this time can also be ignored.

As the number of categories in *ADWebS* will increase the size of term-category matrix will naturally increase and hence time required to add a service to a category will increase but it will not affect the service discoverer in any form instead it will increase the search scope of the discoverer.

Memory usage will not be affected a lot accept for slight redundancy introduced by

using soft categorization whereby a service is categorized in to more than one candidate categories.

Chapter 6

Conclusions and Future Scope

6.1 Conclusions

The thesis proposes a framework named *'ADWebS'* (Automatic Discovery of Web Services Semantically) for semantic categorization and discovery of Web services. *ADWebS* classifies all the services into one or more candidate categories using combination of semantic and statistic measures. Semantic categorization has been accomplished by adding annotations in the WSDL of the Web services and finding semantic association between annotations and candidate categories.

To achieve the set objectives, a comprehensive review of developments related to Web services and semantic Web services discovery has been done and it is observed that addition of semantic annotations is necessary for efficient interpretation of services capabilities. A thorough study of matchmaking techniques used in syntactic and semantic service discovery is carried out and role of ontologies, description logics, machine learning approaches, *etc.* as matchmakers has been analyzed. Work done to find semantic relatedness between associated terms has been extensively studied and use of various knowledge corpus for finding semantic similarity between terms has been

reported. Following paragraphs provide an overview of the steps followed in design and development of *ADWebS*.

Entire framework has been divided into five steps i) Semantic Annotations addition ii) Annotations extraction iii) Finding Semantic relatedness iv) Categorization of services and v) Ranking. Inefficiency in capturing the semantic functionalities of services has been addressed in *ADWebS* by introducing annotations in the existing WSDL file of every Web service in a new tag called < annotations/>. These annotation serve as "add-on" to assure a valid support to the retrieval of functional capabilities of services. After annotating services, service provider publishes them in a registry. Advantage of adding annotation in a separate tag is that there is no need of language dependent preprocessing steps such as part-of-speech tagging or dependency parsing, which can be time consuming or even infeasible at very large scale.

Once the semantic descriptions are published, *ADWebS* extracts these terms and add a Web service into one or more candidate category(s). Initially five candidate category *Zip Code, Country, Stock Market, Weather and Currency* have been considered and all available services have been categorized into one of these category. To determine semantic association between the terms and candidate categories most frequently used and most reliable Web search engine 'Google' has been used. Normalized Google Distance (NGD), a probability based Measure of Semantic Relatedness (MSR) has been used to find semantic similarity between all the terms and categories and term category matrix is generated. Addition of service to one or more candidate category has been done using a statistical measure Kruskal Wallis test, a non-parametric test. Based on the median values achieved for each category after applying the test, a service is permanently allocated to one of the predefined category(s).

After the semantic categorization of the services is done, service discoverer is pro-

vided a 'Google' like interface for category selection. The frontend of *ADWebS* gives a drop down of candidate categories and once user selects a category, he is provided with two drop downs having terms related to the selected category along with a text box for adding query terms.

From the experimental results achieved it is observed that categorization of services reduces the services search time drastically since entire problem of finding relevant services is reduced to looking for similar services within a category. The precision rate of the services with categorization is very high compared to services without categorization. The framework is tested to be scalable both horizontally and vertically i.e increasing the number of terms and/or number of categories does not affect the service retrieval time or precision. In fact, as the number of categories increase the discoverer's search scope increases.

One of the most important contribution of *ADWebS* is that it follows incremental approach, *i.e.* it proposes an enhancement in the existing Web services standards, it does not propose any major change in the existing standards. Further, it significantly reduces the number of recommended services and increases the accuracy of search.

6.2 Future Scope

Adding semantics to the existing Web Services technologies is a fundamental requirement if one wants to deliver workable integration solutions for the next Web generation. Throughout the work the major focus was on easing the efforts of service discoverer by providing an efficient semantic discovery framework. On similar line, framework can be extended to provide semantic in other steps in the Web service lifecycle i.e. service composition, invocation, *etc. ADWebS* proposes addition of semantics by service pub-

lisher, which has been manually monitored in the design stage or semi-automatically extracted from the information provided in the published file, some additional efforts can be put to extract information provided by service publisher so that extracted terms give a precise and clear definition of the functional capabilities of the services being published.

The growing number of RESTful Web services available on the Web raises a challenging search problem as to how the desired Web services should be located. Main advantage of REST is its ease of implementation, agility of the design, and the lightweight approach to things. Another advantage of REST lies with performance: with better cache support, lightweight requests and responses, and easier response parsing, REST allows for nimbler clients and servers, and reduces network traffic, too. but again adding semantic in RESTful environment is a challenge and this needs to be explored further.

Entire framework has been focused around centralized registry like UDDI or ebXML, the implementation of the proposed technique in peer-to-peer architecture need to be considered in future.

Another important area which needs to be explored is how semantics can be useful in cloud computing. Using cloud computing platforms and technologies in conjunction with semantic Web technology and metadata can help popularize the usage of semantic metadata, while improving the semantics of the cloud computing landscape. Combining SOA models and practices with cloud computing technology and resources, as well as semantic Web technology innovation leads to a realm of potential for enhancing the interoperability, performance, and adaptability of modern-day automated solutions.

In the end it can be concluded that available specifications and technologies will have to go through the lengthy standardization process and real effort of consequent

prototype developments, before first commercial solutions for semantic Web services are available to the market. There is a high hope that in future the applications based on Semantic Web services will become commonplace, and a component architecture based on SOA will become the dominant development paradigm with semantics playing the main role.

Bibliography

[1] A. Ankolekar, M. Burstein, J. Hobbs et al., *DAML-S: Semantic Markup for Web Services*, International Semantic Web Working Symposium (SWWS), Stanford University, California, USA, July 30-August 1, 2001.

[2] A. Bernstein, M. Klein, *Toward High-Precision Service Retrieval*, IEEE Internet Computing, 8(1), pp. 30-36, 2004.

[3] A. Bose, R. Nayak, P. Bruza, *Improving Web Service Discovery by using Semantic Models*, Lecture Notes In Computer Science, pp. 366-380,Proceedings of the 9th International Conference on Web Information Systems Engineering, Auckland, New Zealand, 2008.

[4] A. Budanitsky, G. Hirst, *Semantic Distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures*, Workshop on WordNet and Other Lexical Resources, North American Chapter of the Association for Computational Linguistics (NAACL-2000), Pittsburgh, 2000.

[5] A. Budanitsky, G. Hirst, *Evaluating WordNet-based Measures of Semantic Distance*, Computational Linguistics, 32 (1), pp. 13-47, 2006.

[6] A. D. Birell, B. J. Nelson, *Implementing Remote Procedure Calls*, ACM Transactions on Computer Systems, 1(2), pp. 39-59, 1984.

[7] A. Fernandez, A. Polleres, S. Ossowski, *Towards Fine-grained Service Matchmaking by Using Concept Similarity*, International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2), 2007.

[8] A. H. Bouk, S. Li, N. Channa, *Comparative Study of Semantic Web Services*, Journal of Information and Communication Technology, 1(1), 2007.

[9] A. He, E. Johnston, N. Kushmerick, *Machine Learning Techniques for Annotating Semantic Web Services*, Dagstuhl Seminar on Machine Learning for the Semantic Web, 2005.

[10] A. He, E. Johnston, N. Kushmerick, *ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services*, In 3rd International Semantic Web Conference (2004)

[11] A. Haller, E. Cimpian, A. Mocan, E. Oren, C. Bussler, *WSMX - A Semantic Service- Oriented Architecture*, pp. 11-15,International Conference on Web Services (ICWS 2005), Orlando, FL, USA, July 2005.

[12] A. Hess, N. Kushmerick, *Learning to Attach Semantic Metadata to Web Services*, 2nd International Semantic Web Conference (ISWC 2003), Florida, USA, 2003.

[13] A. M. Zaremski, M. W. Jeannette, *Signature Matching: a Tool for Using Software Libraries*, ACM Transactions on Software Engineering and Methodology, 1995.

[14] A. M. Zaremski, M. W. Jeannette, *Specification Matching of Software Components*, ACM Transactions on Software Engineering and Methodology, 1997.

[15] A. Malik, *XML, Ontologies and the Semantic Web: The Second Generation of the Web*, XML Journal, January 2003.

[16] A. Rowstron, P. Druschel, *Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems*, pp. 329-350, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November 2001.

