Extracting Semantic Constraint from Description Text for Semantic Web Service Discovery*

Dengping Wei[1], Ting Wang[1], Ji Wang[2], and Yaodong Chen[1]

[1] Department of Computer Science and Technology, School of Computer, National University of Defense Technology, Changsha,Hunan, 410073, P.R. China
{dpwei,tingwang,yaodongchen}@nudt.edu.cn
[2] National Laboratory for Parallel and Distributed Processing, Changsha,Hunan, 410073, P.R. China
jiwang@mail.edu.cn

Abstract. Various semantic web service discovery techniques have been proposed, many of which perform the profile based service signature (I/O) matching. However, the service I/O concepts are not sufficient to discover web services accurately. This paper presents a new method to enhance the semantic description of semantic web service by using the semantic constraints of service I/O concepts in specific context. The semantic constraints described in a constraint graph are extracted automatically from the parsing results of the service description text by a set of heuristic rules. The corresponding semantic web service matchmaker performs not only the profile's semantic matching but also the matching of their semantic constraints with the help of a constraint graph based matchmaking algorithm. The experiment results are encouraging when applying the semantic constraint to discover semantic web services on the service retrieval test collection OWLS-TC v2.

1 Introduction

Semantic web services (SWS) have attracted a significant amount of attention in recent years. The aggregation, including description and discovery of services plays an important role in various internet-based virtual computing environments [1]. SWS discovery is the process of locating existing web services based on the description of their functional and non-functional semantics [2]. Most SWS matchmakers perform the matching of service profile rather than service process model. SWS profile describes the services capabilities in terms of several elements, including its inputs(I), outputs(O), preconditions/assumptions(P) and effects/postconditions(E) [3].

* This research is supported by the National Grand Fundamental Research Program of China under Grant No. 2005CB321802, the Program for New Century Excellent Talents in University(NCET-06-0926), and the National Natural Science Foundation of China (60403050, 90612009).

A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 146–161, 2008.

Various SWS description languages such as OWL-S [3], WSMO [4], WSDL-S [5], SAWSDL [6], provide different frameworks to describe SWS. There are also various SWS matchmakers based on the respective profile elements: some perform logic based semantic IOPE matching [7] [8], and some others perform logic based semantic service signature (Input/Output) matching [9,10,11,12,13, 14,15,16].

A common characteristic of most current SWS matchmakers is that the semantic matching between a pair of SWS concepts annotated to the input and output parameters almost depends on the subsumption relations in the domain taxonomy. Most current SWS matchmakers treat the SWS signature as a set of concepts which are however not sufficient to discover SWS effectively when using logic based reasoning. Two services with similar real world semantics may fail to match, and even two services with the same input and output concepts may have essential differences in semantics which cannot be detected by logic based reasoning.

In order to overcome this problem, many recent researches have explored various information to complement service I/O concepts for SWS matchmaking. The ranked matching algorithm [9] explores the service category and service quality together with its I/O concepts to compute the combined degree of match between the request and the advertisement. Klusch et al. [7] [10] have proposed a hybrid method for SWS discovery which utilizes both the logic based reasoning and the content (unfolded concept expressions) based Information Retrieval(IR) techniques to remedy this limitation. Kiefer et al. [17] have proposed a new approach to perform SWS matchmaking based on iSPARQL strategies which combines structured and imprecise querying together on a diverse set of syntactic description information (service name, service description text, etc.). The work in [18] describes the relationships between inputs and outputs explicitly and uses OWL ontologies to fix the meaning of the terms used in a service description.

Most SWS discovery approaches consider each SWS signature as a bag of concepts and ignore the relationships between the concepts in the SWS profile. The relationships between the service I/O concepts, called semantic constraints in this paper, can be helpful for expressing the semantics of services and improving the existing SWS discovery methods in practice if they can be generated automatically. Motivated by this idea, we add some restriction relationships to the interface concepts to enhance the semantic description of services. These restriction relationships which may not be defined in the domain ontology can be extracted from the service description text automatically. A novel SWS discovery mechanism has been proposed to perform the matching on both the service I/O concepts and their semantic constraints which are represented by a constraint graph.

The rest of this paper is organized as follows. Section 2 gives the definition of semantic constraint for SWS and the constraint graph. Section 3 describes the semantic constraint extracting method. A Constraint Graph-based Matching (CGM) algorithm is proposed in section 4. Section 5 evaluates the proposed approach on

the OWL-S service retrieval test collection TC v2[1]. Section 6 discusses the related work. The conclusion and future research are given in section 7 finally.

2 Semantic Constraint for Semantic Web Service Discovery

2.1 Motivation

Many SWS matchmakers like OWLM [19], OWLS-MX [10], OWLS-UDDI[2], Lumina[3] perform logic based semantic matchmaking on service I/O concepts. As discussed in section 1, representing SWS by the service I/O concepts is not sufficient enough for service discovery. Some important facts can be observed from the existing SWS collections.

