國立中山大學資訊管理學系

碩士論文

Department of Information Management

National Sun Yat-sen University

Master Thesis

以語意為基礎的網路服務找尋技術之研究

A Semantic-based Approach to Web Services Discovery

研究生：蔡郁槐

Yu-Huai Tsai

指導教授：黃三益 博士

Dr. San-Yih Hwang

中華民國 100 年 6 月

June 2011

國立中山大學研究生學位論文審定書

本校資訊管理學系碩士班

研究生蔡郁槐（學號：M974020011）所提論文

以語意為基礎的網路服務找尋技術之研究

A Semantic-based Approach to Web Services Discovery

經本委員會審查並舉行口試，符合碩士學位論文標準。

學位考試委員簽章：

吳三益	林福仁
楊昊秀	

指導教授(可免) _____

系主任/所長(可免) _____

致謝詞

很快的，學生生涯就要結束了。在中山資管的這些日子，從大學一路到研究所，學到了很多、成長了很多；對於這裡給我的一切，我充滿感激，無論是知識、人際關係、亦或是山海相依的美麗景色。

在這裡求學的期間，我最要感謝的就是我的指導教授，黃三益老師。除了在學術、技術上悉心指導，不厭其煩的和我反覆討論之外，我在老師身上學到更多的還有人生態度：對每一件事情專注且盡心盡力，每個細節都要仔細考慮，一定要做到最好的態度。也是因為老師鼓勵學生走向國際，我才更有自信實現國外求學的夢想，增廣了自己的視野。同時也要感謝所有中山資管的老師，在這個不同於其他系所的環境，我真的獲益良多。

我的家人，一直都是我最強而有力的後盾。從小到大，不管我想做甚麼，他們總是會和我討論、確定我的想法之後，就義無反顧的支持我，讓我可以沒有後顧之憂。即便是決定到歐洲讀書，我的家人也是只有支持，不讓我擔心任何事情。父母對我的教育，讓我學習如何思考、如何對自己負責，這些特質無論何時都是我對自己最自豪的地方。

感謝 LAB 所有成員，特別是一起努力的同學：老謝、泡泡、阿馬、甜字，共同努力的每一個日子，都值得珍惜;謝謝照顧以及指導我的學長姐：建祥、唐筠、昀志、文柏、鈴超，以及帶給我很多歡樂也幫我很多忙的學弟妹：HOT、阿 J、有盛、宜軒。還要感謝資管所 99 級所有的同學們，因為有你們，兩年的研究所生涯變得非常多采多姿。

另外要感謝出國期間一起奮鬥打拼，也一起旅行的同學 Kammy、Alec 和室友 Ethan、Patti、Thomas。有你們的陪伴，國外的生活不但不孤單，反而更加有趣。也要感謝 ESSCA 讓我有機會能到歐洲嘗試不同的學習模式。

中山資管所　99 級　蔡郁槐　2011.6.8

Abstract

Service-oriented Architecture is now an important issue when it comes to program development. However, there is not yet an efficient and effective way for developer to obtain appropriate component. Current researches mostly focus on either textual meaning or ontology relation of the services. In this research we propose a hybrid approach that integrates both types of information. It starts by defining important attributes and their weights for web service discovery using Multiple Criteria Decision Making. Then a method of similarity calculation based on both textual and ontological information is applied. In the experiment, we collect 103 real-world Web services, and the experimental results show that our approach generally performs better than the existing ones.

Keywords: Web service discovery, semantic Web, SOA, ontology, information retrieval

中文摘要

服務導向的軟體架構近年來在程式設計領域漸漸受到重視。然而，由於目前大多的研究都僅專注於比較 Web 服務的字面意義或語意關係的單一方法，還沒有具備效率和效能的方式來搜尋適當的 Web 服務。本文提出一個整合性的架構，首先根據多評準決策方法定義描述 Web 服務各個標籤的重要性；接著修正過去研究中的語意關係比較方法定義相似度的計算方式並加以整合。在實驗階段，本文中使用 103 筆真實資料，並發現本文中所提出的整合性方法整體而言比過去此領域中的研究方法能達到更好的成效。

關鍵字: web 服務搜尋，語意網，服務導向軟體架構，資訊檢索，web 服務表示法

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 - Introduction

1.1. Background

Service-oriented Architecture (SOA) has been widely used and discussed these years in both academia and industry. Its loose-coupled, reusable characteristics enable developers to combine units of functionality into applications to fulfill their needs. An SOA unit is realized on the web as a “service”, providing some specific functionality.

IBM web service tutorial explained web services as follows:
“Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. ... Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.”
It points out that web services are open, and not bound to specific organization or operating platform.

Nowadays, many industrial standards have been proposed for realizing SOA. For example, WSDL (Web Service Description Language) is a XML-based language for the description of a service, SOAP (Simple Object Access Protocol) is a XML-based message format for data transportation of services, mostly atop HTTP, and UDDI (Universal Description, Discovery and Integration) defines mechanism service registration and discovery (Fensel & Bussler, 2002).

Unfortunately, the UDDI supports only simple search and is now seldom used in public. As a result, lots of researches about web services discovery have been proposed. The aim of web services discovery is to identify a set of candidate web services that

meet a particular functional need. Note that non-functional parameters, usually referred to as quality of service (QoS), of a web service are not considered in this thesis. There are other researches that focus on selecting web services that satisfy QoS constraints and/or achieve some QoS goals. This area is usually called *web service selection*, which is beyond the scope of this thesis.

Many researches on web service discovery are based on text or syntactic methodology that decompose WSDL by linguistic approach and enable users to search for web services by giving some keywords, whereas others use ontology-based methods to measure the similarities between inputs and outputs of Web service operations.

1.2. Motivation

Although there are lots of researches about web services discovery, most of them propose approaches that syntactically search through WSDLs. They usually use a text-based searching approach; some of them additionally provide linguistic process finding out the semantic meanings of words to improve effectiveness, and some of them suggest generating index such as VSM (Vector Space Model) index and LSA (Latent Semantic Analysis) index to improve efficiency (Chen Wu & Chang, 2007)(Chen Wu, Chang, & Aitken, 2008). However, there are two main problems. The first problem is that not all information in WSDL is useful for searching, and there is still some helpful information beyond WSDL documents. Practical usage directions and other contextual information might be useful. For example, users may want to know about the provider, including their nationality and description (of themselves, services, and operations). Other users might have given some opinions or put tags on services, operations, and providers. Useless information should be eliminated and other advantageous information should be incorporated when searching web services.

The second problem is that text-based approach of searching views a WSDL document as a textual string and disregards its structure. However, the structure of a WSDL document may be helpful when comparing two services. Even though recent works using text-based approach already try to ensure that semantic meanings are under consideration, they still ignore the structure of web service.

More recently, semantic web has become an important issue, so there are also works using ontology-related methodology to deal with the matchmaking between operations. However, most of them define the similarity between operations by considering only inputs and outputs, ignoring other descriptive attributes such as descriptions and name.

In this thesis, we will utilize both the textual processing techniques and ontology-based methods for discovering web services that satisfy users' functional need. We propose several alternatives that utilize various types of information in different ways. These alternatives are evaluated using real world web services. We conclude that with our integrated methods, it is possible to achieve superior performance when compared to pure text-based and ontology-based methods.

1.3. Thesis Organization

The rest of this paper is organized as follows. Chapter 2 reviews previous works in web service discovery and representation. Our proposed approach is introduced in Chapter 3, followed by Chapter 4 which describes the evaluation experiments. Finally, Chapter 5 summarizes the thesis and gives directions of future works.

CHAPTER 2 - Literature Review

In this chapter, we will introduce some basic concepts about web services. Then web service representations will be described. In the end, we will present several previous researches about web service discovery.

