A Hybrid Approach for Discovery of OWL-S Services Based on Functional and Non-Functional Properties

M. DEEPA LAKSHMI[1], Dr. JULIA PUNITHA MALAR DHAS[2]
[1]Department of Computer Applications
[2]Department of Computer Science and Engineering
Noorul Islam University
Kumaracoil, Thuckalay, Kanyakumari District-629180, Tamilnadu
INDIA
[1]deepasuresh12@gmail.com, [2]julaps113@yahoo.com

Abstract:- Web services are independent software systems designed to offer machine-to-machine interactions over the WWW to achieve well-described operations. Typically, service providers expose their services to the public by providing brief descriptions of the service's operations; the challenge is to discover the right service in response to a specific service request based on rather sparse service descriptions. In this work, we present a hybrid semantic web service discovery framework that offer semantic web service discovery based on both functional (Input, Output, Precondition and Effect - IOPE) and non-functional (text-description) properties of OWL-S (Semantic Markup for Web Services) services. For functional parameter matching, we have used the bipartite graph-based approach for matching the parameters of services. For non-functional parameter matching, we have used natural language processing techniques on the textual-description of a web service. The cumulative similarly measures determine the overall similarity of the advertised service with the service request. We evaluated the performance of our Service Matchmaking framework using the OWLS-TC4 (Test Collection version 4) dataset, and furthermore compared its performance with some existing discovery models. Our results indicate that the proposed web service matchmaking framework offers an improved discovery mechanism with a significant increase in recall rate.

Key-Words:- Bipartite matching, Discovery, Functional parameters, OWL-S, Text-description

1 Introduction

Web Services are loosely coupled software components that are published, located and invoked across the web. Recently, web services have gained an increasing popularity. Since the existing traditional technologies describe web services only in syntactical level, it is difficult for service requestors and service providers to interpret or represent information such as the meaning of inputs and outputs or applicable constraints. Semantic description of web services can help overcome this difficulty. Semantic web service (SWS) is a web service which is semantically rich and defined through service ontology, capable of automatic discovery, execution, composition and interpretation.

OWL-S [1] is one of the proposals for describing semantic metadata about web services, which is based on the OWL (Web Ontology Language) ontology language. The OWL-S ontology is organized in three modules: the Service Profile module describes the functionality of the service; the Service Model module describes how it does it; and the Service Grounding module describes how to access the service. Semantic service discovery is the process of locating existing web services based on the description of their functional and non-functional properties of the Service Profile.

The proposed matching algorithm returns a ranked set of relevant services as its answer set to the user. For this purpose, it first uses the bipartite-graph based matching of the functional properties (IOPE). Next it performs some Natural language Processing (NLP) techniques like Part-Of-Speech (POS) tagging and Word Sense Disambiguation (WSD) and uses Jaccard similarity measure to perform the non-functional parameter (text-description) matching. Finally a cumulative similarity is calculated based on the above two similarity measures and a ranked set of relevant services are returned to the user.

2 Related Work

Majority of current Semantic web service discovery algorithms perform logic-based service profile

matching, and are restricted to OWL-S. The most influencing among them is Paolucci's algorithm [2], which has been cited in subsequent proposals. Paolucci proposed an ontology-based solution, in which matching of input and output parameters of services are done according to the hierarchical concept subsumption relationships defined in an ontology tree. There are four semantic similarity grades: Exact, Subsumes, Plugln, and Fail. Li and Horrocks [3] used a DAML-S (DARPA Agent Markup Language for Services) based ontology and a Description Logic (DL) reasoner to compare ontology based service descriptions. They extended the degrees of match of Paolucci's work by adding an intersection match. The hybrid semantic service matchmaker FCMATCH [4] performs a combined logic-based and text similarity-based matching of monolithic service and query concepts written in OWL-DL. Dong-Wei et al. [5] proposes a matching process in which firstly DL subsumption reasoning method is used to get the coarse set of services, and then more refined semantic distance calculation is made to improve the services distinguish capability. Their algorithm is based on input and output annotations.

Wang and Li [6] proposed a method for PE matching based on description logic SHOIN+(D). This method groups the matchmaking results in four categories: exact match, perfect match, side-effects match, and common match. This algorithm is not suitable for automatic matchmaking since it results in just a categorization of the match results and it is not designed for web services described in OWL-S either. Lamparter [7] presents an approach to hybrid matching of monolithic logic-based service descriptions in OWL-DL extended with pricing policies (modeled in DL-safe Semantic Web Rule Language (SWRL) rules) according to given references by means of SPARQL (Simple Protocol and RDF (Resource Description Framework) Query Language) queries to a given service repository. Similarly, Umesh Bellur's [8] work semantically matches requested and offered parameters, modeling the matchmaking problem as one of matching bipartite graphs. Peng and Shi [9] have replaced the match grades of Paolucci with fine values denoted by real number, and it is used to further rank advertisements. Wang et al.'s [10] work proposes a semantic match algorithm based on improved semantic distance. Bener et al. [11] considers semantic matching of input, output, precondition and effect. They also provide ranking. Liu et al. [12] achieve a fusion with five grades of matching, a collaboration of syntactic and semantic matching, as well as considering QoS (Quality of Service) and other dependency features.