(1)**The domain of concept is not specified**. The service I/O concepts usually describe abstract things like "price", "distance". Normally their meanings are heavily related to the context, that is, it is difficult to get its exact semantics unless they are set in a certain context, such as "the price of a book", "the price of a flight ticket", "the distance between two cities", "the distance between two stars". So if a service is only annotated with the concept ***Price*** as its output, it is still not clarified whose price is returned by this service.

(2) **The property of concept is not specified**. Every concept defined in domain ontology may have several properties to restrict its semantics. All the individuals of each concept can be divided into several categories according to different property restrictions and values. For example, the concept ***Bag*** has a property ***hasColor***, so the individuals of ***Bag*** can be classified into different sets according to their values of the property ***hasColor***, such as the red bags, the blue bags, the green bags and so on. Therefore, in specific context, the semantics of I/O concepts can be better clarified if they are associated with their property values. Meanwhile, the concepts with different property restrictions may correspond to various individuals in respective context. For example, service A returns a kind of food information with the maximum price, while service B returns the food information with brand "Coca Cola". We cannot assert that these two kinds of food information are similar. Also the functions of the two services cannot to be asserted to be similar.

(3) **The relationship between concepts is not specified**. Two web services annotated with the same input and output concepts may have essentially different semantics which cannot be detected by logic based reasoning. The difference may be caused by the diverse relationships between the input and output concepts. For example, both service A and service B have been annotated with concept ***GroceryStore*** for their input parameters and concept ***Food*** for their

[1] http://www-ags.dfki.uni-sb.de/~klusch/owls-mx/
[2] http://www.daml.ri.cmu.edu/matchmaker/
[3] http://lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/

output parameters. But service A returns the food information contained in a certain grocery store while service B returns the food information sold by a certain grocery store. These two web services have the same interfaces and the same annotated concepts which the interface refers to, but their functions are totally different.

2.2 Constraint Types Definition

According to the above facts, the semantics of SWS will be better clarified if the constraint relationships of the concepts have been annotated. Each concept used to annotate web service can take some kinds of constraints. A concept and one of its constraints can be represented by a statement $\langle SC, CT, OC\rangle$ using RDF[4] terms:

- SC (Subject Concept), is the subject of the statement and usually corresponds to the service I/O concepts. It specifies the thing the statement is about.
- OC (Object Concept), is the object of statement. It can be described as another concept or a literal.
- CT (Constraint Type), is the predicate of the statement which identifies the property or characteristic of the subject concept that the statement specifies. There are many constraint relationships between two entities in the real world, which are also true in web service domain. We choose three important abstract constraint types, considering that not all the realistic constraints can be extracted accurately and automatically from description text.
  - *isPropertyObjectOf* Constraint: triple $\langle A, isPropertyObjectOf, B\rangle$ means that concept A is a property object of concept B. It specifies the domain which concept A belongs to, that is, the individuals of concept A in specific service are the property values of individuals of concept B. For example, if a service has been annotated with an output concept ***Price*** and the price has a constraint triple $\langle \boldsymbol{Price}, isPropertyObjectOf, \boldsymbol{Car}\rangle$, it means that this service returns the price of a certain car.
  - *hasPropertyObject* Constraint: this constraint relation is the inverse of *isPropertyObjectOf*. Triple $\langle A, hasPropertyObject, B\rangle$ means that concept A is the one which has an inherent property object concept B. For example, if a service has been annotated with an output concept ***Car*** that has a constraint triple $\langle \boldsymbol{Car}, hasPropertyObject, \boldsymbol{Price}\rangle$, it means that this service returns the information of a car associated with a value of price.
  - *Operation* Constraint: triple $\langle A, Operation, B\rangle$ means that the two concepts entities have a certain association between them. The word "Operation" is an abstract word representing all kinds of properties. For example, if a service has been annotated with an output concept ***Book*** and the concept ***Book*** in this web service has an "*Operation*" constraint triple $\langle Book, \text{"}published\ by\text{"}, \text{"}Springer\text{"}\rangle$, it means that this service returns the books that are published by Springer.

[4] http://www.w3.org/TR/rdf-primer/

2.3 Constraint Graph Definition

After adding constraints defined above to service I/O concepts, the service semantics has been enriched and SWS is described by both service I/O concepts and their constraints. In this paper, a concept together with its constraints is described in a constraint graph. Let C be a set of concepts, a directed constraint graph can be described as $ConstraintGraph(C) = \{\langle SC, CT, OC\rangle | SC \in C\}$.