2.1. Web Service Technology

According to the W3Cschools.com, the basic Web services platform is XML + HTTP, and web services platform elements mainly include:

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

SOAP (Simple Object Access Protocol) is a W3C recommendation which W3Cschools.com explained as follows:" SOAP is a simple XML-based protocol to let applications exchange information over HTTP. Or more simply: SOAP is a protocol for accessing a Web Service."

It is a message lay out which standardizes the way of transporting XML-encoded data. As a communication protocol via internet, it provides a way to pass messages between endpoints binding to HTTP.

Even though there are several techniques like DCOM, RMI and CORBA that work on the local area network, they fail to apply to the Internet environment. Such techniques tightly couple components and conflict with existing firewall technology. Instead, SOAP provides an RPC-like and simple mechanism for communication (Mitra

& Lafon, 2007)(Fensel & Bussler, 2002).

The WSDL (Web Services Description Language) is an XML-based language documented by the W3C and defines web services and their interfaces. Elements in the definition of network services are as follows:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

Abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings in WSDL, allowing the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. Furthermore, a common binding mechanism is defined in WSDL and used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions (Christensen, Curbera, Meredith, & Weerawarana, 2001)(Fensel & Bussler, 2002).

The UDDI (Universal Description, Discovery and Integration) is a registry which provides a mechanism for clients to find web services. Business can search for services

provided by service provider or external partners with UDDI interface. Both web services providers (such as businesses which want to publish service descriptions and usage interfaces) and users (those who look for web services to satisfy their programming needs) are clients of UDDI registry (Clement, Hately, & Riegen, 2004)(Fensel & Bussler, 2002).

However, the public UDDI registry has been shut down permanently since January 12, 2006, which means the most important web services discovery mechanism had been missing from the web services community. Yet for the characteristics of SOA and the open, loosely-coupled, demand-driven business environment, a dedicated public Web service registry is still indispensable. (Chen Wu & Chang, 2007)

2.2. Web Service Representation

WSDL can be used to describe the information about a web service and enables users to know what it can do and how it is used, but when it comes to searching, there are some information that may be important to users yet not included in the current WSDL specification.

2.2.1. Information of Web Service

QoS for web services has become a popular issue over the SOA field. According to W3C document “QoS for Web Services: Requirements and Possible Approaches, ” (Lee, Jeon, & Park, 2003) the commonly used QoS requirements for web services are as follows:

- Performance
- Reliability

- Scalability
- Capacity
- Robustness
- Exception Handling
- Accuracy
- Integrity
- Accessibility
- Availability
- Interoperability
- Security
- Network-Related QoS Requirements

QoS enables users to evaluate a web service before using it, providing a safer and more efficient way to deal with web services.

In addition, after exploring some web service market places and search engines such as seekda.com, we realize that there is still some information which can enable users to understand web services better and find what they need, including:

- Service providers' information
- Service descriptions by providers
- Service description by users
- Operation description by providers
- Tags and categories
- Other QoS aspect

With such information, we can further improve the searching methods.

2.2.2. Ontology-based Web Service Organization

Works about semantic web are thriving these years. It is predictable that we'll be able to access web resources by content instead of just keywords. The OWL-S describes lots of information including functional and non-functional ones. The main goals of representing web services are to enable the design of the following techniques:

- Automatic Web service discovery.
- Automatic Web service invocation.
- Automatic Web service composition and interoperation.

Structuring an ontology of services can help users understand and answer the following questions:

- What does the service provide for prospective clients?
- How is it used?
- How does one interact with it?

In other words, we can better understand how to find and interact with a web service when it is described by a well-formed ontology. When describing a web service, three main parts should be covered:

- Service Profile
- Service Grounding

- Service Model

These three parts aim to provide information to answer the previous three questions and more completely record the service.(Martin et al., 2004)

To illustrate an ontology, OWL-S provides an object-oriented approach to describe web services in terms of classes, properties, and axioms. It is built on earlier web ontology standards such as RDF and RDF Schema and extends those languages with richer modeling primitives. We can use a schematic diagram to illustrate web services. For example, with a model of ontology using a directed graph that the nodes represent “the concepts in WSDL” and the edges represent the relationship of “has” between two concepts, we can easily understand the content and structure of a service. (Medjahed, Bouguettaya, & Elmagarmid, 2003)(Gao, Sperberg-McQueen, & Thompson, 2009)(Peterson, Gao, Malhotra, Sperberg-McQueen, & Thompson, 2009)

2.3. Web Service Discovery

Now that UDDI is no longer operational in public, but the issue of how web services users find what they need remains important. Along with the syntactic way that searches for web services with traditional, text-based techniques, we can also incorporate ontology concepts for search and matchmaking (Umesh Bellur & Kulkarni, 2007).

2.3.1. text-based web services searching

We will introduce a WSDL-based Web service discovery approach, aiming to provide a simple yet effective Web service discovery mechanism that can retrieve

relevant Web services from the Internet (…). The main steps to search for web services are as follows:

1. Web services crawling and preprocessing
2. Web services processing
    i. Content Extraction
    ii. Fragment Expansion
    iii. Linguistic Analysis
3. Vector Space Model Indexing
4. Latent Semantic Analysis Indexing

These steps can help finding web services over WSDLs efficiently including considerations of tokenization, semantic meanings of words, weight of words. In addition, Hierarchical Agglomerative Clustering is provided as extra information. (Chen Wu & Chang, 2007)(Chen Wu, Chang, & Aitken, 2008)

2.3.2. ontology-based web services matchmaking

Semantic web has become asubject active research area in recent years, attracting more and more researchers to work on the related issues. Regarding the input of an operation, it would be acceptable in some level if the candidate operation's input represents a more general concept than the request's input; as for the output of an operation, it's acceptable in some level if the request's output subsumes the candidate operation's output. With this principle, services that are usable and meet the needs can be found.(Paolucci, Kawamura, Payne, & Sycara, 2002) (Jaeger, Rojec-Goldmann, Liebetruth, Mühl, & Geihs, 2005)

In addition to the hierarchy of ontology, there are more punctilious methods when matching two concepts. Three main kinds of methods will be introduced here (Pedersen,

Patwardhan, & Michelizzi, 2004)(Resnik, 1999). The first one is to calculate the length of path between concepts, using it to determine their similarity. The second one is mainly related to the concepts' depth in the ontology. This method refers to the previous work, *Wu & Palmer measure (WUP), defining the similarity between two concepts as follows:*

$$\frac{2 \times \text{depth}(\text{least common ancestor of concept1 and concpet2})}{\text{depth}(\text{concept1}) + \text{depth}(\text{concept2})}$$

The third one uses the Information Content to define the similarity. This method refers to another previous work, *Lin measure (LIN), defining the similarity between two concepts as follows:*

$$\frac{2 \times \text{IC}(\text{least common ancestor of concept1 and concpet2})}{\text{IC}(\text{concept1}) + \text{IC}(\text{concept2})}$$

To illustrate, we cite an instance of a part of ontology from Wordnet, describing a relationship between vehicles as shown in Figure 2.1.

The Information Content of a concept IC (*concept*) = $-\log \text{p}(concept)$, where p(*concept*) indicates the probability of encountering an instance of the concept from root by following random walk. In the example, "car#n#1" and "truck#n#1" have the same depth of 12 but different IC, and both of them can be used to compute the similarity between concepts. More detailed description and example will be given in the following chapter.