The OWLS-MX [13] matchmaker performs hybrid semantic matching that complements logic based reasoning with syntactic IR (Information Retrieval) based similarity metrics. OWL-SLR [14] provides retrieval of services based not only on subsumption relationships, but also exploits the structural information of OWL ontologies. According to the work of Golsa Heidary [15], in first phase, two web services` Input / Output parameters are compared semantically. In second phase, services' parameter type is compared. In third phase, the matching rate of service is computed based on the results of first and second phase. Zhang et al. [16] proposed a way to precisely compute the similarity of concepts after classifying the services into five different matchmaking levels. The weighted semantic distance and the common features of concepts are considered in similarity computation. Cai et.al [17] proposes a semantic matchmaker, which focuses only on manufacturing domain. The similarity matching assumes either the total number of super classes subsuming the compared concepts or the total number of subclasses subsumed by the compared concepts in a shared ontological taxonomy. In addition, constraint reasoning is performed to deal with more complex matches. Sangers et al.'s [18] work uses natural language processing techniques for service discovery. But it focuses on WSMO (Web Service Modeling Ontology) service descriptions in WSML (Web Service Modeling Language), instead of the OWL-S specifications

Majority of the works mentioned above expects user input to be given in the form of a service description. Less support is provided for accepting user request in natural language. Also most of the work focuses only on the Input and Output parameter of the services. So our work proposes a hybrid matchmaking algorithm which accepts user input even in natural language and performs matchmaking based on both the text-description and the IOPE parameters of services and returns back a set of related services based on user request.

3 Semantic Services in OWL-S

Our semantic service matchmaker focuses on semantic services that are described in OWL-S. In the following, we briefly introduce the essentials of OWL-S.

3.1 Overview

OWL-S is an upper ontology used to describe the semantics of services based on the W3C standard ontology OWL and is grounded in WSDL (Web Service Description Language). It has its roots in the DAML Service Ontology (DAML-S) released in 2001, and became a W3C candidate recommendation in 2005. The OWL-S ontology consists of three main components: the *service profile* for advertising and discovering services; the *process model*, which gives a detailed description of a service's operation; and the *grounding*, which provides details on how to interoperate with a service.

In particular, the semantic service profile in OWL-S specifies the semantics of the service signature, which are the inputs required by the service and the outputs generated. Furthermore, since a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile also describes the preconditions to be satisfied before, and the expected effects that result from the execution of the service. The majority of existing OWL-S service matchmakers focuses on semantic service profiles.

3.2 OWL-S service profile

The OWL-S profile ontology is used to describe what the service does, and is meant to be mainly used for the purpose of service discovery. An OWL-S service profile or signature encompasses its functional parameters, i.e. Input, Output, Precondition and Effect, as well as non-functional parameters such as serviceName, serviceCategory, qualityRating, textDescription, and meta-data about the service provider such as name and location. Inputs and outputs relate to data channels, where data flows between processes. Preconditions specify facts of the world (state) that must be asserted in order for an agent to execute a service. Effects characterize facts that become asserted given a successful execution of the service in the physical world (state). In OWL-S, the semantics of each service input and output parameter is defined in terms of a referenced OWL concept in a given ontology, typically in a decidable description logic OWL-DL or OWL-Lite, the preconditions and effects can be expressed in any appropriate first order logic (rule) language such as KIF (Knowledge Interchange Format) or SWRL.

4 Proposed System

The overall architecture of the proposed hybrid Semantic web service discovery framework is shown in fig. 1. The proposed framework accepts the user request which may be in terms of some text description or the functional properties of required service like input, output, precondition, and effect. Next the user query information is parsed for further processing. The IOPE parameters are matched with the available services in the repository using bipartite graph-based approach. The text description is matched after performing some NLP techniques like POS tagging and Word sense disambiguation. Based on the cumulative similarity measure the retrieved relevant services are provided to the user. The architecture of the proposed work is explained below.