The snippet of a food querying service profile is described in Fig.1(a). This service returns the food information in a certain store together with their quantity, annotated with one input concept ***Store*** and two output concepts ***Food*** and ***Quantity***. After adding *isPropertyObjectOf* constraints to concepts ***Food*** and ***Quantity***, the connected constraint graph which is depicted in Fig.1(b) indicates the relationships between ***Food*** and ***Store***, ***Quantity*** and ***Food***. The description of service's semantics in Fig.1(b) is clearer than the set of three concepts described in Fig.1(a).

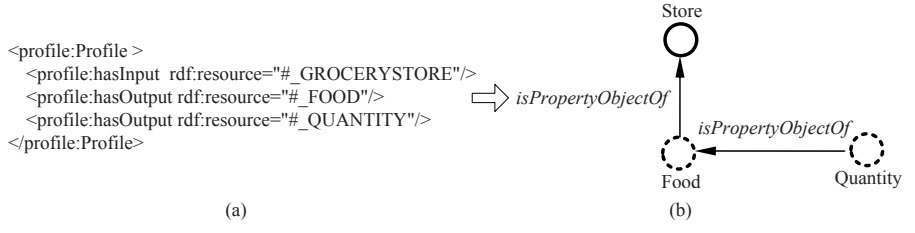

Fig. 1. (a) The snippet of a food service profile; (b) The constraint graph representation of the service (the dashed circle denotes output concept and the solid circle denotes input concept)

3 Extracting Semantic Constraint

The description texts in web services are important knowledge sources for service discovery. The constraints of a certain concept can be extracted from the description texts according to the definition of concept in domain ontology, especially the definition of property whose subject is the concept. However, in practice, few domain ontologies are comprehensive enough to provide plenty constraints information, as every concept in the domain ontology has only limited kinds of properties. In this section, an extraction method based on the parse trees of description text is proposed to obtain the semantic constraints of service I/O concepts.

3.1 Overview

The semantic constraints for input/output concepts of a web service can be extracted from the description text of this web service. Each concept corresponds to a sequence of words (called key-word) in the description text, and the syntactic relations between the key-word and other words may be the semantic

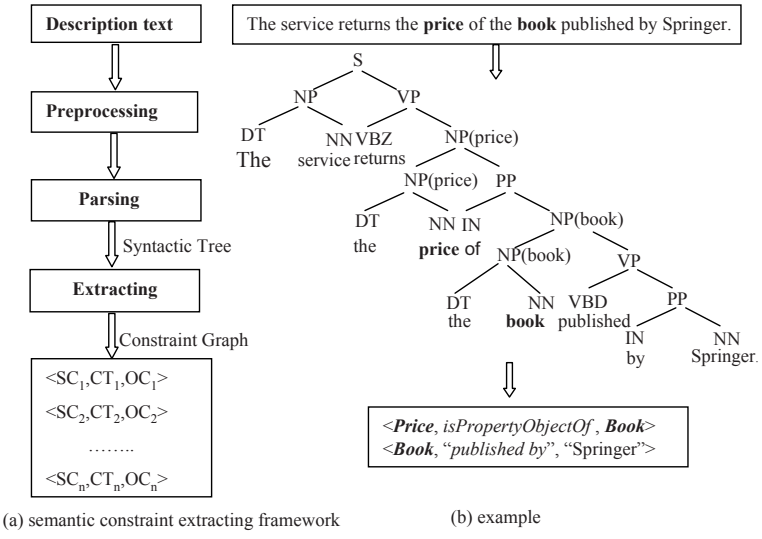

Fig. 2. Semantic constraint extraction

constraints of the corresponding concept. The syntactic structures of the service description text give rich information about the constraint types for service I/O concepts. Thus, the semantic constraints of I/O concepts could be derived from the syntactic tree. Unlike the ontology-based information extraction or relation extraction, we detect the semantic constraints for the service I/O concepts rather than their instances. In ontology-based information extraction, the relation extraction focuses on the relationship between two specified entities.

The semantic constraint extraction which is based on the parsing tree of the sentence consists of preprocessing, syntactic parsing and heuristic-based extracting (shown in Fig.2(a)). During preprocessing, some pre-selected key-words representing the service I/O concepts are tagged in the description text. And then, the text is parsed syntactically to identify the constituents modifying the key-words. Finally, several heuristics rules are used to extract constraint triples about the key-words from the syntactic constituents.

3.2 Preprocessing

The aim of preprocessing is to detect the key-words and process the text in order to improve the precision of syntactic parsing in the next step. Some more details are as follows.

- **Key-words detection**. The concepts in service I/O are annotated to a sequence of words in the description text, that is, each concept is instantiated by a word sequence through scanning all these fragments in text.
- **Name Entity Recognition**. ANNIE Gazetteer in GATE [20] is used to recognize the name entities in the text which are useful for the matching.