[17] A. Sajjanhar, J. Hou, Y. Zhang, *Algorithm for Web Services Matching,* The Sixth Asia Pacific Web Conference (APWeb'04), Hangzhou, China, April 14-17, 2004.

[18] A. Sheth, *Semantic Web Process Lifecycle: Role Of Semantics in Annotation, Discovery, Composition and Execution*, Invited talk at WWW Workshop on e-Services and Semantic Web, Budapest, Hungary, 2003.

[19] A. Tsalgatidou, T. Pilioura, *An Overview of Standards and Related Technology in Web Services*, Distributed and Parallel Databases, 12(2), pp. 135-162, 2002.

[20] A. V. Paliwal, N. R. Adam, H. Xiong, C. Bornhovd, *Web Service Discovery via Semantic Association Ranking and Hyperclique Pattern Discovery*, pp. 649-652, IEEE/WIC/ACM International Conference on Web Intelligence, 2006.

[21] B. Benatallah, M. S. Hacid, A. Leger, C. Rey, F Toumani, *On Automating Web Services Discovery*, VLDB Journal, 14(1), pp. 84-96, 2005.

[22] B. C. Grau, *A Possible Simplification of the Semantic Web Architecture*, WWW, New York, USA, 2004.

[23] B. N. Grosof, *A Courteous Compiler From Generalized Courteous Logic Programs To Ordinary Logic Programs*, IBM Report included as part of documentation in the IBM CommonRules 1.0 software toolkit and documentation, released on http://alphaworks.ibm.com, July, 1999.

[24] B. N. Grosof, *Semantic Web Services: Obstacles and Attractions*, Introduction to panel at WWW- 2003, 12th International Conference on World Wide Web, Budapest, Hungary, 2003.

[25] B. Medjahed, B. Benatallah, A. Bouguettaya, A.H.H. Ngu, A.K. Elmagarmid, *Business-to-Business Interactions: Issues and Enabling Technologies*, VLDB Journal, 12(1), pp. 59-85, 2003.

[26] B. N. Grosof, I. Horrocks, R. Volz, S. Decker, *Description Logic Programs: Combining Logic Programs with Description Logic*, International Conference on the World Wide Web (WWW-2003), Budapest, Hungary, 2003.

[27] http://www.bindingpoint.com

[28] C. Abela , M. Montebello, *DAML Enabled Web Services and Agents in the Semantic Web*, Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web- Services and Database Systems, LNCS, pp. 46-58, 2003.

[29] C. Atkinson, P. Bostan, O. Hummel, D. Stoll, *A Practical Approach to Web Service Discovery and Retrieval*, IEEE International Conference on Web services(ICWS), Salt Lake City, Utah, USA, July 9-13, 2007.

[30] C. Fellbaum (ed) *WordNet: An Electronic Lexical Database*, Bradford Books, 1998.

[31] C. H. Papadimitriou, P. Raghavan, H. Tamaki, S. Vempala, *Latent Semantic Indexing: A Probabilistic Analysis*, pp. 159-168, 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Washington, United States, 1998.

[32] C. Leacock, M. Chodorow, *Combining Local Context and Wordnet Similarity For Word Sense Identification in Wordnet*, An Electronic Lexical Database, MIT Press, pp. 265-283, 1998.

[33] C. Pautasso, O. Zimmermann, F. Leymann, *"RESTful Web" Services vs. "Big" Web Services*, IW3C2, Beijing, 2008.

[34] C. Preist, *A conceptual architecture for semantic web services*, 3rd International Semantic Web Conference (ISWC2004), November, 2004.

[35] C. Petrie, C. Bussler, *Service agents and virtual enterprises: a survey*, IEEE Internet Computer, 7(4), pp. 68-78, 2003.

[36] C. Schmidt, M. A. Parashar, *Peer-to-Peer Approach to Web Service Discovery*, World Wide Web: Internet and Web Information Systems, 7(2), pp. 211-229, 2004.

[37] C. Zhou, L. Chia, B. Lee, *Service Discovery and Measurement based on DAML-QoS Ontology*, Special Interest Tracks and Posters of 14th World Wide Web Conference, 2005.

[38] D. Austin, A. Barbir , C. Ferris, S. Garg (Eds.), *Web Service Architecture Requirements, W3C Working Group Notes*, 2004. (http://www.w3.org/TR/wsa-reqs.)

[39] D. Bollegala, Y. Matsuo, M. Ishizuka, *Measuring Semantic Similarity between Words Using Web Search Engines*, International World Wide Web Conference (IW3C), Alberta, Canada, May 8-12, 2007.

[40] D. Bollegala, Y. Matsuo, M. Ishizuka, *Measuring the Similarity between Implicit Semantic Relations using Web Search Engines*, WSDM'09, Barcelona, Spain, February 9-12, 2009.

[41] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard D (eds.), *Web Services Architecture*, W3C WG Note, 2004.(http://www.w3.org/TR/ws-arch/)

[42] D. C. Hoong, R. Buyya, *Guided Google: A Meta Search Engine and its Implementation using the Google Distributed Web Services*, International Journal of Computers and Applications, ACTA Press, 26(3), pp. 181-187,2004.

[43] D. Chimenti, R. Gamboa, R. Krishnamurthy S. A. Naqvi, S. Tsur, C. Zaniolo, *The LDL System Prototype*, IEEE Transactions on Knowledge and Data Engineering, 2(1), pp. 76-90, 1990.

[44] Delicious: http://www.del.icio.us

[45] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, J. Domingue, *Enabling Semantic Web Services: The Web Service Modeling Ontology*, Springer, 2007.

[46] D. Lin, *Automatic Retreival and Clustering of Similar Words*, pp. 768-774, 17th COLING, Montreal, Quebec, Canada, August 10-14, 1998.

[47] D. Lin, *An Information-Theoretic Definition of Similarity*, pp. 296-304, 15th ICML, Wisconsin, USA, July 24-27, 1998.

[48] D. Martin et al., *OWL-S: Semantic Markup for Web Services*, W3C Member Submission, 22 November 2004. (Available from http://www.w3.org/Submission/OWL-S/)

[49] D. McDermott, *DRS: A set of conventions for representing logical languages in RDF*, January 2004. (Available from http://www.daml.org/services/owls/1.1B/DRSguide.pdf)

[50] D. Milne, I. Witten, *An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links*, AAAI 08 Workshop on Wikipedia and Artificial Intelligence, Chicago, USA, 2008.

[51] D. Oberle, *Semantic management of middleware*, Springer, 2006.

[52] D. S. Coalition, *DAML-S: Web Service Description for the Semantic Web*, ISWC, Sardinia, Italy, 10-12 June, 2002.

[53] D. Kourtesis, I. Paraskakis, *Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery*, LNCS, Vol. 5021, pp. 614-628, 2008.

[54] D. Yang, M.W. David, Powers, *Measuring Semantic Similarity in the Taxonomy of WordNet*, 28th Australasian Computer Science Conference, (ACSC2005), Australia, Australian Computer Society, Vol. 38, 2005.

[55] E. Al-Masri, Q. H. Mahmoud, *QoS-based Discovery and Ranking of Web Services*, pp. 529-534, IEEE 16th International Conference on Computer Communications and Networks (ICCCN), Honolulu, Hawaii, USA, August 13-16, 2007.

[56] *ebXML, Enabling A Global Electronic Market* http://www.ebxml.org/

[57] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana (eds.), *Web Services Description Language (WSDL) 1.1* (http://www.w3.org/TR/wsdl), March 2001.

[58] E. Gabrilovich, S. Markovitch, *Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis*,pp. 1606-1611, Internationl Joint Conference on Artificial Intelligence (IJCAI-07), 2007.

[59] E. Iosif, A. Potamianos, *Unsupervised Semantic Similarity Computation using Web Search Engines*, IEEE/WIC/ACM International Conference on Web Intelligence, Silicon Valley, USA, November 2-5, 2007.

[60] E. Iosif, *Unsupervised Semantic Similarity Computation Between Terms Using Web Documents*, IEEE Transactions on Knowledge and Data Engineering, 22(11), pp. 1637-1647, Sep. 2009.

[61] E. Maximilien, M. Singh, *A Framework and Ontology for Dynamic Web Services Selection*, IEEE Internet Computing, 8(5), pp. 84-93, 2004.