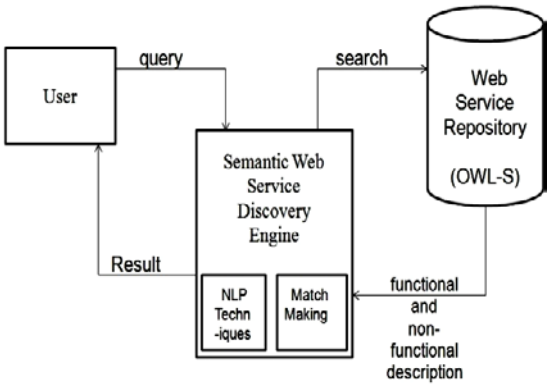


Fig.1. Overall architecture of the proposed system

4.1 Functional parameter matching

A good measure for calculating the degree of similarity of two services is the degree of similarity between their functional properties such as their inputs, outputs, preconditions and effects. For example, consider the OWL-S profile of a user request 'R' and an available advertisement 'A' in Fig. 2, taken from the standard test collection OWLS-TC4 (OWL-S Test Collection version 4). The user request 'R' expects to find a book selling service that accepts a user's payment card and will deliver the requested book and acknowledge to the user after the complete transaction is over. The e-shopping service 'S' like flipkart.com offers arbitrary articles including books that are requested by some customer whose own credit card account gets respectively charged while sending pricing information and the book to the customer. Both services are written in OWL-S with semantic signature (IO) concept definitions in description logic and their logical preconditions and effects (PE)

in SWRL. In the following, we assume the matchmaker to have an appropriate shared ontology (a portion of it is shown in fig. 2) and a service registry (OWL-S TC4) available, over which semantic service selection is performed. Our proposed matchmaking algorithm will be able to perform IOPE matching using bipartite graph approach and match the user request with the mentioned advertisement.

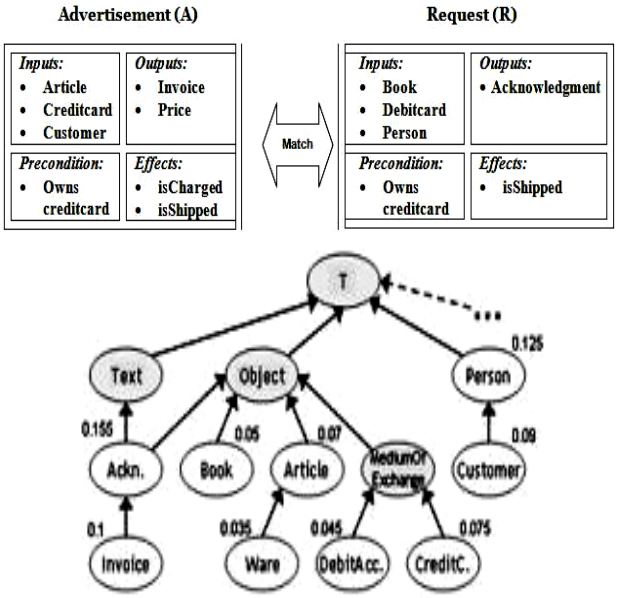


Fig. 2. Example Relevant Advertisement and service request and part of ontology

For performing functional parameter match using the bipartite graph approach, as a first step, the service reader extracts information from the <Profile:hasInput>, <Profile:hasOutput>, <profile:hasResult>, <profile:hasPrecondition> tags of the OWL-S description file. A snippet of an OWL-S file is shown in fig. 3.

```
<profile:Profile rdf:ID="EBookOrderProfile">
  <service:presentedBy rdf:resource="#EBookOrder1Service" />
  <profile:serviceName xml:lang="en">
EBookOrder</profile:serviceName>
  <profile:textDescription xml:lang="en">An e-book order web
service, where an ebook request is given by title and the
required book is placed into the users shopping cart. The
service also checks availability of the book in the stock and
verifies user's account.</profile:textDescription>
  <profile:hasInput rdf:resource="#EBookRequest" />
  <profile:hasInput rdf:resource="#UserAccount" />
  <profile:hasOutput rdf:resource="#EBook" />
   <profile:hasPrecondition rdf:resource="#Authorization" />
   <profile:hasResult rdf:resource="#BookOrdered" />
</profile:Profile>
```

Fig. 3. Snippet of an OWL-S service

Each service has a set of inputs and a set of outputs and each input/output parameter is semantically annotated with mapping it to an OWL concept. Therefore, the problem here is to calculate the similarity of two sets of concepts. We have used the bipartite graph based approach for this. In addition the precondition and effects are also considered. In the bipartite graph based matchmaking algorithm, the search procedure accepts a query as input and tries to match its output concepts and input concepts with each advertisement. If there exist a match in both input and output concepts, it appends the advertisement to the result set. To match inputs as well as it outputs, it uses a modified approach followed by Paolucci, where four degrees of match are considered (Exact, Subsume, Plug-in and Fail). And when multiple such matches occur, it invokes Hungarian algorithm [19], [20] on the graph created with suitable weights to compute an optimal matching of the graph. The degree of match is defined by the weight of the maximum-weight edge in the matching. In addition, the precondition and effects are also matched with the user request. In the end, a list of related advertised services are returned.