For example, the word "Japan" is annotated as a country, so it can match the concept ***Country*** in a geographical ontology.
- **Tokenization**. Corresponding to each service I/O concept, a key-word may include several tokens. A key-word should be a terminal node in the parsing tree in order to extract its constraints correctly. Therefore, each key-word is considered as one token.
- **Part Of Speech tagging for special words**. Several rules are designed to assign Part Of Speech (POS) tags to some important ambiguous words which can help improving the precision of parsing. Based on the observation that concepts in domain ontologies are usually nouns, we specify each key-word to noun and some important words to verb, e.g., "return", "provide".

3.3 Extracting Semantic Constraint

We firstly obtain the syntactic trees by parsing the sentences which contain the key-words in the description text. The semantic constraints of the key-words are identified according to syntactic relationships in parsing trees such as modification and represented in triples each of which includes a key-word, a constraint type and a constraint constituent. The extraction includes three phases as follows.

Candidate Constituent Detection. From the observation, the constraints of a key-word are probably contained in the phrase whose head word is the key-word. All such phrases can be detected by propagating the key-word from the bottom to the top of the syntactic tree. The propagation path is expressed as a sequence of interior nodes labeled by nonterminal categories in the parsing tree, e.g. a node sequence "NP NP" in Fig.2(b) is the propagation path of key-word "price".

The constituents contained in each phrase which is in the propagation path are called candidate constituents which may contain modification information of the key-word. For example, Fig.2(b) describes the parsing tree of the sentence "The service returns the price of the book published by Springer.", in which the key-word is "price". The propagation path of the key-word "price" is "NP NP". The candidate constituents contained in the propagation path are "DET" and "PP".

Constraint Constituents Filtering. In candidate constituents, some function words, such as "DET" in Fig.2(b), are not valuable modifiers for the key-word. Only the constituents that contain useful modifiers of the key-word are considered as constraint constituents. Thus, the constraint constituent of the key-word "price" is the constituent tagged with "PP".

Various constraint types defined in section 2 often have different syntactic characteristics and are expressed in diverse constituents, so respective rules are designed to filter the useful constraint constituents from all the candidate constituents. The second column in Table 1 presents the rules that can filter the useful constituents from the candidate constituents for the constraint types shown in the first column. As the *hasPropertyObject* constraint relationship is the inverse

Table 1. The semantic constraint extraction rules

Constraint Type	Candidate Constituents Filtering Rules	Modifiers Extraction Rules
isProperty ObjectOf	Rule 1: The candidate constituents tagged with "JJ", "PP", and "Pronoun" are indicator of constituents in which *isPropertyObjectOf* constraint locates.	Rule 2: Extract the noun string from the adjective if it's suffix is "'s".
		Rule 3: Extract the key phrase from the PP phrase whose head word is "of".
		Rule 4: Identify the reference word in possessive pronoun phrase.
Operation	Rule 5: The candidate constituents tagged with "VP" and "SBAR" are indicator of constituents in which *Operation* constraint locates.	Rule 6: Extract the verb, preposition, noun from the VP phrase.
		Rule 7: Extract the verb, preposition, noun from the SBAR phrase.

of *isPropertyObjectOf* constraint, we can get the *hasPropertyObject* constraints from the *isPropertyObjectOf* ones.

By analyzing the structure of the parsing tree, the *isPropertyObjectOf* constraint of the key-word can be extracted from either its sibling nodes or its parent's sibling nodes which are tagged with "JJ", "PP" or "Pronoun". The *Operation* constraint often locates in the constituents tagged with "VP" or "SBAR". Rule 1 and Rule 5 are designed to extract the candidate constituents for *isPropertyObjectOf* constraint and Operation constraint respectively. As depicted in Fig.2(b), the *isPropertyObjectOf* constraint of "price" locates in the constituent tagged with "PP".

Extracting Modifier. After the above two steps, the modifiers now can be extracted from the constraint constituents identified by the Rule 1 and Rule 5 in Table 1. Not all the constraint constituents provide useful constraints for the key-word. Some rules listed in the third column in Table 1 are also used to extract the modifiers of the key-word for specific type.

Rules 2-4 for *isPropertyObjectOf* constraint are motivated by the observation that the *isPropertyObjectOf* modifiers to nouns are usually locates in a PP phrase, an adjective or a possessive pronoun. Rule 2 states that only the adjectives like "book's" are the constraint constituents of the key-word. Taking the noun phrase "book's price" for example, the constraint constituent of the key-word "price" is the adjective "book's". Only the word "book" which can be extracted from the adjective is the *isPropertyObjectOf* modifier of the key-word "price". Rule 3 indicates that the *isPropertyObjectOf* modifier often locates in the PP phrase whose head word is "of". Rule 4 is supported by the observation that nouns are usually modified by a possessive pronoun like "its". The noun that refers to the pronoun is identified as the *isPropertyObjectOf* modifier.