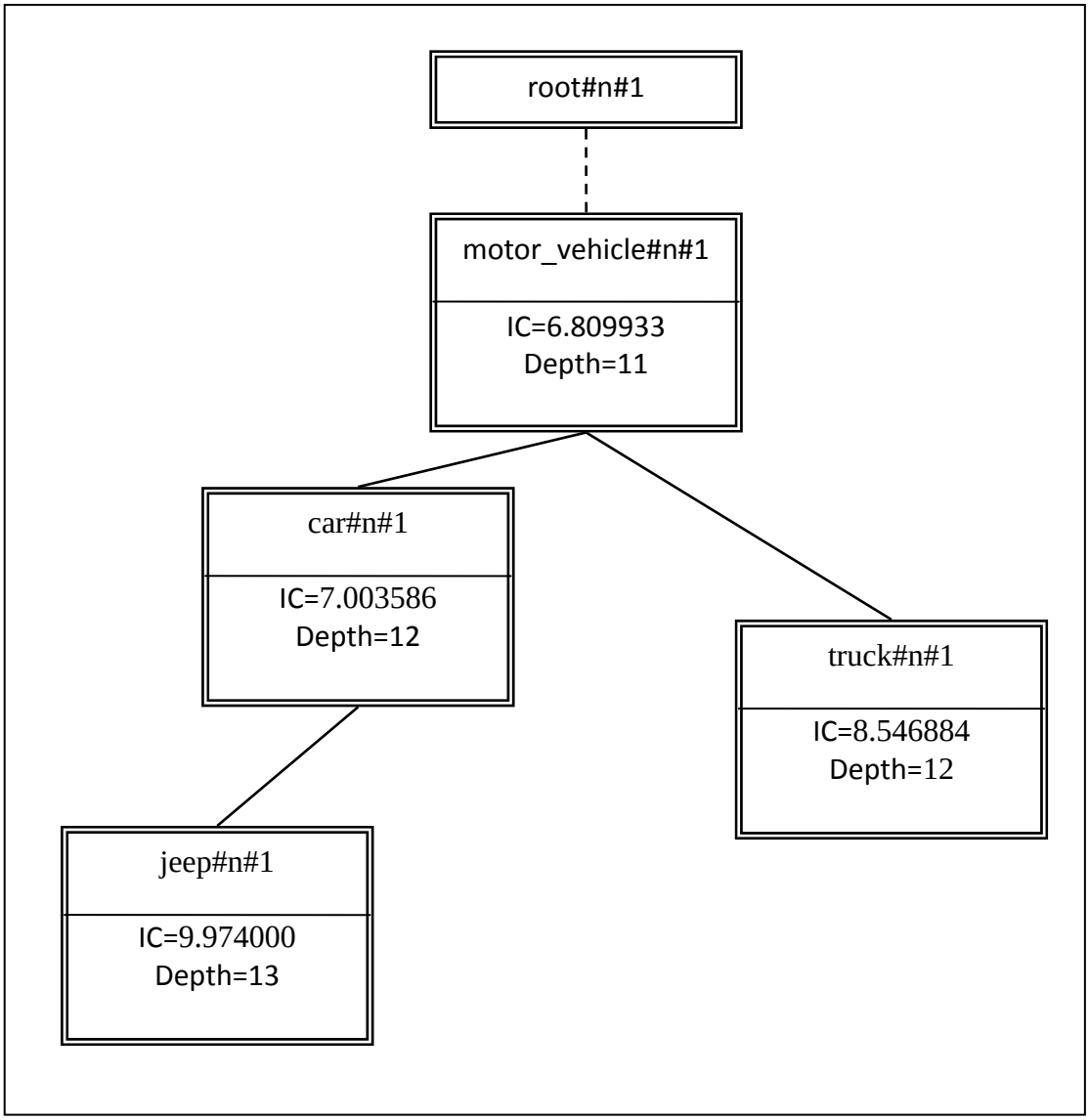

Figure 2.1: Partial ontology of Wordnet, where solid lines represent superclass/subclass relations between concepts and dashed line indicates some omitting of intervening concepts

2.3.3. Analytic hierarchy process

When computing overall score of similarity between web services, we have to consider multiple attributes pertaining to them. There are many ways to decide the weights of attributes, including equal weighting and diversified weighting derived from

survey data. However, equal weighting scheme ignores the fact that some attributes are indeed more important than the other. With questionnaires filled by users or experts, trustworthy weight can be obtained, but consistency problem still need to be solved when attributes are compared in pair.

Analytic hierarchy process (AHP) is the most commonly used method in Multiple Criteria Decision Making (MCDM). It is a four-step decision making process:

1. Define the problem and determine the kind of knowledge sought.
2. Structure the decision hierarchy.
3. Construct a set of pairwise comparison matrices.
4. Use the priorities obtained from the comparisons to weigh the priorities in the level immediately below.

These steps provide an objective way to define the weight of each criterion with consistency checking. For example, if we want to determine weight of three attributes, *a*, *b*, and *c*, and the questionnaire result gives that

- IM_a:IM_b = 1:2 (equation 1),
- IM_b:IM_c = 1:2 (equation 2), and
- IM_a:IM_c = 1:3 (equation 3),

where IM_i:IM_j represents the relative importance of attribute *i* and *j* that user indicate in the questionnaire.

According to equation 1 and equation 2, IM_a:IM_c should be 1:4, where

questionnaire result gives that IM_a:IM_c = 1:3, making us unable to decide the true weights. This is what we call the consistency problem. AHP provides a method that solves this problem with a series of mathematic procedure. Table 2.1 illustrates the overall concept of the method. First of all, each value is divided by the biggest value of each column. Then, compute the mean value of each row. In the end, normalize the values computed in second step (such that the sum of all reliable weights is 1).

After the procedure, we can than conclude that w_a:w_b:w_c = 11:20:36, which successfully obtain the consistency and provide trustworthy weights. More detailed description and example will be given in the following chapter (Saaty, 2008).

Table 2.1 consistency solving

	a	b	c	average
a	$\frac{1}{3}$	$\frac{\frac{1}{2}}{2}$	$\frac{\frac{1}{3}}{1}$	$\frac{11}{36}$
b	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{\frac{1}{2}}{1}$	$\frac{5}{9}$
c	$\frac{3}{3}$	$\frac{2}{2}$	$\frac{1}{1}$	1

CHAPTER 3 – Our Approach

In this chapter, we propose an integrated approach to handling web service discovery problem. Our approach starts with the representation of a web service, followed by defining the similarity of web services.

3.1. Organizing a web service with a class diagram

Figure 3.1 shows the structure of a web service description using a class diagram.

After browsing some existing web service search engines such as seekda.com, we identify the information that are important in web service discovery.