The algorithm for computing the degree of match of the output concepts is given in algorithm 1. Here outA and outQ represents an Advertisement's and Query's output concept respectively.

Algorithm: match(outA, outQ)
if outA = outQ then
    return Exact
else if outQ superclass of outA then
    return Plug-in
else if outQ subsumes outA then
    return Plug-in
else if outA subsumes outQ then
    return Subsumes
else
    return Fail

Algorithm 1. Degree of match

A bipartite graph is a graph in which the vertex set can be divided into two disjoint sets such that no edge of the graph lies between the two vertices in the same set. In our work, a bipartite graph is constructed using the IO concepts of Query and Advertisement.

A matching of a bipartite graph G = (V, E) is a sub graph G' = (V, E') such that no two edges e_1, e_2 in E' share the same vertex. Let the set of output concepts for query and advertisement be Q and A.

We will construct a graph G = (Q + A, E) which has one vertex corresponding to each concept in query and advertisement. If there exists a degree of match (≠ Fail) between a concept v_1 belonging to Q and a concept v_2 belonging to A, then we define an edge (v_1, v_2) with weight as the degree of match. We need a matching in which all the output concepts of Q are matched with some concept of A. If such a matching exists, we would say that the advertisement and the query match. If there exist multiple such matching, we will choose the one which is optimal. For, this we would assign different numerical weights to edges with different degrees of match. In this implementation we have assigned weights as 1 for Exact, multiplying number of vertices in Q by Exact weight plus one for Plug-in and multiplying number of vertices in Q by Plug-in weight plus one for Subsume [8]. Let max(w_i) be the maximum weighted edge in the matching. An optimal matching in this case would be a complete matching with minimum max(w_i). Hungarian algorithm is used for optimal matching, which computes a complete matching for a weighted bipartite graph such that sum of weights of all the edges in the matching is minimized. By this approach, a set of relevant services based on the user request is retrieved.

4.2 Non-functional parameter Matching

Sometimes logical matching fails because of the lack of the logical relationship between a pair of concepts in a domain ontology. Also, users may not be expert enough to specify their requirements in terms of OWL-S query. In such situations, totally relying on functional (signature) similarity measures would fail to discover two similar web services. Therefore, in order to get better results, specification (i.e. textual description) similarity measures should be combined with signature similarity measures. Service descriptions often contain parts which include textual information and the similarity of terms which are in these parts of the service descriptions can be used as an alternative measure for similarity of services. Usually, a textual description of a web service provides a brief functional description of what it is. For this, the information from the <textDescription> tag can be retrieved. The text-based similarity is done by performing three key steps. The first step of the proposed approach involves service reader, which will extract the information from the user query. Likewise, the description files of web services described in semantic language (OWL-S) is also parsed to get the text-description. Then the retrieved description of the web service and user request is tagged using POS tagger into set of nouns and verbs. The next step deals with word sense disambiguation, which disambiguates the word sense given by the user and description retrieved from the description file of web service (Algorithm 2). Finally Jaccard matching is used to discover the related web services based on the user's request. The steps are diagrammatically illustrated in fig. 4.

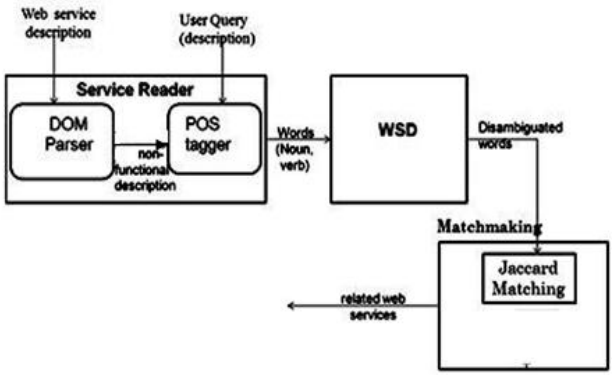


Fig. 4. Various phases of Text-based similarity matching

4.2.1 Service reader

The DOM (Document Object Model) parser is used to parse the required information from the web service Profile file. The information available in the <profile:textDescription> tag is extracted here.