[62] E. Motta, M. Sabou, *Next Generation Semantic Web Applications*, 1st Asian Semantic Web Conference, Beijing, China, LNCS, Springer, Heidelberg, Sep. 2006.

[63] E. Petrakis, G. Varelas, A. Hliaoutakis, P. Raftopoulou, *Xsimilarity: Computing Semantic Similarity Between Concepts from Different Ontologies*, Journal of Digital Information Management, 4(4), pp. 233-238, 2006.

[64] E. Rahm, P. Bernstein, *A Survey of Approaches to Automatic Schema Matching*, VLDB Journal, 10(4), pp. 334-350, 2001.

[65] E. Sirin, J. Hendler, B. Parsia, *Semi-Automatic Composition of Web Services using Semantic Descriptions*, Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS, pp. 17-24, 2003.

[66] E. Stroulia, Y.Wang, *Semantic Structural Matching for Accessing Web Service Similarity*, International Journal of Cooperative Information Systems, 14(4), pp. 407-437, 2005.

[67] http://www.esynaps.com

[68] E. Teich, P. Fankhauser, *Wordnet for Lexical Cohesion Analysis*, pp. 326-331, Second Global Wordnet Conference, Brno, Czech Republic, 2004.

[69] *Extensible Markup Language: XML*, XML Technical Committee, 2000. (http://www.w3.org/TR/REC-xml)

[70] F. Banaei-Kashani, C. Chen, C. Shahabi, *WSPDS: Web Services Peer-to-Peer Discovery Service* International Symposium on Web Services and Applications (ISWS), Las Vegas, NV, USA, 2004.

[71] F. Giunchiglia, P. Shvaiko, M. Yatskevich, *S-match: an algorithm and an implementation of semantic matching*, pp. 61-75, Proceedings of the European Semantic Web Symposium (ESWC), 2004.

[72] F. Keller, M. Lapata, *Using the Web to Obtain Frequencies for Unseen Bigrams*, Computational Linguistics, 29(3), pp. 459-484, 2003.

[73] $http://www.flickr.com$

[74] F. p. Coyle, *XML, Web Services, and the Data Revolution*, Addison Wesley, 2002.

[75] G. Agre, T. Atanasova, J. Nern *Migrating from Web Services to Semantic Web Services: Infrawebs Approach*, 1st Workshop on Semantic Web Applications, EUROMEDIA 2005, IRIT, Universit Paul Sabatier, Toulouse, France, April 11-13, 2005.

[76] G. Alonso, F. Casati, H. Kuno, V. Machiraju, *Web Services: Concepts, Architecture, and Applications*, Springer, Berlin Heidelberg, New York, 2003.

[77] G. Hirst, D. St-Onge, *Lexical Chains as Representations of Context for Detection and Correction of Malapropisms*, Christaine Fellbaum, (Ed.) Word- Net: An Electronic Lexical Database, pages 305-332. The MIT Press, Cambridge, Massachusetts, 1998.

[78] G. Miller, W. Charles, *Contextual Correlates of Semantic Similarity* , Language and Cognitive Processes, 6(1), pp. 1-28, 1998.

[79] http://www.google.com

[80] H. Chen, M. Lin, Y. Wei, *Novel Association Measures using Web Search with Double Checking*, pp. 1009-1016, COLING/ACL, 2006.

[81] H. Lausen, T. Haselwanter, *Finding Web Services*, 1st European Semantic Technology Conference, Vienna, Austria, 2007.

[82] H. Schutze, J. O. Pedersen, *A Cooccurrence-Based Thesaurus and Two Applications to Information Retreival*, Information Processing and Management, 33(3), pp. 307-318, 1997.

[83] I. Clarke, *Freenet: A Distributed Anonymous Information Storage and Retrieval System*, Workshop on Design Issues in Anonymity and Unobservability, Lecture Notes in Computer Science, Springer, pp. 46-66, 2000.

[84] I. Constantinescu, B. Faltings, *Efficient Matchmaking and Directory Services*, IEEE/WIC International Conference on Web Intelligence, 2003.

[85] I. Constantinescu, W. Binder, B. Faltings, *Flexible and Efficient Matchmaking and Ranking in Service Directories*, IEEE International Conference on Web Services(ICWS'05), Orlando, Florida, USA, July 11-15, 2005.

[86] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof , M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Member Submission 21 May 2004.

[87] I. Matveeva, G. Levow, A. Farahat, C. Royer, *Term Representation with Generalized Latent Semantic Analysis*, Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria, September 21-23, 2005.

[88] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, pp. 149-160, 2001 SIGCOMM Conference, San Diego, CA, USA, August 2001.

[89] J. Cardoso, *Quality of Service and Semantic Composition of Work*, Ph.D. Thesis, University of Georgia, 2002.

[90] J. de Bruijn (ed.), *The Web Service Modeling Language WSML*, WSMO Final Draft v0.2, 2005. (http://www.wsmo.org/TR/d16/d16.1/v0.2/.)

[91] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. Konig-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, M. Stollberg, *Web Service Modeling Ontology (WSMO)*, W3C Member Submission 3 June 2005. (http://www.w3.org/Submission/WSMO/)

[92] J. Cardoso, *Semantic Web Services: Theory, Tools, and Applications*, IGI Global, 2007.

[93] J. Cardoso, A. Seth (Eds.), *Semantic Web Services, Processes and Applications*, Springer, 2006.

[94] J. Colgrave, R. Akkiraju, R. Goodwin, *External matching in UDDI*, Proceedings of the IEEE International Conference on Web Services, 2004.

[95] J. Davies, R. Studer, P. Warren, *Semantic Web Technologies, Trends and Research in Ontology-based Systems*, John Wiley and Sons Ltd., 2006.

[96] J. F. Roddick, K. Hornsby, D. de Vries, *A Unifying Semantic Distance Model for Determining the Similarity of Attribute Values*, 26th Australasian Computer Science Conference (ACSC2003), Vol. 16, Adelaide, Australia, 2003.

[97] J. Garofalakis, Y. Panagis, E. Sakkopoulos, A. Tsakalidis, *Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?*, International Workshop on Web Engineering, Munich, Germany, July 28-30, 2004.

[98] J. Gonzalez-Castillo, D. Trastour, C. Bartolini, *Description Logics for Match Making of Services*, HP Technical Reports, October 2001.

[99] J. Gracia, R. Trillo, M. Espinoza, E. Mena, *Querying the Web: A Multiontology Disambiguation Method*, pp. 241.248, 6th International Conference on Web Engineering, Palo Alto, California, USA, 2006.

[100] J. Gracia, E. Mena, *Web-Based Measure of Semantic Relatedness*, J. Bailey et al. (Eds.), WISE-2008, LNCS 5175, pp. 136-150, 2008.

[101] J. Javier Samper, F. Javier Adell, Leo van den Berg, J. Jos Martinez, *Improving Semantic Web Service Discovery*, Journal of Networks, 3(1), January, 2008.

[102] J. Jiang, D. Conrath, *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy*, International Conference on Research on Computational Linguistics, (ROCLING X), Taiwan, 1997.

[103] J. Kopecky, E. Simperl, D. Fensel, *Semantic Web Service Offer Discovery*, pp. 314-322, OTM Confederated International Conference on The Move to Meaningful Internet Systems, Vol. I, Vilamoura, Portugal, 2007.

[104] J. Lin, D. Gunopulos, *Dimensionality Reduction by Random Projection and Latent Semantic Indexing*, Third SIAM International Conference on Data Mining, San Francisco, CA, USA, 2003.

[105] J. M. Gomez-Berbis, R. C. Palacios, J. L. Cuadrado, I. J. Carrasco, A.G. Crespo, *SEAN: Multi-ontology semantic annotation for highly accurate closed domains.* International Journal of Physical Sciences, 6(6), pp. 1440-1451., 2011.

[106] J. Ma, J. Cao, Y. Zhang, *A Probabilistic Semantic Approach for Discovering Web Services*, pp. 1221-1222, 16th International Conference on World Wide Web, Banff, Alberta, Canada, 2007.

[107] J. Ma, J. Cao, Y. Zhang, *A Probabilistic Semantic Approach for Discovering Web Services*, WWW, Banff, Alberta, Canada, May 8-12, 2007,

[108] J. Ma, Y. Zhang, J. He, *Efficiently Finding Web Services Using a Clustering Semantic Approach*, CSSSIA, Beijing, China, April 22, 2008.