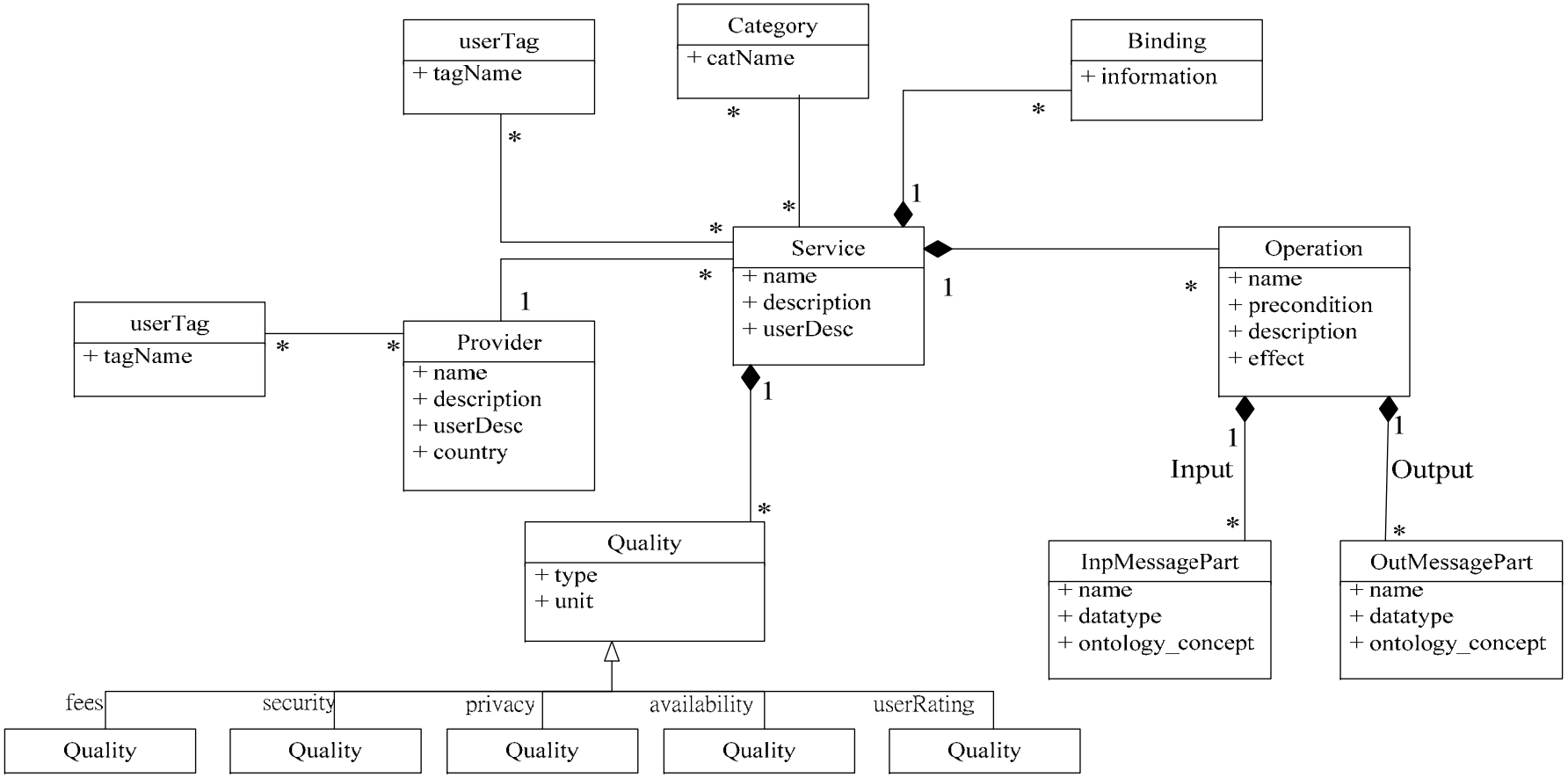

Figure 3.1: structure of web service description in class diagram

Figure 3.2 shows the corresponding XML schema.

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="service" type="Service"/>
 <xs:complexType name="Service">
  <xs:sequence>
   <xs:element name="Name" type="xs:string" minOccurs="1" maxOccurs="1" />
   <xs:element name="binding" type="xs:string" minOccurs="1" maxOccurs="unbounded" />
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="provider" type="Provider" minOccurs="1" maxOccurs="1" />
   <xs:element name="quality" type="Quality" minOccurs="1" maxOccurs="1" />
   <xs:element name="userDesc" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="userTag" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
   <xs:element name="operation" type="Operation" minOccurs="1" maxOccurs="unbounded" />
   <xs:element name="category" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="Provider">
  <xs:sequence>
   <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
   <xs:element name="link" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1" />
   <xs:element name="userDesc" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="userTag" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
   <xs:element name="country" type="xs:string" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
 </xs:complexType>
<xs:complexType name="Qos">
  <xs:sequence>
   <xs:element name="fees" type="xs:decimal" minOccurs="1" maxOccurs="1" />
   <xs:element name="security" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="privacy" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="availability" type="xs:string" minOccurs="1" maxOccurs="1" />
   <xs:element name="userRating" type="xs:integer" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
 </xs:complexType>
<xs:complexType name="Operation">
  <xs:sequence>
   <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
   <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="precondition" type="xs:string" minOccurs="0" maxOccurs="1" />
   <xs:element name="effect" type="xs:string" minOccurs="0" maxOccurs="1" />
```

```
    <xs:element name="input" type="Input" minOccurs="0" maxOccurs="1" />
    <xs:element name="output" type="Output" minOccurs="0" maxOccurs="1" />
   </xs:sequence>
  </xs:complexType>
<xs:complexType name="Input">
    <xs:element name="parameter" type="Parameter" minOccurs="1" maxOccurs="unbounded" />
</xs:complexType>
  <xs:complexType name="Output">
    <xs:element name="parameter" type="Parameter" minOccurs="1" maxOccurs="unbounded" />
  </xs:complexType>
  <xs:complexType name="Parameter">
   <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
    <xs:element name="datatype" type="xs:string" minOccurs="1" maxOccurs="1" />
   </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Figure 3.2 Schema of the web service description

3.2. Web service similarity

In order to provide a method for users to find web services fulfilling their requirements, we first have to define the similarity between web services based on their attributes described in Figure 3.1. Note that QoS attributes are excluded from our subsequent discussion as our focus in this thesis is on functional aspects of web services. The other attributes are categorized into two types, namely text-based and ontology-based. Then we will describe a method for computing the weights of attributes. Finally, the similarity between two web services is the weighted sum of the similarities of their respective attribute values.

3.2.1. Ontology-based attribute similarity

For operation inputs/outputs attributes, their values are mapped to concepts of an

ontology, by which we use the Wordnet ontology in our experiments.

To compute attribute similarities, we map an attribute value in Figure 3.1 to a concept in Wordnet ontology, which is then used for our similarity determination (Miller et al., 2010). Taxonomy and meaning of the concept should both be under consideration in the mapping procedure. For example, if we have an attribute of "car", which means a four-wheel auto vehicle that man can drive for transportation, represented by "car#n#1". Its definition in Wordnet ontology is "*a motor vehicle with four wheels; usually propelled by an internal combustion engine*", and the trace from root node to it is as follows:

Root*#n#1, entity#n#1, physical_entity#n#1, object#n#1, whole#n#2, artifact#n#1, instrumentality#n#3, conveyance#n#3, vehicle#n#1, wheeled_vehicle#n#1, self-propelled_vehicle#n#1, motor_vehicle#n#1, car#n#1

Figure 3.3 trace from "Root*#n#1" to "car#n#1"

After attributes are mapped to concepts, we adapt the methods described in Section 3.2 of Chapter 2 to compute the similarity between concepts.

Table 3.1 Scores set for each condition in method of operation ontology

Grade	Score
Exact	1
Subsume	0.5
Plug-in	0.5
failed	0

The first method is based on super/subclass relationship between concepts, and its pseudo code is shown in Figure 3.4. It compares the input and output of a pair of operations and returns a similarity score of the two operations. This method is originally proposed by (Paolucci et al., 2002). In the original method, similarities are defined as exact, subsume, plug-in, or failed. For each condition, we give it a score in order to compute the overall similarity. The corresponding score of each grade is shown in table 3.1.