Next, POS tagging is performed. A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word, such as noun, verb, adjective, etc. The Stanford POS tagger is used in our work. This software is a Java implementation. Each description is processed with the POS tagger. As a result, for each description D = w_1, w_2, …w_n, a string of Part-Of-Speech tags p_1,p_2,…p_n is produced, where $p_i \in P$ is the Part-Of-Speech tag chosen by Stanford tagger for word w_i, and P = {N, V} is a simplified set of syntactic categories (respectively, nouns and verbs).

4.2.2 Word sense disambiguation

Structural Semantic Interconnections (SSI) [21] is a knowledge based iterative approach to Word Sense Disambiguation. The purpose of SSI algorithm is to choose the correct sense of the given description.

```
Input:
  T : list of terms to be disambiguated
  I: list of interpretation synsets (initially empty)
Output:
  I: the list of sense-tagged keywords

for each t ∈ T
   if (t is monosemous) I(t) = the only sense of t
   P = {t ∈ T : I(t) = Ø}
// while there are more terms to disambiguate
do
{
    P' = P
   for each t ∈ P'          // for pending terms
   {
        bestSense = Ø
        maxValue = 0
        // for each possible interpretation of t
        for each sense S of t in WordNet
        {
                 f[S] = 0
                for each synset S' E I
                {
                 X = 0
// for each semantic path between S and S'
                X = X + weight(S, S')
                 f[S] = f[S] + X
                }
                 if (f[S] > maxValue)
                  {
                   maxValue = f[S]
                   bestSense = S
                  }
          }
                if(maxValue > 0)
                {
                  I(t) = bestSense
                  P = P \ {t}
                }
   }
}while(P ≠ P')

return I
```

Algorithm 2. Structural Semantic Interconnection

For example, the keyword *mouse* is referenced in WordNet [22] by the following four synsets:

• mouse#n#1: {mouse} (with the gloss *any of numerous small rodents typically resembling diminutives rats,...*).

• mouse#n#2: {shiner, black eye, mouse} (with the gloss *a swollen bruise caused by a blow to the eye*)

• mouse#n#3: {mouse} (with the gloss *a swollen bruise caused by a blow to the eye*)

• mouse#n#4: {mouse, computer mouse} (with the gloss *a hand operated electronic device* ...).

Given W, an ordered list of words to be disambiguated, the SSI algorithm performs as follows. During the initialization step, all monosemous words are included into the set I of already interpreted words, and the polysemous words are included in P (all of them pending to be disambiguated). At each step, the set I is used to disambiguate one word of P, selecting the word sense which is closer to the set I of already disambiguated words. Once a sense is disambiguated, the word sense is removed from P and included into I. The algorithm finishes when no more pending words remain in P. The pseudo code of the SSI algorithm is given in algorithm 1.

4.2.3 Jaccard matching

For matching the user request sense with a set of senses from web service description, the Jaccard matcher [23], is used. It computes the similarity between two sets and so can compare the different set of senses:

$$senseSim(ss_u, ss_w) = \frac{|ss_u \cap ss_w|}{|ss_u \cup ss_w|} \quad (1)$$

By dividing the number of senses which appear in both sets by the total number of senses in both sets, a similarity coefficient can be calculated. With this approach the Jaccard matcher retrieves the similar services.

4.3 Compound Similarity

Finally, the compound similarity of two services S_1 and S_2 is calculated by the linear combination of bipartite matching (functional similarity) and text-based similarity as follows:

$$Sim(S_1, S_2) = (FS(S_1, S_2) + TS(S_1, S_2))/2 \quad (2)$$

Where FS is the Functional Similarity and TS is the Text-based Similarity of two services S_1 and S_2.

5 Implementation

The proposed work is implemented using Java. It can search web services annotated using OWL-S. To read the OWL-S files which are basically XML

(eXtensible Markup Language) files, the DOM parser is used. For the Part-Of-Speech tagging, the Stanford parser [24] is used. The stanford-postagger.jar file is used for this purpose. For the Word Sense Disambiguation, WordNet [25] is used through two WordNet APIs, in order to find senses belonging to words. The Java WordNet Library (JWNL) [26] is used for the morphological analysis of each word, and the MIT Java WordNet Interface (JWI) [27] is subsequently employed for retrieving WordNet synsets. Next, JWordNetSim [28] is used to calculate the similarity between two WordNet senses.

The overall implementation is done in Java due to the availability many predefined external packages. The implementation of the framework is based on the work of [18], which mainly focuses on WSMO services. Also additionally, as part of our framework, the functional properties (IOPE) of services are also considered, which is matched by using Bipartite Graph matching algorithm. The Pellet reasoner [29] is used to classify the loaded ontologies. The Jena API (Application Programming Interface) [30] is used to query the reasoner for concept relationships.