[109] J. Madhavan, P. Bernstein, E Rahm, *Generic Schema Matching with Cupid*, 27th VLDB Conference, Italy, Rome, 2001.

[110] J. R. Firth, *A Synopsis of Linguistic Theory 1930-55*, Studies in Linguistic Analysis (special volume of the Philological Society), Oxford, The Philological Society, pp. 1-32, 1957.

[111] J. W. Lloyd, *Foundations of Logic Programming*, Springer Series in Symbolic Computation. Springer-Verlag, New York, 1987, (Second, Extended Edition).

[112] J. Wu, Z. Wu, *Similarity-based Web Service Matchmaking*, pp. 287-294, IEEE International Conference on Services Computing, Florida, USA, July 11-15, 2005.

[113] J. Yu, H. Su, G. Zhou, K. Xu, *SNet: Skip Graph based Semantic Web Services Discovery*, pp. 1393-1397, ACM Symposium on Applied Computing, Seoul, Korea, 2007.

[114] J. Pathak, N. Koul, D. Caragea, V. G. Honavar, *A Framework for Semantic Web Services Discovery*, WIDM'05, Bremen, Germany, November 5, 2005,

[115] K. Church, P. Hanks, *Word Association Norms, Mutual Information and Lexicography*, Computer Linguist, 16(1), pp. 22-29, 1990.

[116] K. Decker, K. Sycara, M. Williamson, *Middle-Agents for the Internet*, 15th IJCAI. Nagoya, Japan, pp. 578-583, 1997.

[117] K. Sivashanmugam, K. Verma, J.A. Miller, *Adding Semantics to Web Services Standards*, pp. 395-401, International Conference on Web services, ICWS'03, 2003.

[118] K. Sivashanmugam, K. Verma, R. Mulye, Z. Zhong, A. Sheth, *Speed-R: Semantic P2P environment for diverse Web Service Registries*, 2004.

[119] K. Sycara, M. Klusch, S. Widoff, J. Lu, *Larks: Dynamic Matchmaking among Heterogeneous Software Agents in Cyberspace*, Autonomous Agents and Multi-Agent Systems, 5(2), Kluwer, 2002.

[120] K. Sycara, M. Paolucci, M. van Velsen, J. Giampapa, *The RETSINA MAS Infrastructure*, Technical Report CMU-RI-TR-01-05, Robotics Institute Technical Report, Carnegie Mellon, 2001.

[121] K. Sycara, M. Klusch, S. Widoff, J. Lu, *Dynamic Service Matchmaking among Agents in Open Information Environments*, A. Ouksel, A. Sheth (Eds.), Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems, 1999.

[122] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar J. Miller, *METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services*, Journal of Information Technology and Management, Special Issue on Universal Global Integration, 6(1), Kluwer Academic Publishers, pp. 17-39, 2005.

[123] L. Bai, M. Liu, *Fuzzy sets and similarity relations for semantic web service matching*, Computer Mathematics with Applications, 61, pp. 2281-2286, 2011

[124] L. Cabral, J. Domingue, E. Motta, T. Payne and F. Hakimpour *Approaches to Semantic Web Services:An Overview and Comparisons*, pp. 225-239, ESWS 2004

[125] L. Gong, *JXTA: A Network Programming Environment*, IEEE Internet Computing, (5)3, pp. 88-95, May/June 2001.

[126] L. Kovacs, A. Micsik, P. Pallinger, *Handling User Preference and Added Value in Discovery of Semantic Web Services*, pp. 225 - 232, IEEE International Conference of Web Services, ICWS, Utah, USA, July 9th - 13th, 2007.

[127] L. Li, I. Horrocks, *A Software Framework for Matchmaking Based on Semantic Web Technology*, International Journal of Electronic Commerce, 8(4), pp. 39-60, 2004.

[128] L. Moreau, A. Avila-Rosas, V. Dialani, S. Miles, X. Liu, *Agents for the Grid: A Comparison with Web Services (part II: Service Discovery)*, pp. 52-56, Proceedings of Workshop on Challenges in Open Agent Systems , Italy, 2002.

[129] L. Reeve, H. Han, *Survey of Semantic Annotation Platforms*, pp.1634-1638, ACM symposium on Applied computing, Santa Fe, SA New Mexico, 2005.

[130] M. A. Salahli, *An Approach For Measuring Semantic Relatedness Between Words Via Related Terms*, Mathematical and Computational Applications, 14(1), pp. 55-63, 2009.

[131] M. Armugam, A. Sheth, I. B. Arpinar, *Towards peer-to-peer Semantic Web: A distributed environment for sharing semantic knowledge on the Web*, 11th International World Wide Web Conference, Honolulu, Hawaii, USA, 7-11 May, 2002.

[132] M. Bruno, G. Canfora et al., *An Approach to Support Web Service Classification and Annotation*, IEEE International Conference on e-Technology, e-Commerce and e-Service, Hong Kong, China, 29 March-1 April, 2005.

[133] M. C. Daconta, L. J. Obrst, K. T. Smith, *The Semantic Web: A guide to the future of XML, Web Services and Knowledge Management*, Wiley, 2003.

[134] M. Carman, L. Serafini, P. Traverso, *Web Service Composition as Planning*, Workshop on Planning for Web Services, Trento, Italy, June, 2003.

[135] M. Crasso, A. Zunino, M. Campo, *Awsc: An approach to Web service classification based on machine learning techniques*, Inteligencia Artificial, Revista Iberoamericana de IA, 12 (37), pp. 25-36, 2008.

[136] M. Damashek, *Gauging Similarity with N-Grams*, Language-Independent Categorization of Text Science, Vol. 267, pp. 843-848, 1995.

[137] M. Elkstein, *Learn REST: A Tutorial*, http://learnrest.blogspot.com/2008/02/what-is-rest.html

[138] M. F. Moens, *Information Extraction: Algorithms and Prospects in a Retrieval Context*, Springer, 2006.

[139] M. F. Porter, *An Algorithm for Suffix Stripping*, Program, 14, pp. 130-137, 1980.

[140] M. Genesereth. KIF- Knowledge Interchange Format: Draft proposed American national standard (dpANS). Technical Report NCITS.T2/98-004, ANSI KIF Ad Hoc Group, 1998. Available from http://logic.stanford.edu/kif/dpans.html.

[141] M. H. Nodine, J. Fowler, J. Ksiezyk, B. Perry, M. Taylor, A. Unruh, *Active Information Gathering In Infosleuth*, International Journal of Cooperative Information Systems, 9(1-2), pp. 3-28, 2000.

[142] M. Klusch, B. Fries, K. Sycara, *Automated Semantic Web Service Discovery with OWLS-MX*, AAMAS, Hakodate, Hokkaido, Japan, May 8-12, 2006.

[143] M. Jarmasz, S. Szpakowicz, *Rogets Thesaurus and Semantic Similarity*, pp. 212-219, Proc. of Recent Advances in Natural Language Processing (RANLP-03), Borovets, Bulgaria,September 10-12, 2003.

[144] M. Kifer, G. Lausen, J. Wu, *Logical Foundations of Object-Oriented and Frame Based Languages*, JACM, 42(4), pp. 741-843, 1995.

[145] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, D. Fensel, *A Logical Framework for Web Service Discovery*, 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan, 2004.

[146] M. Klein, B. Koenig-Ries, *Coupled Signature and Specification Matching for Automatic Service Binding*, pp. 183-197, European Conference on Web Services, Springer, LNAI, 2004.

[147] M. Lapata, F. Keller, *Web-based Models of Natural Language Processing*, ACM Transactions on Speech and Language Processing, 2(1), 2005.

[148] M. Lesk, *Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone*, pp. 24-26, 5th Annual International Conference On Systems Documentation (SIGDOC '86), 1986.

[149] M. P. Singh, M.N. Huhns, *Service-Oriented Computing Semantics, Processes, Agents*, Wiley and Sons, New York, 2005.

[150] M. P. Papazoglou, J. Dubray, *A Survey of Web Service Technologies*, Technical report DIT-04-058.

[151] M. Paolucci, T. Kawamura, T. R. Payne, K. Sycara, *Importing the Semantic Web in UDDI*, Web Services, E-Business and Semantic Web Workshop, International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002.

[152] M. Paolucci, T. Kawamura, T. Payne, K. Sycara, *Semantic Matching of Web Services Capabilities*, The CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002.