For example, the similarity between “jeep#n#1” and “truck#n#1” is defined as 0, because they do not have subsume, subclass, or superclass relation.

```
SimilarityOfAttributes(I_Q, I_C) {
I_Q = an input part of an operation from the query service
I_C = an input part of an operation from the candidate service
IC_Q = the corresponding concept of I_Q,
IC_C = the corresponding concept of I_c
      If the data type of I_Q and I_C are not equal
            then return 0;
      else {
      If IC_Q is equal to IC_C
            then return 1
      elseif IC_c subsumes IC_Q (IC_c is at a higher level then IC_Q in the ontology)
            then return 0.5;
      else        return 0;
      }
}

SimilarityOfAttribute(O_Q, O_C) {
O_Q = an output part of a operation from the query service
O_C = an output part of a operation from the candidate service
OC_Q = the corresponding concept of O_Q
OC_C = the corresponding concept of O_c
      If the data type of O_Q and O_C are not equal
            then return 0;
      else {
      if OC_Q is equal to OC_C
            then return 1;
      elseif OC_Q subsumes OC_C (OC_Q is at a higher level than OC_C in ontology)
            then return 0.5;
      elseif OC_C subsumes OC_Q (OC_c is at a higher level than OC_Q in ontology)
            then return 0.5;
      else        return 0;
      }
}
```

Figure 3.4 Similarity measures for input/output attributes of web service operations

The second method refers to the previous work, *Wu & Palmer measure (WUP)*, which counts the similarity based on paths between concepts. The formula is as follows:

$$\text{SimilarityOfAttribute} = \frac{2 \times \text{depth(least common ancestor of concept1 and concpet2)}}{\text{depth(concept1)} + \text{depth(concept2)}}$$

where depth(*concept*) is the path length from root node to *concept* in the ontology.

The least common ancestor of “jeep#n#1” and “truck#n#1” in the example ontology is “motor_vehicle#n#1”. Therefore the score is:

$$\text{SimilarityOfAttribute} = \frac{2 \times \text{depth(motor\_vehicle\#n\#1)}}{\text{depth(jeep\#n\#1)} + \text{depth(truck\#n\#1)}} = \frac{2 \times 11}{13 + 12} = 0.88$$

The third method refers to another previous work, *Lin measure (LIN)*, which counts the similarity based on Information Content (IC) of concepts. The score function is:

$$\text{SimilarityOfAttribute} = \frac{2 \times \text{IC(least common ancestor of concept1 and concpet2)}}{\text{IC(concept1)} + \text{IC(concept2)}}$$

When computing the similarity between “jeep#n#1” and “truck#n#1” using this method, the score is:

$$\text{SimilarityOfAttribute} = \frac{2 \times \text{IC(motor\_vehicle\#n\#1)}}{\text{IC(jeep\#n\#1)} + \text{IC(truck\#n\#1)}} = \frac{2 \times 6.809933}{9.974000 + 8.546884} = 0.7354$$

3.2.2. Text-based attribute similarity

For other attributes such as service name, description, tags, and operation name, there are no associated ontology concepts. Hence, text-based methodology can be used here to determine the similarity.

Two words, w1 and w2, are compared lexically using the following method:

$$\text{Text\_Similarity} = \frac{2 \times \text{Length}(\text{MaxCommonString}(w1, w2))}{\text{Length}(w1) + \text{Length}(w2)}$$

,where MaxCommonString is the longest continuous characters in common.

To compare attributes of multiple words, we regard each attribute value as a paragraph. The method to compare two paragraphs is similar to the word comparison by using words (rather than alphabets) as the basic units. Specifically, the similarity of two paragraphs, p1 and p2 is defined as follows:

$$\text{Paragraph\_Similarity} = \frac{2 \times \text{WordCount}(p1 \cap p2)}{\text{WordCount}(p1) + \text{WordCount}(p2)}$$

,where p1∩p2 represents the set of words appear in both paragraphs p1 and p2, and WordCount(p1) returns the number of words in paragraph p1.

3.2.3. Weight Determination

To determine the similarity between two Web services, each involving multiple attributes, we have to determine the weight of each attribute. To this end, we use the methodology of the first step of AHP (Analytical Hierarchy Process), the most commonly used method of MCDM (Multiple Criteria Decision Making), to ensure

consistency of the weights.

First of all, we design a questionnaire that enables us to understand the relative importance between attributes. The questionnaire design is based on the “Pairwise comparison scale for AHP preferences” which describes the preference of one attribute over another, see Table 3.2.

Table 3.2 Pairwise comparison scale for AHP preferences

Pairwise comparison scale for AHP preferences	
Level of preference	score
Extremely preferred	9
Very strongly to extremely	8
Very strongly preferred	7
Strongly to very strongly	6
Strongly preferred	5
Moderately to strongly	4
Moderately to preferred	3
Equally to moderately	2
Equally preferred	1

After collecting the answer of questionnaire given to a domain expert, the result is represented as a matrix R:

$$R = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

,where a_{xy} represents the level of preference of x^{th} attribute to y^{th} attribute. To ensure consistency, the following step will be done:

1. Each value is divided by the biggest value of each column x_i

$$R' = \begin{bmatrix} \frac{a_{1,1}}{x_1} & \frac{a_{1,2}}{x_2} & \cdots & \frac{a_{1,n}}{x_n} \\ \frac{a_{2,1}}{x_1} & \frac{a_{2,2}}{x_2} & \cdots & \frac{a_{2,n}}{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n,1}}{x_1} & \frac{a_{n,2}}{x_2} & \cdots & \frac{a_{n,n}}{x_n} \end{bmatrix}$$

2. Compute the mean value of each row

$$a_i = \frac{a_{i,1} + a_{i,2} + \cdots + a_{i,n}}{n}$$

3. The reliable weight of attribute *i*, w_i, is obtained by normalizing the values computed in second step (such that the sum of all reliable weights is 1), as shown below:

$$w_i = \frac{a_i}{a_1 + a_2 + \cdots + a_n}$$

3.2.4. Operation and Service similarity

After determining similarity of attributes in pairwise, we will introduce how we sum them up. We have to compute the similarity between operations first. The algorithm is shown in Figure 3.5. Note that the average of the highest similarity for each candidate's input and request's output will be defined as the overall similarity of operation inputs and outputs. With this setting, we can ensure that the service we found is usable by us and best fulfill our needs.

```
SimilarityOfOperations(OP_Q, OP_C) {
OP_Q=an operation of the query service
OPc=an operation of the candidate service
I_Q = an input part of OP_Q
I_C = an input part of OPc
O_Q = an output part of the operation from the query service
O_C = an output part of the operation from the candidate service
w_i=the weight of input
w_o=the weight of output
w_opn=the weight of operation name
w_opd=the weight of operation description

    For each I_C {
        score(I_C) = MAX_{I_Q}(SimilarityOfAttribute (I_Q, I_C) )
    }
    score(Input)= AVG ( score(I_C) )
    For each O_Q {
        score(O_Q) = MAX_{O_C} (SimilarityOfAttribute (O_Q, O_C) )
    }
    score(Output)= AVG ( score(O_Q) )

    OperationScore= score(Input)* w_i + score(Output)* w_o +
        score(OperationName)* w_opn+score(OperationDescription)* w_opd
    return OperationScore
}
```

Figure 3.5 algorithm of operation similarity determination

Finally, we can compute the overall service similarity with all the scores and weight. The service-level algorithm is shown in Figure 3.6. We use the highest similarity of operations and sum it up with other attributes. In our approach, the highest operation matching score will be chosen.

```
SimilarityOfServices(S_Q, S_C) {
S_Q= query service
Sc=candidate service
OP_Q=an operation of the query service
OPc=an operation of the candidate service
w_sn=the weight of service name
w_sdu=the weight of service description by user
w_sdp=the weight of service description by provider
w_ts=the weight of user tag on service
w_tp=the weight of user tag on provider
      for each operation OP_Q in S_Q
      {
            score(OP_Q) =  AVG_OPC  ( SimilarityOfOperations(OP_Q, OP_C) )
      }
      OP_Similarity=AVG_OPQ  (score(OP_Q))
      ServiceSimilarity = Text_similarity (S_Q.ServiceName, S_C.ServiceName)* w_sn+
      Paragraph_Similarity (S_Q.ServiceDescriptionByUser,
      S_C.ServiceDescriptionByUser)* w_sdu+
      Paragraph_Similarity (S_Q.ServiceDescriptionByProvider,
      S_C.ServiceDescriptionByProvider)* w_sdp+
      Text_similarity (S_Q.UserTagOnService, S_C.UserTagOnService)* w_ts +
      Text_similarity (S_Q.UserTagOnProvider, S_C.UserTagOnProvider)* w_tp +
      OP_Similarity

      return ServiceSimilarity;
}
```

Figure 3.6 algorithm of service similarity determination

CHAPTER 4 - Evaluation

To evaluate our approaches, we've executed an experience comparing our proposed method with previous work. In the experiment, we provide multiple steps to illustrate the effects of each factor.