5.1 Evaluation Setup

For discovery process, the atomic OWL-S services are selected from the OWLS-TCv4 collection [31], which is a publicly available collection of OWL-S services, used to evaluate and compare different matchmaking algorithms. It comprises 1083 services, which uses reference ontology with 4694 concepts from 9 different domains. The service IOPE parameters and service text-description are considered during the discovery process.

5.2 Experimental Results

To test the performance of the proposed framework, two other frameworks (our previous works) are implemented. The first framework performs only the non-functional parameter matching. It uses NLP techniques like stop word removal and POS tagging on the text description and uses the Jaccard matching for calculating the similarity [32]. The second framework performs only functional parameter matching. It performs functional matching of Input, Output, Precondition and Effects. It uses Bipartite Graph matching approach for this purpose. Our proposed framework performs both functional matching (Input, Output, Preconditions and Effects) and Non-functional parameter matching (POS tagging, Word Sense Disambiguation, Jaccard matching).

The results of our proposed framework will be displayed in the ranked order. Retrieved web services with rank value greater than 0.8 is considered to be highly relevant since some level of match occurs for all the IOPE parameters requested by user and the text descriptions also match. Services with rank value between 0.4 and 0.7 is considered relevant since some level of match occurs for all the IO parameters and there may not be a match for either precondition or effect requested by user. Services with rank value less than 0.4 are considered less relevant since there may not be a correct match for IOPE parameters. These services are matched based on their non-functional text-description only.

The performance of various discovery frameworks is measured based on the precision and recall rates. Precision and recall are the basic measures used in evaluating search strategies. It is possible to measure how well a search performed with respect to these two parameters. As shown in fig. 5 these measures assume:

- There is a set of services in the database which is relevant to the search topic
- Services are assumed to be either relevant or irrelevant
- The actual retrieval set may not perfectly match the set of relevant services.

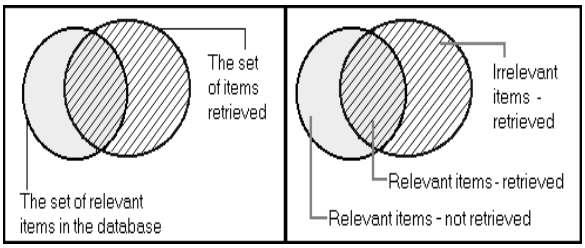
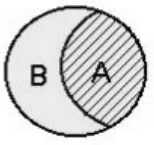


Fig. 5. Available services vs. Retrieved services

RECALL is the ratio of the number of relevant services retrieved to the total number of relevant services in the database as shown in fig. 6.

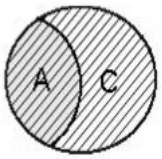


A: Number of relevant services retrieved
B: Number of relevant services not retrieved

$$\textbf{RECALL} = \frac{A}{A+B} \times 100\%$$

Fig. 6. Recall rate computation

PRECISION is the ratio of the number of relevant services retrieved to the total number of irrelevant and relevant services retrieved as shown in fig. 7.



A: Number of relevant services retrieved
C: Number of irrelevant services retrieved

$$\textbf{PRECISION} = \frac{A}{A+C} \times 100\%$$

Fig. 7. Precision rate computation

For the sample test collection used, there are about 164 services relevant to the user query. The number of relevant and irrelevant services retrieved by the various discovery approaches is shown in Table 1.

Table 1. Details of Service retrieval

SWS discovery frameworks	A	B	C
NFP matching	19	145	0
FP matching	24	140	0
Proposed matching	160	4	504*

* Including even the least relevant services

The performance of various approaches namely Simple text similarity of Non-functional parameter (NFP), Functional parameter (FP) matching and the proposed Hybrid matching (Functional Parameter (FP) and Non-functional Parameter (NFP)) matching based on the recall and precision rates is shown in table 2.

Table 2. Precision and Recall rates of various discovery frameworks

SWS discovery frameworks	Recall (in %)	Precision (in %)
NFP matching	11.59	100
FP matching	14.63	100
Proposed matching	97.57	24.1

Our proposed approach retrieves 160 highly relevant services. The precision rate of the proposed approach seems to be low because here the rest of the services are considered irrelevant (Though they are either relevant or least relevant). Hence, it is evident from the above table that the proposed hybrid approach for discovery of services based on both functional and non-functional parameters is efficient when compared to the other two approaches. The graphical representation of the performance of various discovery frameworks is illustrated in fig. 8.