[153] M. Rodrguez, M. Egenhofer, R. Rugg, *Assessing Semantic Similarities among Geospatial Feature Class Definitions*, A. Vckovski, K. Brassel , H. J. Schek (Eds.), pp. 189-202, Second International Conference on Interoperating Geographic Information Systems, INTEROP'99, LNCS, Vol. 1580, 1999.

[154] M. Rodrguez, M. A. Egenhofer, *Putting Similarity Assessment into Context: Matching Distance with the User's Intended Operations*, P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti F. Castellani (Eds.), 2nd International and Interdisciplinary Conference on Modeling and Using Context, CONTEXT-99, Vol. 1688, pp. 310-323, Lecture Notes in Artificial Intelligence, Springer, Trento, Italy, 1999.

[155] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, *A Bayesian approach to filtering junk e-mail*, Learning for Text Categorization, pp. 55-62, AAAI Workshop, 1998.

[156] M. Schlosser, M. Sintek, S. Decker, Nejdl, *A Scalable and Ontology-based P2P Infrastructure for Semantic Web Services*, Second International Conference on Peer-to-Peer Computing (P2P'02), 2002.

[157] M. Strube, S. Ponzetto, *Exploiting Semantic Role Labeling, Wordnet and Wikipedia for Coreference Resolution*, Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pp. 192-199, New York, 2006.

[158] M. Strube, S Ponzetto, *Wikirelate! Computing Semantic Relatedness using Wikipedia*, AAAI Press, 2006.

[159] M. W. Berry, S. T. Dumais, G. W. O'Brien, *Using Linear Algebra for Intelligent Information Retrieval*, SIAM Review, 37(4), pp. 573-595, 1995.

[160] M. Sabou, J. Pan, *Towards Improving Web Service Repositories through Semantic Web Techniques*, Workshop on Semantic Web Enabled Software Engineering (SWESE), 7 Nov 2005.

[161] M. Stollberg, U. Keller, *Semantic Web Service Discovery*, DERI Technical Report, October 2006.

[162] N. Fukuta, T. Ito, *Comparing Semantic Web Service Frameworks in a Context of Auction Services*, International Journal of Computer Science and Network Security, 6(4), April 2006.

[163] N. Kokash, A. Birukou, V. D'Andrea, *Web Service Discovery Based on Past User Experience*, 10th International Conference on Business Information Systems, pp. 95-107, BIS 2007, Poznan, Poland, April 25th - 27th, 2007.

[164] N. Kokash, *A Comparison Of Web Service Interface Similarity Measures*, Frontiers in Artificial Intelligence and Applications, Vol. 142, pp. 220-231, Proceeding of conference on STAIRS, IOS Press, 2006.

[165] N. Mitra, (ed.) *SOAP Version 1.2* (http://www.w3.org/TR/2003/REC-soap12-part0-20030624/) W3C Recommendation, June 2003.

[166] *North American Industry Classification System*, http://www.census.gov/epcd/www/naics.html

[167] *OASIS UDDI Specifications TC - Committee Specifications*, 2002. (http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm)

[168] *OWL Web Ontology Language Reference*, W3C Recommendation, World Wide Web Consortium, 2004. (http://www.w3.org/TR/owl-ref/)

[169] OMG, *Unified Modeling language (UML)*, Version 1.5, http://www.omg.org/technology/documents/formal/uml.htm

[170] P. Cimano, S. Handschuh, S. Staab, *Towards the Self-Annotating Web*, 13th WWW, New York, NY, USA, May 17-20, 2004.

[171] P. K. Bhowmick, D. Roy, S. Sarkar, A. Basu, *A Framework for Manual Ontology Engineering for Management of Learning Material Repository*, International Journal of Computer Science and Applications, Special Issue on Web Semantics Ontology-II, 7(2), pp. 30-51, 2010.

[172] P. Mika, *Ontologies are us: A Unified Model of Social Networks and Semantics*, ISWC, Osaka, Japan, IEEE Computer Society, October 18-21, 2005.

[173] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*, 14th International Joint Conference on Artificial Intelligence (IJCAI-95), pp. 448-453, Morgan Kaufmann, San Francisco, CA, 1995.

[174] P. Resnik, *Semantic Similarity in Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language*, Journal of Artificial Intelligence Research, Vol. 11, pp. 95-130, 1999.

[175] P. Resnik, N. A. Smith, *The Web as a Parallel Corpus*, Computational Linguistics, 29(3), pp. 349 -380, 2003.

[176] P. Turney, *Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL*, L. De Raedt P. Flach (Eds.), pp. 491-502, Twelfth European Conference on Machine Learning (ECML-2001), Freiburg, Germany, 2001.

[177] P. Turney, *Similarity of Semantic Relations*, 32(3), Association for Computational Linguistics, 2006.

[178] P. Turney, *Expressing Implicit Semantic Relations Without Supervision*, Coling/ACL'06, pp. 313-320, 2006.

[179] P. Vitanyi, Universal Similarity, pp. 238-243, IEEE ISOC ITW 2005 on Coding and Complexity, Rotorua, New Zealand, 2005.

[180] P. Weinstein, W. Birmingham, *Agent communication with differentiated ontologies: eight new measures of description compatibility*, Technical report, Department of Electrical Engineering and Computer Science, University of Michigan, 1999.

[181] PSL Technical Committee, *Process specification language (PSL)*, 2003

[182] Qi Yu, Xumin Liu, Athman Bouguettaya, Brahim Medjahed, *Deploying and Managing Web Services: Issues, Solutions, and Directions*, The VLDB Journal, Springer, Vol. 17, pp. 537-572, 2008.

[183] R. Akkiraju, R. Goodwin, P. Doshi, S. Roeder, *A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI*, IJCAI-03 Workshop on Information Integration on Web, 2003.

[184] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. Schmidt, A. Sheth, K. Verma, *Web Service Semantics - WSDL-S*, W3C Member Submission 7 November 2005. (http://www.w3.org/Submission/WSDL-S/)

[185] R. Baeza-Yates, B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, 1999

[186] R. Cilibrasi , P. Vitanyi, *The Google Similarity Distance*, IEEE Transactions on Knowledge and Data Engineering, 19(3), pp. 370-383, 2007.

[187] R. Cilibrasi, P. Vitanyi, *Normalized Web Distance and Word Similarity*, Handbook of Natural Language Processing, Second Edition, Nitin Indurkhya and Fred J. Damerau Eds., CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010

[188] RDF Vocabulary Description Language 1.0: RDF Schema, W3C, (http://www.w3.org/TR/rdf-schema/), 2004.

[189] Resource Description Framework (RDF) (http://www.w3.org/RDF/), 2002.

[190] R. Feldman, I. Dagan, H. Hirsh, *Mining Text using Keyword Distributions*, Journal of Intelligent Information System, 10(3), pp. 281-300, 1998.

[191] R. Gligorov, Z. Aleksovski, W. ten Kate, *Using Google Distance to Weight Approximate Ontology Matches*, WWW 2007, Alberta, Canada, May 8-12, 2007.

[192] R. Guo, J. Le, X. Xia, *Capability Matching of Web Services Based on OWL-S*, 16th International Workshop on Database and Expert Systems Applications (DEXA'05), 2005.

[193] R. Lara, M. Angel Corella, P. Castells, *A Flexible Model for Web Service Discovery*, 1st International Workshop on Semantic Matching and Resource Retrieval - Issues and Perspectives, Seoul, South Korea, September 11, 2006.

[194] R. Lindsey, V.D. Veksler, A. Grintsvayg, W.D. Gray *Be Wary of What Your Computer Reads: The Effects of Corpus Selection on Measuring Semantic Relatedness*, 8th International Conference of Cognitive Modeling (ICCM) , Ann Arobor, MI, 2007.

[195] R. Nayak, *Using Data Mining In Web Services Planning, Development and Maintenance*, International Journal of Web Services Research, Vol. 5, pp. 62-80, 2008.

[196] R. Nayak, B. Lee, *Web Service Discovery with additional Semantics and Clustering*, IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), Silicon Valley, USA, 2007.

[197] R. R. Korfhage, *Information Storage and Retrieval*, John Wiley Sons, 1997

[198] R. Richardson, A. Smeaton, J. Murphy, *Using Wordnet as a Knowledge Base for Measuring Semantic Similarity Between Words*, Technical Report Working Paper CA-1294, School of Computer Applications, Dublin City University, 1994.