Rules 6-7 for *Operation* constraint are supported by the observation that the candidate constituents tagged with "VP" and "SBAR" are good indicator of constituents where *Operation* constraint locates. The verbs in verb phrase and SBAR clause usually describe the relationship between two entities and the following preposition is a good indicator of the voice of the sub-sentence in which the VP phrase locates. If the verb is a transitive verb and the following word is a noun, then the *Operation* constraint is composed by the verb and the noun which is the object of the verb. A preposition "by" indicates that the noun following it would be extracted as the subject of the operation which the verb represents.

Finally, the constraint triples can be represented by the key-words and the extracted modifiers. In Fig.2(b), the modifier of "price" is "book" and the constraint triple is $\langle \boldsymbol{Price}, isPropertyObjectOf, \boldsymbol{Book} \rangle$. For the snippet of sentence "the book published by Springer", the constraint triple $\langle \boldsymbol{Book}$, "*published by*", "*Springer*"$\rangle$ indicates that the subject of the verb "publish" is "Springer" and it's object is the noun "book".

In this paper, we only consider these three constraint relations of the concept at a coarse granularity level. The extraction rules are very specific in order to achieve the high precision of extracting. If there need to extract more information, more rules should be added accordingly.

4 Matching Algorithm

According to the definition of constraint graph introduced in section 2, we have designed a three levels' matching algorithm to measure the match between two constraint graphs.

4.1 Constraint Graph Matching(CGM)

The degree of matching between the $ConstraintGraph(C_r)$ of the request and the $ConstraintGraph(C_s)$ of a service is computed by the following formula:

$$ConstraintGraphMatch(C_r, C_s) = \sum_{i=1}^{P} \max_{j \in P'}(TripleMatch(RT_i, ST_i))/P$$

where P is the number of triples contained in the constraint graph $ConstraintGraph(C_r)$, P' is the number of the triples contained in the constraint graph $ConstraintGraph(C_s)$ and the function $TripleMatch(RT_i, ST_i)$ is used to estimate the match between two triples $RT_i \in ConstraintGraph(C_r)$ and $ST_j \in ConstraintGraph(C_s)$.

4.2 Triples Matching

If there is a triple T_r in the constraint graph of the request and a triple T_s in the constraint graph of web service, the matching between two triples is computed as following:

- *Step1*: compute the degree of match between two subjects.
- *Step2*: compute the degree of match between two objects if the constraint types of the two triples are similar and their subjects are matched.
- *Step3*: compute the weighted sum of match value obtained from the above operations as the match value if all the elements in the triple are matched.

This matching algorithm states that the subsumption relation between two subjects only means that the triples are possibly matching. Only when all the three elements in each triple are relative, can we say that the two triples are matched and the degree of match can be measured.

4.3 Concept Matching

The matching between two concepts is based on the subsumption relationship reasoning on the taxonomies of domain ontologies. The logic based semantic matchmaker we use here is much more relaxed compared to the algorithm described in [7]. Five different levels for the degree of semantic matching are defined, that is, Exact, Plug-in, Subsumed-by, Intersect, and Fail. Let c be a concept, then $Parents(c)$ returns the set of the generic concepts that are the parents of concept c and $Children(c)$ returns the set of the specific concepts that are the children of concept c. The details of the logic based semantic matching are as follows.

- **Exact match**: concept r of the request exactly matches the concept s of service if and only if $r = s$. The concept r of the request perfectly matches the concept s with respect to logic based equivalence of their formal semantic.
- **Plug-in match**: concept s of service plug-in matches the concept r of request if and only if $r \in Parents(s) \vee s \in Children(r)$.
- **Subsumed-by match**: concept r of the request subsumed-by matches the concept s of service if and only if $s \in Parents(r) \vee r \in Children(s)$.
- **Intersect match**: concept r of request intersect matches the concept s of service if and only if

$$\frac{\left\|Parents(r) \wedge Parents(s)\right\|}{\max\left(\left\|Parents(r)\right\|, \left\|Parents(s)\right\|\right)} \geq 0.5$$

.
- **Fails**: concept r of the request does not match any concept of service according to any of the above match levels.

The definitions of the plug-in match and the subsumed-by match seem reduplicate. A is the parent of B logically equals that B is the children of A, but it is not always true in ontology reasoning. In ontology library, there exist some kinds of reference relationships between two ontologies, such as "import" relation. If two concepts come from different ontologies and there is no reference relationship between these ontologies, then the two concepts match fails. Let ontology ont_1 be imported into another ontology ont_2, that the concept B in ont_2 is the

child of concept A in ont_1 cannot infer that concept A is the parent of concept B, because the two concepts belong to different ontologies and the reasoning is based on different knowledge bases.

When the constraint graph based matchmaker coordinate with other matchmaker, the logic based semantic matchmaker used in it can be adapted accordingly. The semantic similarity between two literals is also measured by the distance in the lexical hierarchy which defined in WordNet dictionary. The degree of match is 1 if the two literals belong to the same synonym set.