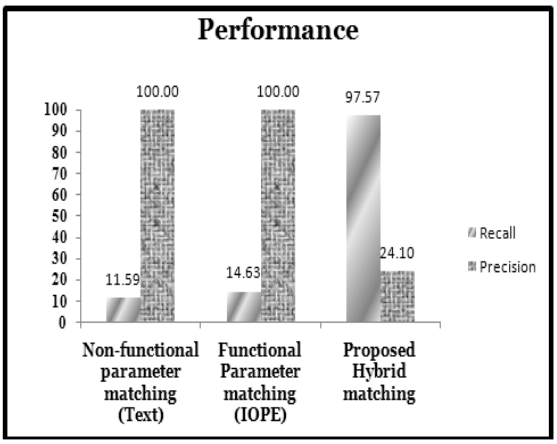


Fig. 8. Performance of various discovery frameworks

The retrieval of related web services based on the user query (which may be in either natural language or in the form of IOPE parameters or both) by our proposed hybrid framework is comparatively high than other frameworks.

6 Conclusion

The increasing number of web services on the web results in difficulty to discover the required web service. To overcome this difficulty and to discover the required Semantic web service, a new framework is developed. The Semantic web service discovery framework proposes a natural language based discovery process with matchmaking algorithms for searching web services that are described using semantically enriched annotations. In this work we have introduced a cumulative similarity (combination of functional similarity and their text similarities) of OWL-S annotated web services. Functional (IOPE) similarity is determined using bipartite matching of services. To measure textual similarity, POS tagging, Word Sense Disambiguation and Jaccard similarity is used. To summarize, our proposed framework presents a discovery mechanism that enables web service

discovery based on Input, Output, Preconditions and Effects parameters and/or keywords written in natural language. Also the experimental results show the effectiveness of our proposed approach.

7 Future Work

We have proposed hybrid approach for discovery of OWL-S services based on functional and non-functional parameters. It is one of the efficient methods for OWL-S service discovery, but with some limitations. Mainly the word sense disambiguation process takes a longer time. It is evident that the performance of service discovery will be significantly reduced when the number of services increases. But this discovery framework is promising because it has a better recall rate. Therefore, our future work will focus on reducing the search time by limiting the search space. This can be achieved by including a preprocessing (filtering - removing irrelevant services) stage before the actual discovery process [33]. To achieve maximum performance, this preprocessing can be performed in a distributed manner. The filtering followed by the actual discovery process can be performed simultaneously on different domains based on the user request. The independent results obtained can be finally combined to get the relevant services. Thus the spilt-and-merge technique [34] can be used to obtain the relevant services. It is expected that this technique will significantly reduce the search time and provide better results.

References:

[1] David Martin et al., "OWL-S Semantic Markup for Web services", *W3C Member Submission*, 22 November 2004, Available at < http://www.w3.org/Submission/OWL-S/>.

[2] Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P., "Semantic Matching of web service Capabilities", *Springer Verlag, LNCS, International Semantic Web Conference*, 2002, pp. 333 – 347.

[3] Lei Li and Ian Horrocks, "A software framework for matchmaking based on Semantic Web technology", *Proceedings of the 12th International Conference on WWW*, 2003, pp. 331-339.

[4] D. Bianchini, V. D. Antonellis, M. Melchiori, D. Salvi, "Semantic-enriched service discovery", *Proceedings of IEEE ICDE 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI06)*, Atlanta, USA, 2006.

[5] Dong-Wei, B., Chuan-Chang, L., Yong, P., & Jun-Liang, C., "Web services matchmaking with incremental semantic precision", *Wireless communications, networking and mobile computing*, 2006, pp. 1–4.

[6] Wang, H., & Li, Z., "A semantic matchmaking method of web services based on SHOIN+(D)", *Asia–Pacific conference on Services Computing*, 2006, pp. 26–33.

[7] S. Lamparter, A. Ankolekar, "Automated selection of configurable web services", 8. *Internationale Tagung Wirtschaftsinformatik. Universitaetsverlag Karlsruhe*, Germany, 2007

[8] Umesh Bellur, Roshan Kulkarni, "Improved Matchmaking Algorithm for Semantic web services based on Bipartite Graph Matching", *IEEE International Conference on Web Services*, 2007, pp. 86-93.

[9] H. Peng, Z. Shi, L. Chang, and W. Niu, "Improving Grade Match to Value Match for Semantic web service discovery" *The IEEE International Conference on Natural Computation (ICNC)*, IEEE Computer Society, Jinan, China, 2008, pp. 232 -236.

[10] Gongzhen Wang, Donghong Xu, Yong Qi, Di Hou, "A Semantic Match Algorithm for web services based on Improved Semantic Distance" IEEE, *4th International Conference on Next Generation Web Services Practices*, 2008.