[199] R. Roy, H. Mili, E. Bicknell, M. Blettner, *Development and Application of a Metric on Semantic Nets*, IEEE Transactions on Systems, Man, and Cybernetics, 19(1), pp. 17-30, 1989.

[200] R. Saraolu, K. Ttnc, N. Allahverdi, *A Fuzzy Clustering Approach for Finding Similar Documents using a Novel Similarity Measure*, Expert Systems With Applications: An International Journal, 33(3), pp. 600-605, 2007.

[201] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, University of California, 2000.

[202] R. T. Laura, P. I. Sergio, S. Bergamaschi, E. Mena, *Using semantic techniques to access web data*, Information System, 36, pp. 117-133, 2011.

[203] S. A. McIlraith, D. Plexousakis, F. van Harmelen, (Eds.), *A Conceptual Architecture For Semantic Web Services*, pp. 395-409, Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, Vol. 3298 of LNCS, Springer, Berlin/Heidelberg, Nov. 2004.

[204] S. Agarwal, R. Studer, *Automatic Matchmaking of Web Services*, pp. 45-54, IEEE International Conference of Web Services (ICWS), Chicago, USA, September 18-22, 2006.

[205] S. Bajaj et al., *Web Services Policy Attachment (WS-PolicyAttachment)*, Technical report, BEA, IBM, Microsoft, SAP, Sonic Software, Verisign, September, 2004.

[206] S. Banerjee, T. Pedersen, *An adapted Lesk algorithm for word sense disambiguation using WordNet*, pp. 136-45, Proceedings of the third international conference on intelligent text processing and computational linguistics. Mexico City, Mexico, 2002.

[207] S. Bansal, J. Vidal, *Matchmaking of Web Services based on the DAML-S Service Model*, Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), July 15-19, 2002, Bologna, Italy.

[208] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, S. Tabet, *Semantic Web Services Framework (SWSF) Overview*, W3C Member Submission, http://www.w3.org/Submission/SWSF/, September 2005.

[209] S. Chaiyakul, K. Limapichat, A. Dixit, E. Nantajeewarawat, *A Framework for Semantic Web Service Discovery and Planning*, CIS, 2006.

[210] S. Chernov, T. Iofciu, W. Nejdl, X. Zhou, *Extracting Semantic Relationships between Wikipedia Categories*, SemWiki2006 Workshop, collocated with ESWC, 2006.

[211] S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, M. Mongiello, *Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an e-Marketplace*, 6th International Conference on Electronic Commerce (ICEC), 2004.

[212] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Hashman, *Indexing by Latent Semantic Indexing*, Journal of the American Society for Information Science, 1990.

[213] Semantic Web Activity Statement, (http://www.w3.org/2001/sw/Activity), 2001.

[214] S. J. Vaughan-Nichols, *Web Services: Beyond the Hype*, IEEE Computer, 35(2), pp. 18-21 2002.

[215] S. Kulkarni, D. Caragea, *Computation of the Semantic Relatedness Language Combining OWL and RuleML*, 2003. (Available at http://www.daml.org/2003/11/swrl/)

[216] S. Kumar, R.B. Mishra, *Semantic Web Service Composition*, IETE Technical Review, 25(3), May-Jun 2008.

[217] S. Lamparter, B. Schnizler, *Trading Services in Ontology-driven Markets*, pp. 1679-1683, ACM symposium on Applied Computing, Dijon, France, 2006.

[218] S. McIlraith, T. Son, H. Zeng, *Semantic Web Services*, IEEE Intelligent Systems, Special Issue on the Semantic Web, 16(2), pp. 46-53, 2001.

[219] S. McIIraith, T. Son, H. Zeng, *Mobilizing the Semantic Web with DAML-Enabled Web Services*, Semantic Web Workshop, Hongkong, China, 2001.

[220] S. Mohammad, G. Hirst, *Distributional Measures as Proxies for Semantic Relatedness*, Kluwer Academic Publishers, 2005.

[221] S. Overhage, *On Specifying Web Services Using UDDI Improvements*, 3rd Annual International Conference on Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World Net Object Days, Germany, 2002.

[222] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, *A scalable content-addressable network*, pp. 161-172, ACM SIGCOMM, San Diego, CA, USA, August, 2001.

[223] S. Sharma, S. Batra, *Applying Association Rules For Web Services Categorization*,International Journal of Computer and Electrical Engineering, 2(3), pp. 465-468, June 2010.

[224] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D. F. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*, Prentice Hall PTR, 2005.

[225] S. Yoshida, T. Yukawa, K. Kuwabara, *Constructing and Examining Personalized Cooccurrence-based Thesauri on Web pages*, 12th International International World Wide Web Conference, Budapest, Hungary, May 20-24, 2003.

[226] S. Yu, J. Zhang, X. Ge, G. Wu, *Semantics Based Web Services Discovery*, LNCS, Vol.3309, pp. 388-393, 2004.

[227] S. Zhang, O. Bodenreider, *Alignment of multiple ontologies of anatomy: Deriving indirect mappings from direct mappings to a reference*, pp. 864-868, AMIA Symposium Proceedings, 2005.

[228] http://www.strikeiron.com

[229] T. Berners-Lee, *Semantic Web Road Map*, September 1998. ¡http://www.w3.org/DesignIssues/Semantic.html¿.

[230] T. Berners-Lee, *Weaving the Web*, Harper, San Francisco, 1999.

[231] T. Berners-Lee, J. Hendler, et al., *The Semantic Web*, Scientific American, 2001.

[232] T. Bellwood et el., *UDDI version 3.0* , (http://uddi org/pubs/uddi-v3.00-published-20020719.htm), July, 2002.

[233] T. Erl, *Service-Oriented Architecture, Concepts, Technology, and Design*, Prentice Hall, Indiana, 2006

[234] T. Hofmann, *Probabilistic Latent Semantic Analysis*,pp. 50-57, Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval, Berkeley, California, ACM Press, 1999.

[235] T. Hofmann, *Probabilistic Latent Semantic Indexing*, Proceedings of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval, 1999.

[236] T. K. Landauer, S. T. Dumais, *A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*, Psychological Review, 104(2), pp. 211-240, 1997.

[237] T. Kawamura, J. Blasio, T. Hasegawa, M. Paolucci, K. Sycara, *Public Deployment of Semantic Service Matchmaker with UDDI Business Registry*, pp. 752-766, 3rd International Semantic Web Conference (ISWC2004), LNCS 3298, Springer-Verlag, Berlin, 2004.

[238] T. R. Gruber, *A Translation Approach To Portable Ontologies*, Knowledge Acquisition, 2(5), pp. 199-220, 1993.

[239] T. R. Payne, M. Paolucci, K. Sycara, *Advertising and Matching DAML-S Service Descriptions*, Semantic Web Working Symposium (SWWS), 2001.

[240] T. Subramaniam, H. A. Jalab, A. Y. Taqa , *Overview of textual anti-spam filtering techniques*, International Journal of Physical Sciences, 5(12), pp. 1869-1882, 2010.

[241] T. Syeda-Mahmood, G. Shah, R. Akkiraju, A. Ivan, R. Goodwin, *Searching Service Repositories by Combining Semantic and Ontological Matching*, Third International Conference on Web Services (ICWS), Florida, 2005.

[242] T. Tsalgatidou, A. Pilioura, *, An Overview of Standards and Related Technology in Web Services*, Distributed and Parallel Databases, Vol. 12, pp. 135 -162, 2002.

[243] T. Veale, *Wordnet sits the sat: A Knowledge-based Approach to Lexical Analogy*, pp. 606-612, Proc. of ECAI'04, 2004.

[244] T. Zesch , G. I. Mller Christof, *Using Wiktionary for Computing Semantic Relatedness*, pp. 861-868, Proceedings of AAAI, 2008.

[245] *The United Nations Standard Products and Services Code*, http://www.unspsc.org/

[246] U. Chukmol, *A Framework for Web Service Discovery: Service's Reuse, Quality, Evolution and User's Data Handling*, Proceedings of the Second SIGMOD Ph.D. Workshop on Innovative Database Research (IDAR 2008), Vancouver, Canada, June 13, 2008.

[247] U. Keller, R. Lara, H. Lausen, A. Polleres, D. Fensel, *Automatic Location of Services*, 2nd European Semantic Web Conference, LNCS 3532, 2005.