5 Experiment Results and Analysis

The proposed method has been evaluated on the service retrieval test collection OWL-S TC v2, which consists of 576 web services from 7 domains, 28 queries (each query represents one request) with their relevance sets. Each web service in this dataset has only one operation. In this experiment, 27 queries are evaluated, except the one without output parameter. Two sets of web services using in this experiment have been transferred from OWL-S TC v2 by annotating output concepts with constraints: Dataset1 and Dataset2. In Dataset1, the semantic constraints of the output concepts in request and web service are manually annotated by two people and mainly described by service I/O concepts; while the semantic constraints of concepts in Dataset2 are automatically extracted using the method represented in section 3. The constraint graph based SWS matchmaker (described in section 4) called CGM is implemented in JAVA using Jena[5].

In the experiments, we measured the constraint graph based service retrieval performance. The Macro Averaged Recall Precision curves are shown in Fig. 3. OWLS-M0 is a pure logic based matchmaker on the service I/O concepts [10]. OWLS- M4 is reported to be the best-performing matchmaker variant of the OWLS-MX matchmaker [10] which uses Jensen-Shannon information divergence to compare the request and service based on the unfolded concept expressions. We also use the nearest-neighbor as the minimum degree of match and a value of 0.7 as syntactic similarity threshold for OWLS-M4. These values were suggested by the authors of OWLS-MX to obtain better results for OWLS-TC v2. InOutConstraint matchmaker and AutoConstraint matchmaker use CGM to compare two services based on their semantic constraint graph, which have been evaluated on Dataset1 and Dataset2 respectively. M0+InOutConstraint matchmaker uses CGM to filter the results of OWLS-M0 on Dataset1. M0+AutoConstraint matchmaker uses CGM to filter the results of OWLS-M0 on Dataset2. M4+InOutConstraint matchmaker uses CGM to filter the results of OWLS-M4 on Dataset1. M4+AutoConstraint matchmaker uses CGM to filter the results of OWLS-M4 on Dataset2. For running OWLS-MX variants, we use OWLS-iMather[6]. All these datasets and matchmakers are available for download[7]. From

[5] http://jena.sourceforge.net/

[6] http://www.ifi.uzh.ch/ddis/research/semweb/imatcher/

[7] http://nlp.nudt.edu.cn/∼dpwei/

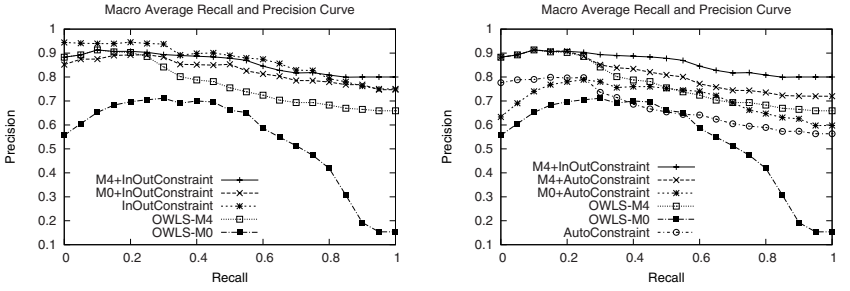

Fig. 3. Macro Average Recall-Precision Curves. (a). The performance on Dataset1; (b). The performance on Dataset2.

the preliminary experimental results depicted in Fig.3, the following facts can be observed.

(1) Both InOutConstraint and OWLS-M4 outperform OWLS-M0 as shown in Fig.3(a). This fact indicates that pure service I/O concepts based matchmaker is not effective to discovery web services. OWLS-M0 often returns some irrelevant services rather than relevant services according to the logic-based semantic filter criteria. While OWLS-M4 can use the syntactical similarity filter (matching of the unfolded concept expressions) to find relevant services that OWLS-M0 would fail to retrieve. InOutConstraint can perform the semantic matching more accurately by enriching the description of SWS. The logic based matchmaker for two concepts which differs from that of OWLS-MX variants uses more relaxed notions for matching in order to retrieve more services, and the constraints of I/O concepts can help filter the irrelevant ones. It also indicates that we should use more relaxed semantic based matchmaker when using the constraint graph based matchmaker.

(2) Fig.3(a) also shows that both M0+InOutConstraint and M4+InOutConstraint outperform their corresponding OWLS-MX variants in OWLS-TC v2. InOutConstraint can filter the irrelative services which are returned by OWLS-MX variants because of the constraints matching. In OWLS-TC v2, there are about 10% web services that have no input parameter and also some web services with matching I/O concepts but different functionalities, which increases the probability of returning irrelevant services by OWLS-MX variants. For example, the "_food_Exportservice.owls" service only has an output concept ***Food*** and returns the exported food information. The query "grocerystore_food_service.owls" which requires services that return the food owned by a certain grocery store has an input parameter ***GroceryStore*** and an output parameter ***Food***. The query and the service implement different functions but OWLS-MX variants judge that the service matches the query because of the same output concept.