[11] B. Bener, V. Ozadali, and E. S. llhan, "Semantic Matchmaker with Precondition and Effect Matching Using SWRL", *Expert Systems with Applications*, Vol. 36, Issue 5, 2009.

[12] M. Liu, Q. Gao, W. Shen, Q. Hao, and I. Van, "A Semantic-Augmented Multi-level Matching Model of web services", *Service Oriented Computing and Applications*, Vol. 3, Issue 3, 2009, pp. 205-215.

[13] K. Klusch M, Fries B, Sycara, "OWLS-MX: A Hybrid Semantic web service matchmaker for OWL-S services", *Journal of Web Semantics: Science, Services and Agents on the WWW*, Vol. 7, No. 2, 2009, pp. 121–133.

[14] Georgios Meditskos and Nick Bassiliades, "Structural and Role-Oriented web service discovery with Taxonomies in OWL-S", *IEEE Transaction on Knowledge and Data Engineering*, Vol. 22, No. 2, 2010, pp. 278 – 290.

[15] Golsa Heidary, Kamran Zamanifar, Naser Nematbakhsh, "A Three phase Semantic Web Matchmaker", *International Journal of Smart Home* Vol. 4, No.3, 2010.

[16] Yang Zhang, Fagui Liu, Nan Zhang, "Toward Fine Grained Matchmaking of Semantic web services based on Concept Similarity", *Journal of Information & Computational Science*. Vol. 8, No.2, 2011, pp. 377-384.

[17] M. Cai, W. Y. Zhang, & K. Zhang, "ManuHub: A Semantic Web System for Ontology-Based Service Management in Distributed Manufacturing Environments", *IEEE Transaction on Systems, Man & Cybernetics-Part A: Systems & Humans*, Vol. 41, No. 3, 2011.

[18] Jordy Sangers, Flavius Frasincar, Frederik Hogenboom , Vadim Chepegin, "Semantic web service discovery using natural language processing techniques", *Expert Systems with Applications*, 40, 2013, pp. 4660–4671.

[19] H. Kuhn, "The Hungarian Method for the Assignment Problem", *Naval Research Logistic Quarterly*, 1955.

[20] Rahul Jain, "Combinatorial Algorithms (Algorithms in Bipartite Graphs)", 2013, Available at <www.comp.nus.edu.sg/~rahul /.../cs6234-13-combinatorial-algorithm.ppt >.

[21] Navigli, R., and Velardi, P., "Structural semantic interconnections: A knowledge-based approach to word sense disambiguation", *IEEE Transactions on Pattern analysis and Machine intelligence*, Vol. 27, 2005, pp. 1075–1086.

[22] "WordNet – A Lexical database for English", Available at <http://wordnet.princeton.edu/>.

[23] "Jaccard, P.", 1901, Available at <http://people.revoledu.com/kardi/tutorial/Similarity /Jaccard.html>.

[24] "The Stanford natural language processing group. Stanford log-linear part-of-speech tagger", 2009, Available from <http://nlp.stanford.edu/software/tagger.shtml>

[25] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K., "Introduction to WordNet: An on-line lexical database", *International Journal of Lexicography*, 3(4), 1990, pp. 235–244.

[26] Walenz, B., & Didion, J., "JWNL: Java WordNet library", 2011, Available at <http:// sourceforge.net/projects/jwordnet/>.

[27] Finlayson, M., "JWI: The MIT java WordNet interface", 2012, Available at <http:// projects.csail.mit.edu/jwi/>.

[28] Greenwood, M., "JWordNetSim", 2007, Available at <http://nlp.shef.ac.uk/result/ software.html>.

[29] E. Sirin et al., "Pellet: An OWL DL Reasoner", Journal *of Web Semantics*. 2005.

[30] "JENA: Java Framework for Building Semantic Web Applications", Available at <http://jena.sourceforge.net/.>

[31] "OWL-S Service Retrieval Test Collection. Version 4.0", Available at <http://projects. semweb central.org /projects/ owls-tc/>.

[32] M. Deepa Lakshmi, Dr. Julia Punitha Malar Dhas, "An user-friendly and improved Semantic-based web service discovery approach using Natural Language Processing Techniques", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 1, Issue 10, December 2013, pp. 2435-2442.

[33] M. Deepa Lakshmi, Dr. Julia Punitha Malar Dhas, "An improved Light-weight matchmaking mechanism for discovering OWL-S services based on SPARQL, Bipartite and NLP approach", *Malaysian Journal of Science*, Vol. 33, No.1, June 2014, pp. 68-77.

[34] Szénási, S., "Distributed Implementations of Cell Nuclei Detection Algorithm", *Recent Advances in Image, Audio and Signal Processing, WSEAS Press*, Budapest, 2013, pp. 105-109.