4.1. Experimental Design

Our experiment will compare six different methods with two levels of accuracy, ensuring our approaches can make a significant improvement in web service discovery.

4.1.1. Data Source

As experiment sample, we've collected 103 services mainly from seekda.com, the most popular and biggest web services search engine, and some from goole.com. These services are searched by giving a keyword "translation". Some of them are related to natural language translation, programming language translation, biological data translation, and other translation. Some of them are not actually related to translation, they were searched only because there are words like "translation" in their WSDLs. Here we can find out that using keyword searching can be questionable.

However, most of the existing services do not provide ontology-mapping information since ontology-based searching is not yet in common use. As to verify our approaches, we add the ontology mapping information for all the inputs/outputs attributes, making the sample data more complete with all the information for the discovery procedure. The ontology we adopt is the Wordnet ontology, the most

abundant and common-used ontology in recent years. Wordnet provides sufficient concepts to map to, and is used in various applications that may be helpful.

To compare performance of each method, we give a request as follows:

Table 4.1 Request web service for experiment

Attribute	request
Service name	translationService
Service description by user	take multiple languages text as input and translate into other languages as output
Service description by provider	translate words or articles between some languages
User tag on service	Interpretation , dictionary
User tag on provider	language , translation
Operation name	translate
Operation description	do the translation method , translate the input into another language
input	languageMode / nation#n#1 textContent / natural_language#n#1
output	translateResult / natural_language#n#1

Among the 103 candidates, we first identify 22 of them as “total”, which means they are functionally the same as the request or cover all functionalities in the request. 16 of them are then discovered as “partial”, which represents that they are quite similar

but not totally meeting the needs.

4.1.2. Performance Metrics

The method to be compared in the experiment will be introduced as follows:

1. WSDL matchmaking:

   This method compares two WSDL as a non-structured textual document, using the paragraph similarity method to obtain the result.

2. Text matchmaking:

   This method compares two services column-to-column, using word similarity or paragraph similarity for each matchmaking according to the natures of the attributes. If multiple tags exist, the average of scores is taken here. If the service provides more than one operations, the highest score will be adopted.

3. Operation ontology matchmaking:

   This method compares services considering only operation inputs/outputs attributes. Inputs/outputs will be compared pairwisely, and the average score of each attributes' highest matchmaking will be seen as the operation's matchmaking score. Also, if the service provides more than one operations, the highest score will be adopted.

4. Integrated method with concept relationship:

   As we mentioned in previous chapter, the inputs/outputs attributes will be judge as "exact", "subsume or plug-in", "fail" comparing to the request. Other attributes will be scored using text-based method. Further, weight will be adapted to each attribute to compute the overall score.

5. Integrated method with LIN:

The method we proposed in previous chapter, inputs/outputs attributes matchmaking will be done using the LIN measure, considering the depth of the concept that each attribute mapped to. Also, other attributes will be scored using text-based method, and weight will be adapted to each attribute to compute the overall score.

6. Integrated method with WUP:

   The method we proposed in previous chapter, inputs/outputs attributes matchmaking will be done using the WUP measure, considering the Information Content (IC) of the concept that each attribute mapped to. Also, other attributes will be scored using text-based method, and weight will be adapted to each attribute to compute the overall score.

For each method, we test it by computing the precision and recall for “total” or “total” and “partial”. The definitions are as follows:

$$\text{Precision for "total"} = \frac{\text{numbers of "total" in the top 22}}{22}$$

$$\text{Precision for "total" or "partial"} = \frac{\text{numbers of "total" or "partial" in the top 38}}{38}$$

$$\text{Recall for "total"} = \frac{22}{\text{numbers to cover all 22}}$$

$$\text{Recall for "total" or "partial"} = \frac{38}{\text{numbers to cover all 38}}$$

4.1.3. Implementation

In this section, we will introduce the implementation of our approach and the tools we use. Our tools mainly cover two parts: ontology-based matchmaking and text-based matchmaking.

Since our inputs/outputs parameters are mapped to the ontology from Wordnet, we seek for applications using the Wordnet. *Wordnet::similarity* is a Perl module for computing measures of semantic relatedness, containing different kinds of methods including LIN and WUP we used here, provided by Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi and Satanjeev Banerjee. Wordnet::similarity is commonly used when it come to Wordnet and semantic web because of its completeness and ease of use. We run the *Wordnet::similarity 2.05* with *Perl 5.8.9* and compute scores for inputs/outputs matchmaking with it.

For paragraph matchmaking, we use the *Text-Similarity*, another Perl function provided by the same authors of *Wordnet::similarity*. It computes similarity between two paragraphs or files, giving scores of various measures such as Precision, Recall, Cosine, and also the F-measure we've mentioned in previous work. We utilize Text-Similarity 0.07 for implementing our approach.

Similar_text is a PHP function that computing the F-measure similarity of two strings in percentage. It can be used to compare two single words simply. We run the *Similar_text* function under *PHP 5.2.9* to realize our approach.

4.1.4. Parameter settings

In previous chapter, we've introduced the procedure to determine the weight of each attributes for integrated methods. After deriving a questionnaire from a high-level engineer from IBM, who is well-experience of web service development and usage, we obtain the raw matrix as follows, where attributes are in order of service name, service

description by user, service description by provider, user tag on service, user tag on provider, operation name, operation description, input, and output:

$$R = \begin{bmatrix} 1 & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} \\ 5 & 1 & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} \\ 5 & 5 & 1 & \frac{1}{8} & \frac{1}{8} & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{5} \\ 8 & 8 & 8 & 1 & \frac{1}{8} & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} \\ 8 & 8 & 8 & 8 & 1 & \frac{1}{5} & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} \\ 5 & 5 & 5 & 5 & 5 & 1 & \frac{1}{5} & \frac{1}{8} & \frac{1}{8} \\ 5 & 5 & 5 & 5 & 5 & 5 & 1 & \frac{1}{5} & \frac{1}{5} \\ 8 & 8 & 8 & 8 & 8 & 8 & 5 & 1 & \frac{1}{8} \\ 8 & 8 & 5 & 8 & 8 & 8 & 5 & 8 & 1 \end{bmatrix}$$

Then each value is divided by the biggest value of each column to get R':

$$R' = \begin{bmatrix} \frac{1}{8} & \frac{1}{40} & \frac{1}{40} & \frac{1}{64} & \frac{1}{64} & \frac{1}{40} & \frac{1}{25} & \frac{1}{64} & \frac{1}{8} \\ \frac{5}{8} & \frac{1}{8} & \frac{1}{40} & \frac{1}{64} & \frac{1}{64} & \frac{1}{40} & \frac{1}{25} & \frac{1}{64} & \frac{1}{8} \\ \frac{5}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{64} & \frac{1}{64} & \frac{1}{40} & \frac{1}{25} & \frac{1}{64} & \frac{1}{5} \\ \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{1}{8} & \frac{1}{64} & \frac{1}{40} & \frac{1}{25} & \frac{1}{64} & \frac{1}{8} \\ \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{1}{8} & \frac{1}{40} & \frac{1}{25} & \frac{1}{64} & \frac{1}{8} \\ \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{25} & \frac{1}{64} & \frac{1}{8} \\ \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{5}{8} & \frac{1}{5} & \frac{1}{40} & \frac{1}{5} \\ \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{5}{5} & \frac{1}{8} & \frac{1}{8} \\ \frac{8}{8} & \frac{8}{8} & \frac{5}{8} & \frac{8}{8} & \frac{8}{8} & \frac{8}{8} & \frac{5}{5} & \frac{8}{8} & 1 \end{bmatrix}$$

Then compute the mean value of each row to get matrix a:

$$[a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9]$$
$$= [0.0458 \quad 0.1124 \quad 0.1874 \quad 0.3718 \quad 0.4812 \quad 0.3812 \quad 0.4639 \quad 0.8056 \quad 0.9583]$$

Finally, normalize values making the sum of all values is 1.

$$[w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad w_9]$$
$$= [0.0120 \quad 0.0295 \quad 0.0492 \quad 0.0976 \quad 0.1264 \quad 0.1001 \quad 0.1218 \quad 0.2116 \quad 0.2517]$$

where w_1, w_2,…w_9 are the weights for our attributes.