[248] U. S. Manikrao, T.V.Prabhakar, *Dynamic Selection of Web Services with a Recommendation System*, International Conference on Next Generation Web Services Practices, Seoul, Korea, August 2005.

[249] V. D. Veksler, Ryan Z. Govostes, Wayne D. Gray, *Defining the Dimensions of the Human Semantic Space*, pp. 1282-1287, 30th Annual Meeting of the Cognitive Science Society, Washington D.C., USA, 2007.

[250] V. N. Vapnik, H. Druck, D. Wu, *Support Vector Machines for spam categorization*, IEEE Trans. on Neural Networks, 10(5), pp. 1048-1054, 1999.

[251] W. Abramowicz, K. Haniewicz, M. Kaczmarek, D. Zyskowski, *Architecture for Web Services Filtering and Clustering*, Internet and Web Applications and Services, (ICIW '07), 2007.

[252] W. Chen, M. Kifer, D. S. Warren, *HiLog: A foundation for higher-order Logic Programming*, Journal of Logic Programming, 15(3), pp. 187-230, 2003.

[253] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string distance metrics for name-matching tasks. IJCAI-03 Workshop on Information Integration on the Web (IIWeb), Acapulco, August, 2003.

[254] W3C:Web Services Architecture, (http://www.w3.org/TR/wsarch/),2003.

[255] What is Web 2.0?,: http:// www.oreilly.com/pub/a/oreilly/tim/news/2005

[256] http://www.webservices.org/

[257] http://www.webservicex.net

[258] *Wikipedia. Representational State Transfer.* http://en.wikipedia.org/wiki/REST

[259] X. Dong, A. Halevy, J. Madhavan, E. Nemes, J. Zhang, *Similarity Search for Web Services*, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[260] X. Du, W. Song, M. Zhang, *A Context-based Framework and Method for Learning Object Description and Search*, Proc. of 6th International Conference on Web-based Learning (ICWL 2007), LNCS Vol. 4823, Springer, Edinburgh, United Kingdom, Aug. 15-17, 2007.

[261] X. Huang, A. G. Ridgewell, D. Sharma, *Efficacious Transmission Technique for XML Data on Networks*, International Journal of Computer Science and Network Security, 6(3B), pp. 14-19, 2006.

[262] X. Su, *Semantic Enrichment for Ontology Mapping*, Hermelan, 2004.

[263] *http://www.xmethods.com/*

[264] Y. Hao, Y. Zhang, J. Cao, *WSXplorer: Searching for desired Web services*, pp. 173-187, 19th International Conference on Advanced Information System Engineering, CaiSE 2007, Trondheim, Norway, June 2007.

[265] Y. Li, Z. Bandar, D. McLean, *An Approach For Measuring Semantic Similarity Between Words Using Multiple Information Sources*, IEEE Trans. on Knowledge and Data Engineering, 15(4), pp. 871-882, 2003.

[266] Y. Li, F. Zou, Z. Wu, F. Ma, *PWSD: A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network*, pp. 291-300, Proc. APWeb04, LNCS 3007, 2004,

[267] Y. Matsuo, T. Sakaki, K. Uchiyama, M. Ishizuka, *Graph-based Word Clustering using Web Search Engine*, Proceedings of EMNLP 2006.

[268] Y. Zhou, L. Zhang, B. Xie, H. Mei, *User Feedback-Based Refinement for Web Services Retrieval using Multiple Instance Learning*, pp. 471 - 478, IEEE International Conference of Web Services, ICWS 2006, Chicago, USA, September 18th-22nd, 2006.

[269] http://www.yahoo.com

[270] Z. Aleksovski, M. Klein, W. ten Kate, F. van Harmelen, *Matching unstructured vocabularies using a background ontology*, pp. 182-197, Proceedings of Knowledge Engineering and Knowledge Management (EKAW), 2006.

[271] Z. Aleksovski, W. ten Kate, F. van Harmelen, *Exploiting the structure of background knowledge used in ontology Matching*, Workshop at International Semantic Web Conference (ISWC), 2006.

[272] Z. Harris, *Mathematical Structures of Language*, Interscience Publishers, 1968.

[273] Z. Hu, *Using Ontology to Bind Web Services to the Data Model of Automation Systems Revised Papers from the NODe*, pp. 154-168, Web and Database-Related Workshops on Web, Web- Services, and Database Systems, 2002.

[274] Z. Kedad, E. Metais, *Dealing with Semantic Heterogeneity during Data Integration*, J. Akoka, B. Mokrane, I. Comyn-Wattiau E. Metais (Eds.), Eighteenth International Conference on Conceptual Modelling, Vol. 1728, pp. 325-339, LNCS, Springer, Paris, France, 1999.

[275] Z. Zhuang, P. Mitra, A. Jaiswal, *Corpus-based Web Services Matchmaking*, Proceeding of the 20th National Conference on Artificial Intellience AAAI 2005, Pennsylvania, USA. Copyright American Association for Artificial Intelligence, July 9-13, 2005.

List of Publications

In Referred Journals

Published

1) Shalini Batra, Seema Bawa, *Semantic Categorization of Web Services*, International Journal of Recent Trends in Engineering, Vol. 2, No. 2, pp. 19-23, Nov. 2009.

2) Shalini Batra, Seema Bawa, *Web Services Categorization using Normalized Similarity Score*, International Journal of Computer Theory and Engineering, Vol. 2, No. 1, pp. 139-142, Feb. 2010.

3) Shalini Batra, Seema Bawa, *Review of Machine Learning Approaches to Semantic Web Service Discovery*, Journal of Advances in Information Technology, Special Issue on Advances in Ubiquitous Computing, Vol. 1, No. 3, pp. 146-151, August 2010.

In Book

4) Shalini Batra, Seema Bawa, *Using LSI and its Variants in Text Classification*, Advanced Techniques in Computing Sciences and Software Engineering, Ed. Khaled Elleithy, Springer, pp. 313-316, 2010.

Accepted

5) Shalini Batra, Seema Bawa, *Semantic Discovery of Web Services using Principal Component Analysis*, International Journal of the physical Science, (ISI Indexed Journal; Impact Factor: 0.554), Vol. 6(18), pp. 4466-4472, September, 2011

In International Workshop

6) Shalini Batra, Seema Bawa, *A Framework for Semantic Discovery of Web Services*, 5th International Workshop on Ubiquitous and Collaborative Computing, Scotland (UK), Sep. 2010. Published online with the British Computing Society (eWIC).

Appendix

<?xml version = "1.0" encoding = "utf − 8"? >
< wsdl : definitions xmlns : http = "http : //schemas.xmlsoap.org/wsdl/http/"
xmlns : soap = "http : //schemas.xmlsoap.org/wsdl/soap/"
xmlns : s = "http : //www.w3.org/2001/XMLSchema"
xmlns : soapenc = "http : //schemas.xmlsoap.org/soap/encoding/"
xmlns : tns = "http : //ws.fraudlabs.com/"
xmlns : tm = "http : //microsoft.com/wsdl/mime/textMatching/"
xmlns : mime = "http : //schemas.xmlsoap.org/wsdl/mime/"
targetNamespace = "http : //ws.fraudlabs.com/"
xmlns : wsdl = "http : //schemas.xmlsoap.org/wsdl/" >
< wsdl : types >
< s : schema elementFormDefault = "qualified"
targetNamespace = "http : //ws.fraudlabs.com/" >
< s : element name = "ZIPCodeWorldUS">
< s : complexType >
< s : sequence >
< s : element minOccurs = "0" maxOccurs = "1" name = "ZIPCode" type = "s : string"/ >
< s : element minOccurs = "0" maxOccurs = "1" name = "LICENSE" type = "s : string"/>
< /s : sequence >
< /s : complexType >
< s : element minOccurs = "0" maxOccurs = "1" name =" ZIPCode" type = "s : string"/>
< s : element minOccurs = "0" maxOccurs = "1" name = "LICENSE" type = "s : string"/>
< /s : sequence >
< /s : complexType >
< /s : element >
< s : element name = "ZIPCodeWorldUSResponse">
< s : complexType >
< s : sequence >
< s : element minOccurs = "0" maxOccurs = "1" name = "ZIPCodeWorldUSResult" type="tns:ZIPCODEWORLDUS" />
< /s : sequence >
< /s : complexType >