(3) InOutConstraint outperforms both the composed matchmaker M4+ InOutConstraint and OWLS-M4 as illustrated in Fig.3(a). The relaxed logic based semantic matchmaker defined in section 4.3 can retrieve more web services to

improve the recall, while the constraints which make the matching of concepts more accurately are benefit for filtering the irrelative services, because the constraints that InOutConstraint use are annotated by people carefully.

(4) M0+AutoConstraint outperforms OWLS-M0 and M4+AutoConstraint also outperforms OWLS-M4 as shown in Fig.3(b). This fact illustrates that the semantic constraint extracting method for service I/O concepts is effective to filter the irrelative services that OWLS-MX variants return, although the improved performance is lower than that in Dataset1. With the improvement of the extracting performance, the curve of M0+AutoConstraint would approach the curve of M0+InOutConstraint. AutoConstraint depicted in Fig.3(b) has lower performance than M4+AutoConstraint, even OWLS-M4, which suggests that the constraints extracted from the description text automatically are not good enough to filter all the irrelative services that the relaxed logic based semantic matchmaker returns.

From the above experiments, we can see that the semantic constraints of service input/output concepts enrich the description of services capabilities and alleviate the unclear problem of semantic web services that described by only I/O concepts. Semantic constraint of service I/O concepts can distinguish the similar web services to improve the precision of discovery task. The semantic constraint based matchmaker could combine to other SWS matchmakers to improve their performance. The matchmaker CGM explores the constraints information to filter the irrelevant services which probably match the request by pure logical reasoning, while the hybrid matchmaker emphasizes particularly on retrieve those services that fail to retrieve according to pure logic based matching. So they can work with each other without conflict. However, the performance of our matchmaker depends on the performance of constraints extraction. The best performance displayed in Fig.3(a) is InOutConstraint which indicates that semantic constraints of service I/O concepts are excellent in SWS discovery task if supported by a good constraints extracting result.

6 Related Work

Semantic web service discovery is a hot topic in the fields of both semantic web and web service. An abundance of different approaches for service matchmaking focuses on different aspects of service description including functional and non-functional ones. The work presented in this paper concerns only the function based matchmaking. The functional properties of SWS mainly include service inputs, service outputs, preconditions and effects [3]. Several studies concentrate on describing web service with richer semantics for discovery [7] [18], in which the matching has high complexity. Hull et al. [18] describe the relationships between inputs and outputs explicitly and use OWL ontologies to fix the meaning of the terms used in a service description. The description capability depends on the domain ontologies, and the service descriptions are mainly established by domain experts. The work presented in [21] [22] focus on annotating services I/O parameters with ontology concepts (semi-) automatically, while our work is to add the semantic constraints to these concepts.

The majority of the research work in the literature [9, 10, 12] focus on the matchmaking of service I/O, i.e. the data semantics of a web service. Different methods extend service I/O matching from different aspects of web service.

Paolucii et al. [12] consider the service profile and its inputs and outputs for determining the match between the request and advertisement. In [9], the ranked matching algorithm is based on service description, including service inputs, service outputs, service quality and service category. The matching of service inputs and outputs depends on the subsumption relations in domain ontology. The proposed method in this paper extends the matching of service I/O concepts by not only the taxonomy hierarchy in domain ontology but also the semantic constraints of each concept extracted from specific service context.

OWLS-MX [10] uses a hybrid SWS matching that complements logic based reasoning with approximate matching based on syntactic similarity computations. The matchmaker in this paper is similar to OWLS-MX but differs from it in several aspects. Firstly, they explore different information for matchmaking except service I/O concepts. The information of web service used in OWLS-MX is the unfolded concept expressions and the corresponding matching depends on the structures of the domain ontologies. In our method, the constraint information is extracted from the description text of web service and the matchmaker is less dependent on the domain ontologies. Secondly, they have different ways to improve the recall and precision. The reason that the hybrid variants in OWLS-MX outperforms the OWLS-M0 is that the matching of unfolded concept expressions can return those syntactically similar but logically disjoint services as the answer set. The reason that our method outperforms the OWLS-M0 is that it can remove those false web services that match the request on service I/O concepts by their constraints matching. Finally, our method mainly focuses on semantic constraint of service I/O concepts, it is expected to be easily utilized in other kinds of SWS matchmakers.