4.2. Experimental Result

After all the tools and environment being prepared, we can compute a score for each service with different method, and sort them from the highest score to the lowest score. To specifically indicate the difference between methods, we proceed cross validations and generate figures of comparison in four dimensions.

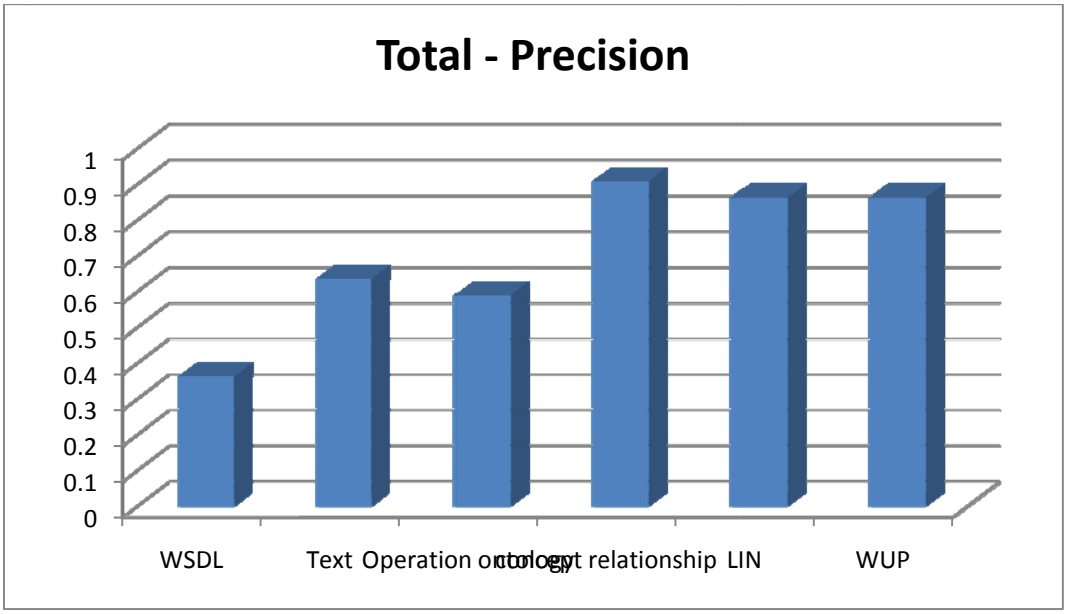

Figure 4.1 Precision for total matchmaking

In Figure 4.1, we can see that on the precision for total, approaches we proposed perform better than other methods. Performances of the three methods we proposed are approximately equal. In this dimension, score of operation ontology method is even lower than which of text-based method.

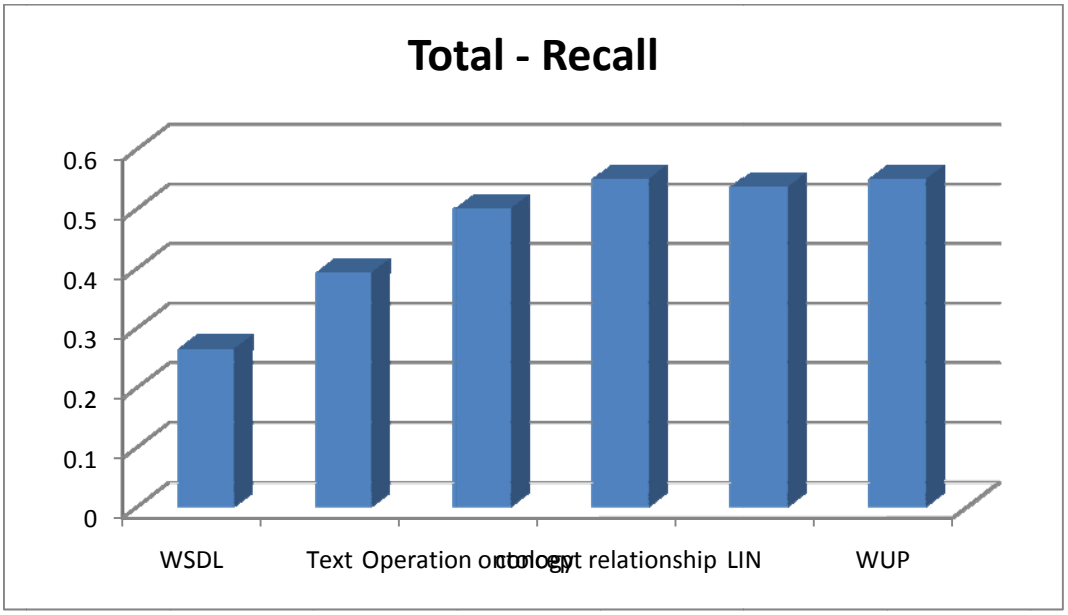

Figure 4.2 Recall for total matchmaking

In Figure 4.2, the performances of our approaches are obviously greater than others on the recall for total. Method of operation ontology performs better in this dimension but still worse than what we proposed.

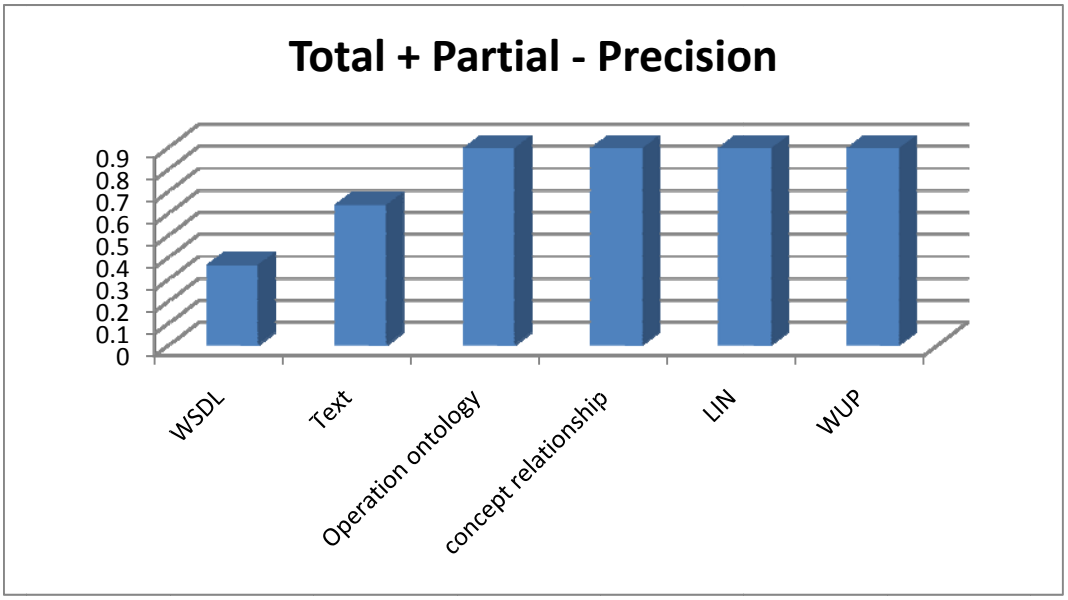

Figure 4.3 Precision for total or partial matchmaking

In Figure 4.3, although the method of operation ontology performs as well as our approaches, ours' scores are still respectively higher than the others'.