$< /s : element >$
$< s : complexTypename =$ "$ZIPCODEWORLD$US"$>$
$< s : sequence >$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CREDITSAVAILABLE"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "ZIPCODE" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$STATE"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CITY"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$AREA$CODE" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CITY$ALIASABBR" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CITY$ALIASNAME" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$STATE$FIPS" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$COUNTY$FIPS" type="s:string" />
$< s : elementminOccurs =$ "$1"maxOccurs =$ "$1"name =$ "$LATITUDE"type =$ "$s : float"/>$
$< s : elementminOccurs =$ "$1"maxOccurs =$ "$1"name =$ "$LONGITUDE"type =$ "$s : float"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CITY$TYPE" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$COUNTY$NAME" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$TIME$ZONE" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "DAYLIGHTSAVING" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$ELEVATION"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$MSA2000"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$PMSA"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CBSA"type =$ "$s : string"/>$
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CBSA$DIV" type="s:string" />
$< s : elementminOccurs =$ "$0"maxOccurs =$ "$1"name =$ "$CBSA$TITLE" type="s:string"/>

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$PERSONS$PERHOUSEHOLD” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$ZIPCODE$POPULATION” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$COUNTRIES$AREA” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$HOUSEHOLDS$PERZIPCODE” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$WHITE$POPULATION” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$BLACK$POPULATION” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$HISPANIC$POPULATION” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$INCOME$PERHOUSEHOLD” type=“s:double” />

$<$ s : $elementminOccurs$ = “1″$maxOccurs$ = “1″$name$ = “$AVERAGE$HOUSEVALUE” type=“s:double” />

$< s : elementminOccurs$ = “0″$maxOccurs$ = “1″$name$ = “$MESSAGE$″$type$ = “$s : string$″/>

$< /s : sequence >$

$< /s : complexType >$

$<$ s : $elementname$ = “$ZIPCODEWORLD$US” nillable=“true” type=“tns:ZIPCODEWORLDUS” />

$< /s : schema >$

$< /wsdl : types >$

$< wsdl : messagename$ = “$ZIPCodeWorld$US SoapIn”¿

$< wsdl : partname$ = “$parameters$″$element$ = “$tns : ZIPCodeWorld$US” />

$< /wsdl : message >$

$< wsdl : messagename$ = “$ZIPCodeWorld$US SoapOut”>

$<$ $wsdl$: $partname$ = “$parameters$″$element$ = “tns : $ZIPCodeWorld$US Response” />

$< /wsdl : message >$

$< wsdl : messagename$ = “$ZIPCodeWorld$US HttpGetIn”>

$< wsdl : partname$ = “$ZIPCode$″$type$ = “$s : string$″/>

$< wsdl : partname$ = “$LICENSE$″$type$ = “$s : string$″/>

$< /wsdl : message >$

$< wsdl : messagename$ = “$ZIPCodeWorld$US HttpGetOut”>

$< wsdl : partname$ = “$Body$″$element$ = “$tns : ZIPCODEWORLD$US” />

$< /wsdl : message >$

$<wsdl:messagename = \text{“}ZIPCodeWorld\text{US HttpPostIn”}>$
$<wsdl:partname = \text{“}ZIPCode\text{″}type = \text{“}s:string\text{″}/>$
$<wsdl:partname = \text{“}LICENSE\text{″}type = \text{“}s:string\text{″}/>$
$</wsdl:message>$
$<wsdl:messagename = \text{“}ZIPCodeWorld\text{US HttpPostOut”}>$
$<wsdl:partname = \text{“}Body\text{″}element = \text{“}tns:ZIPCODEWORLD\text{US”} />$
$</wsdl:message>$
$<wsdl:portTypename = \text{“}ZIPCodeWorldUS\text{WebService Soap”}>$
$<wsdl:operationname = \text{“}ZIPCodeWorld\text{US”}>$
$<wsdl:inputmessage = \text{“}tns:ZIPCodeWorld\text{US SoapIn”} />$
$<wsdl:outputmessage = \text{“}tns:ZIPCodeWorld\text{US SoapOut”} />$
$</wsdl:operation>$
$</wsdl:portType>$
$<wsdl:portTypename = \text{“}ZIPCodeWorldUS\text{WebService HttpGet”}>$
$<wsdl:operationname = \text{“}ZIPCodeWorld\text{US”}>$
$<wsdl:inputmessage = \text{“}tns:ZIPCodeWorld\text{US HttpGetIn”} />$
$<wsdl:outputmessage = \text{“}tns:ZIPCodeWorld\text{US HttpGetOut”} />$
$</wsdl:operation>$
$</wsdl:portType>$
$<wsdl:portTypename = \text{“}ZIPCodeWorldUS\text{WebService HttpPost”}>$
$<wsdl:operationname = \text{“}ZIPCodeWorld\text{US”}>$
$<wsdl:inputmessage = \text{“}tns:ZIPCodeWorld\text{US HttpPostIn”} />$
$<wsdl:outputmessage = \text{“}tns:ZIPCodeWorld\text{US HttpPostOut”} />$
$</wsdl:operation>$
$</wsdl:portType>$
$<wsdl:bindingname = \text{“}ZIPCodeWorldUS\text{WebService Soap”}$ type=“tns:ZIPCodeWorldUSWebService Soap”>
$<soap:bindingtransport = \text{“}http://schemas.xmlsoap.org/soap/http\text{″}style = \text{“}document\text{″}/>$
$<wsdl:operationname = \text{“}ZIPCodeWorld\text{US”}>$
$<soap:operationsoapAction = \text{“}http://ws.fraudlabs.com/ZIPCodeWorld\text{US”}$ style=“document” />
$<wsdl:input>$
$<soap:bodyuse = \text{“}literal\text{″}/>$
$</wsdl:input>$
$<wsdl:output>$
$<soap:bodyuse = \text{“}literal\text{″}/>$
$</wsdl:output>$
$</wsdl:operation>$
$</wsdl:binding>$
$<wsdl:bindingname = \text{“}ZIPCodeWorldUS\text{WebServiceHttpGet”}$ type=“tns:ZIPCodeWorldUSWebService HttpGet”>
$<http:bindingverb = \text{“}GET\text{″}/>$
$<wsdl:operationname = \text{“}ZIPCodeWorld\text{US”}>$
$<http:operationlocation = \text{“}/ZIPCodeWorld\text{US”} />$

$< wsdl : input >$
$< http : urlEncoded/>$
$< /wsdl : input >$
$< wsdl : output >$
$< mime : mimeXmlpart =$ "$Body''/>$
$< /wsdl : output >$
$< /wsdl : operation >$
$< /wsdl : binding >$
$< wsdl : bindingname =$ "$ZIPCodeWorldUS$WebServiceHttpPost" type="tns:ZIPCodeWorldUSWebService HttpPost">
$< http : bindingverb =$ "$POST''/>$
$< wsdl : operationname =$ "$ZIPCodeWorld$US">
$< http : operationlocation =$ "$/ZIPCodeWorld$US" /¿
$< wsdl : input >$
$< mime : contenttype =$ "$application/x - www - form - urlencoded''/>$
$< /wsdl : input >$
$< wsdl : output >$
$< mime : mimeXmlpart =$ "$Body''/>$
$< /wsdl : output >$
$< /wsdl : operation >$
$< /wsdl : binding >$
$< wsdl : servicename =$ "$ZIPCodeWorldUS$WebService">
$< wsdl : portname =$ "$ZIPCodeWorldUS$WebServiceSoap" binding="tns:ZIPCodeWorldUSWebServiceSoap">
$< soap : addresslocation =$ "$http : //ws.fraudlabs.com/zipcodeworldUS$webservice.asmx" />
$< /wsdl : port >$
$< wsdl : portname =$ "$ZIPCodeWorldUS$WebServiceHttpGet" binding="tns:ZIPCodeWorldUSWebServiceHttpGet">
$< http : addresslocation =$ "$http : //ws.fraudlabs.com/zipcodeworldUS$webservice.asmx" />
$< /wsdl : port >$
$< wsdl : portname =$ "$ZIPCodeWorldUS$WebServiceHttpPost" binding="tns:ZIPCodeWorldUSWebService HttpPost">
$< http : addresslocation =$ "$http : //ws.fraudlabs.com/zipcodeworldUS$webservice.asmx" />
$< /wsdl : port >$
$< /wsdl : service >$
$< /wsdl : definitions >$