7 Conclusions and Future Work

Web service discovery is a significant challenge. Because of the low precision of current key word based discovery mechanism, most work has focused on logic based discovery of semantic web service recently. The majority of the work performs profile based service signature (I/O) matching. However, the service I/O is not sufficient to describe the function of web service clearly. This paper mainly works on enhancing the semantics of web service through introducing semantic constraints to service I/O concepts. Constraint graph is designed to describe the semantic constraints of the service I/O concepts. The semantic constraints of concepts can be extracted automatically from the parsing trees of the description text. Meanwhile, a matching algorithm for the constraint graph is proposed. The semantic similarity between the request and service is measured by the degree of matching between their corresponding constraint graphs.

Preliminary results of our comparative experiments show that building constraint graph based SWS matchmakers is more sufficient than purely service

I/O based matchmaker. Semantic constraints of service I/O concepts can improve the precision of semantic matching and easily be plugged into any other SWS matchmakers as long as they have the same logic based semantic filters.

The performance of our method depends on the performance of constraints extraction, finding more efficient extraction method to get better results of extraction is the work in the future. We are planning to extract more constraint relationships for the concepts that appear in the web service description, then web service can be represented by a more complicated graph and the matching algorithm will be more sophisticate.

References

1. Lu, X., Wang, H., Wang, J.: Internet-based virtual computing environment(ivce): Concepts and architecture. Science in China Series F: Information Sciences 49(6), 681–701 (2006)
2. Klusch, M.: Semantic services coordination. In: Schumacher, M., Helin, H., Schuldt, H. (eds.) CASCOM - Intelligent Service Coordination in the Semantic Web, ch. 4. Springer, Heidelberg (2008)
3. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., Mcllraith, S., Narayanan, S., Paolucci, M., Parsia, B., Terry, P., Sirin, E., Srinivasan, N., Sycara, K.: Owl-s: Semantic markup for web services (2004), `http://www.w3.org/Submission/OWL-S/`
4. de Bruijn, J., Bussler, C., John, D., Fensel, D., Hepp, M., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Oren, E., Polleres, A., Scicluna, J., Stollberg, M.: D2v1.3. web service modeling ontology (wsmo) (2006), `http://www.wsmo.org/TR/d2/v1.3/`
5. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., Verma, K.: Web service semantics - WSDL-S (2005), `http://www.w3.org/Submission/WSDL-S/`
6. Farrell, J., Lausen, H.: Semantic annotations for WSDL and XML schema (2007), `http://www.w3.org/TR/sawsdl/`
7. Kaufer, F., Klusch, M.: Wsmo-mx:a logic programming based hybrid service matchmaker. In: 4th European Conference on Web Service, Zurich, Switzerland, pp. 161–170. IEEE CS Press, Los Alamitos (2006)
8. Stollberg, M., Keller, U., Lausen, H., Heymans, S.: Two-phase web service discovery based on rich functional descriptions. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 99–113. Springer, Heidelberg (2007)
9. Jaeger, M.C., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., Geihs, K.: Ranked matching for service descriptions using owl-s. In: KiVS 2005, Informatik Aktuell, pp. 42–113 (2005)
10. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with owls-mx. In: 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS), Hakodate, Japan, pp. 915–922. ACM, New York (2006)
11. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., Miller, J.: Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. Information Technology and Management 6, 17–39 (2005)

12. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
13. Constantinescu, I., Faltings, B.: Efficient matchmaking and directory services. In: IEEE/WIC International Conference on Web Intelligence, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2003)
14. Srinivasan, N., Paolucci, M., Sycara, K.: Semantic web service discovery in the owl-s ide. In: 39th Hawaii International Conference on System Sciences, Washington, DC, USA, vol. 6. IEEE Computer Society, Los Alamitos (2005)
15. Klusch, M., Fries, B., Khalid, M., Sycara, K.: Owls-mx:hybrid semantic web service retrieval. In: 1st International AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA (2005)
16. Srinivasan, N., Paolucci, M., Sycara, K.: An efficient algorithm for owl-s based semantic search in uddi. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 96–110. Springer, Heidelberg (2005)
17. Kiefer, C., Abraham, B.: The creation and evaluation of isparql strategies for matchmaking. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 463–477. Springer, Heidelberg (2008)
18. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding semantic matching of stateless services. In: 21st National Conference on Artificial Intelligence (AAAI 2006), pp. 1319–1324 (2006)
19. Jaeger, M.C., Tang, S.: Ranked matching for service descriptions using daml-s. In: CAiSE Workshops, Riga, Latvia, pp. 217–228 (2004)
20. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: Gate: A framework and graphical development environment for robust nlp tools and applications. In: 40th Anniversary Meeting of the Association for Computational Linguistics (ACL 2002) (2002)
21. Heß, A., Kushmerick, N.: Assam:a tool for semi-automatically annotating semantic web services. In: 3rd International Semantic Web Conference, pp. 320–334 (2004)
22. Oldham, N., Thomas, C., Sheth, A., Verma, K.: Meteor-s web service annotation framework with machine learning classification. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 137–146. Springer, Heidelberg (2005)