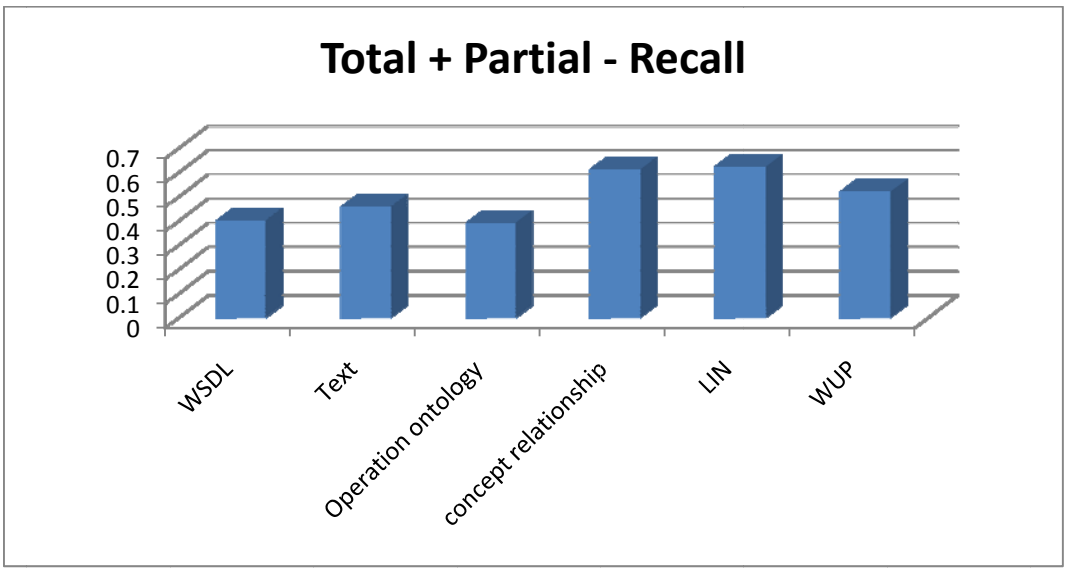

Figure 4.4 Recall for total or partial matchmaking

In Figure 4.4, even if the integrated WUP method is a little bit worse than another two methods we give, three of them are all better than the ones in previous works. And again, score of operation ontology method is even lower than which of text-based method, even lower than the score of WSDL method.

Results are collected in Table 4.2. Altogether, we can say that our approach performs relatively better than those from previous works and scores are quite high.

Table 4.2 Experimental result in comparing six methods with four dimentions

Method	Total		Total + partial	
	Precision	Recall	Precision	Recall
WSDL	$\frac{8}{22}$	$\frac{22}{83}$	$\frac{14}{38}$	$\frac{38}{95}$
Text	$\frac{14}{22}$	$\frac{22}{56}$	$\frac{24}{38}$	$\frac{38}{83}$
Operation ontology	$\frac{13}{22}$	$\frac{22}{44}$	$\frac{34}{38}$	$\frac{38}{97}$
Integrated method with concept relationship	$\frac{20}{22}$	$\frac{22}{40}$	$\frac{34}{38}$	$\frac{38}{62}$
Integrated method with LIN	$\frac{19}{22}$	$\frac{22}{41}$	$\frac{34}{38}$	$\frac{38}{61}$
Integrated method with WUP	$\frac{19}{22}$	$\frac{22}{40}$	$\frac{34}{38}$	$\frac{38}{73}$

CHAPTER 5 - Conclusion

In this thesis, we've investigated the necessary information for web services for discovery, and offer a representation to organize them. Although there are a lot of existing representations, none of them covers all the useful information for user. Since some of the information are only related to web service selection or usage, we concentrate on those attribute related to web service discovery in our methodology and evaluation. To set up a efficient framework of web services discovery, three integrated method are proposed, covering text-based and ontology-based matchmaking for different attribute, and weights are added. After the experiment, we can find out that our approaches perform evidently better than methods of previous works.

For the future work, there are still some issues can be put under consideration. There are still two attributes: precondition and effect, not being in the discovery process since they are not yet well-defined in the existing web services. With these two attributes, services usage and result may be matched better. Also, more specific methods considering semantic meanings can be used to compute the similarities of text-based attributes.

References

Chen Wu, & Chang, E. (2007). Searching services "on the web": A public web services discovery approach. *Signal-Image Technologies and Internet-Based System, 2007. SITIS '07. Third International IEEE Conference on,* 321-328.

Chen Wu, Chang, E., & Aitken, A. (2008). An empirical approach for semantic web services discovery. *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on,* 412-421.

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S. (2001). *Web service definition language (WSDL).* Retrieved 6/11/2010, 2010, from http://www.w3.org/TR/wsdl

Clement, L., Hately, A. & Riegen, C. R., T. (2004). *UDDI spec technical committee draft.* Retrieved 6/11/2010, 2010, from http://uddi.org/pubs/uddi_v3.htm

Fensel, D., & Bussler, C. (2002). The web service modeling framework WSMF. *Electronic Commerce Research and Applications, 1*(2), 113-137.

Gao, S., Sperberg-McQueen, C. M. & Thompson, H. S. (2009). *W3C XML schema definition language (XSD) 1.1 part 1: Structures.* Retrieved 6/11/2010, 2010, from http://www.w3.org/TR/xmlschema11-1/

Jaeger, M., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., & Geihs, K. (2005). Ranked matching for service descriptions using owl-s. *Kommunikation in Verteilten Systemen (KiVS),* 91-102.

Lee, K. C., Jeon, J. H., Lee, W. S., Jeong, S. H. & Park, S. W. (2003). *QoS for web services: Requirements and possible approaches.* Retrieved 6/11/2010, 2010, from http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

Martin, D., et al. (2004). *OWL-S: Semantic markup for web services.* Retrieved 6/11/2010, 2010, from http://www.w3.org/Submission/OWL-S/

Medjahed, B., Bouguettaya, A., & Elmagarmid, A. K. (2003). Composing web services on the semantic web. *The VLDB Journal, 12*(4), 333-351.

Miller, G. A., Fellbaum, C., Tengi, R., Langone, H., Ernst, A. & Jose, L. (2010). *WordNet-A lexical database for english.* Retrieved 6/11/2010, 2010, from http://wordnet.princeton.edu/

Mitra, N., & Lafon, Y. (2007). *SOAP version 1.2 part 0: Primer (second edition).* Retrieved 6/11/2010, 2010, from http://www.w3.org/TR/soap12-part0/

Paolucci, M., Kawamura, T., Payne, T., & Sycara, K. (2002). Semantic matching of web services capabilities. *The Semantic Web—ISWC 2002*, , 333-347.

Pedersen, T., Patwardhan, S., & Michelizzi, J. (2004). Wordnet:: Similarity-measuring the relatedness of concepts. *Proceedings of the National Conference on Artificial Intelligence,* 1024-1025.

Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C. M. & Thompson, H. S. (2009). *W3C XML schema definition language (XSD) 1.1 part 2: Datatypes.* Retrieved 6/11/2010, 2010, from http://www.w3.org/TR/xmlschema11-2/

Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research, 11*(4), 95-130.

Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *International Journal of Services Sciences, 1*(1), 83-98.

Umesh Bellur, & Kulkarni, R. (2007). Improved matchmaking algorithm for semantic web services based on bipartite graph matching. *IEEE International Conference on Web Services, ICWS 2007*. 86-93.