Malaysian Journal of Science **33** (1): 68-77 (2014)

AN IMPROVED LIGHT-WEIGHT MATCHMAKING MECHANISM FOR DISCOVERING OWL-S SERVICES BASED ON SPARQL, BIPARTITE AND NLP APPROACH

M.Deepa Lakshmi*[1], Dr. Julia Punitha Malar Dhas[2]

[1]Research Scholar, N.I University, Kumaracoil, K.K.Dist., Tamil Nadu, India
[2]Professor & Head, Dept. of CSE, N.I University, Kumaracoil, K.K.Dist., Tamil Nadu, India
*deepasuresh12@gmail.com (Corresponding Author)

ABSTRACT Semantic Web services integrate the meaningful content of the Semantic Web with the business logic of Web services and thus enable industries and individuals to access these services. But as the number of available Web services increase, there is a growing demand for a mechanism for effective retrieval of required services. We propose an improved Semantic Web service discovery method for finding OWL-S (Web Ontology Language for Services) services by combining functional similarity matching (using bipartite graph) and textual similarity matching. However, discovering relevant Semantic Web service is a heavyweight task. Performance of service discovery is significantly reduced when the number of services increases. To overcome this issue, a lightweight filtering stage is also introduced before the discovery mechanism. Filtering is performed by semantic-based SPARQL (Simple Protocol and RDF Query Language) query. It will significantly reduce the input for the discovery process. Thus the search space and the time required to find the relevant services will be reduced. The proposed techniques are applied to a sample test collection and experimental results are presented, which demonstrate the effectiveness of the idea.

(Keywords: discovery, filtering, OWL-S, Semantic Web service, SPARQL)

INTRODUCTION

Today there exist a vast amount of tools that support and facilitate the development, deployment and invocation of Web services, and there are virtually no limits as to what Web services can do within their realm. With the rapid development of Web services, retrieval of relevant services has become a challenge. The keyword-based discovery mechanism using Universal Description, Discovery and Integration [1] and Web Service Description Language [2] is inefficient due to the retrieval of a large amount of irrelevant information. This is due to the lack of semantic information of the service. The Semantic Web initiative addresses this problem by creating a set of XML (eXtensible Markup Language) based documents and ontology. Semantic Web services integrate the meaningful content of the Semantic Web with the business logic of Web services and thus enable all to access these services. But as the number of available Web services increase, there is a growing demand for a mechanism for effective retrieval of required services.An important component of the discovery process is the matchmaking algorithm. In order to overcome the limitations of existing syntax-based search, match-making algorithms based on semantic techniques have been proposed. Most of them are based on an algorithm originally proposed by Paolucci [3]. This is a logic-based semantic matchmaking approach and so has certain limitations. At this juncture, improving the existing discovery mechanism constitutes a vital step. Therefore, we propose an improved Semantic Web service discovery method by modifying the Paolucci's algorithm using bipartite matching of Input / Output (I/O) parameters. In addition, one of the crucial steps in an efficient Web service search is to understand what users mean in their request. The search request is usually in the form of natural language. Most of the current popular discovery algorithms expect that the user request (Query) be given only as an OWL-S query. But, this work searches through the set of Web services for matches with a user query, which consists of just keywords, so that knowledge about semantic languages is not required. A text-based similarity match is performed on the description of Web service.

Several discovery techniques are already available to discover services. But, the ability to deal with a large search space of available services is a major issue that is not addressed in these discovery techniques. In order to overcome this problem, there are some proposals that provide different techniques to improve the discovery performance, such as indexing or caching

descriptions [4], using several matchmaking stages and hybrid approaches [5] that include non-semantic techniques. In our work, this issue is addressed by filtering services using SPARQL query.

Services can be described using OWL-S, WSMO (Web Service Modeling Ontology), SAWSDL (Semantic Annotations for WSDL and XML Schema), WSMO-Lite which defines the features and functionality of services in terms of input, output parameters, and non-functional aspects. In this work, OWL-S service [6] descriptions are considered. The following implementations are done using Java as part of our work.

- Preprocessing:-
  - Exact Filtering [7] (Existing System)
  - Semantic-based Filtering (Proposed System)
- Matchmaking:-
  - Paolucci's algorithm (Existing System)
  - Bipartite matching algorithm [8] (Existing System)
  - Bipartite with text-based matching (Proposed System)
- Proposed bipartite with text-based matching after performing semantic-based filtering

Using OWL-S TC2 (Test Collection version 2), we tested our algorithm and compared it with existing approaches. The experimental results have shown that the proposed approach could discover the most relevant advertised services corresponding to user's request and provide better recall rate and acceptable precision compared to other approaches.

The rest of the paper is organized as follows: Section II highlights some related works. Section III explains major existing approaches. Section IV presents an overview of proposed approaches. Our evaluations are presented in Section V. Finally, in Section VI we present the conclusion.

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 [3], 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 [12] used a DAML-S based ontology and a Description Logic 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 [13] performs a combined logic-based and text similarity-based matching of monolithic service and query concepts written in OWL-DL. Lamparter [14] presents an approach to hybrid matching of monolithic logic-based service descriptions in OWL-DL extended with pricing policies (modeled in DL-safe SWRL rules) according to given references by means of SPARQL 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 [15] 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 [16] work proposes a semantic match algorithm based on improved semantic distance. Bener et al. [17] considers semantic matching of input, output, precondition and effect.

They also provide ranking. Liu et al. [18] achieve a fusion with five grades of matching, a collaboration of syntactic and semantic matching, as well as considering QoS and other dependency features. The OWLS-MX [19] matchmaker performs hybrid semantic matching that complements logic based reasoning with syntactic IR based similarity metrics. OWL-SLR [20] 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 [21], 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. [22] 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 [23] 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.

All the above mentioned work 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 the matching process is a heavy weight task, which needs more time for performing discovery. In order to reduce search time, Maria et al. [7] proposed a light-weight preprocessing before the actual discovery process. Though this approach reduces search time considerably, many relevant services are filtered out during the filtering process. So our work proposes a hybrid matchmaking algorithm which accepts user input even in natural language and also enhances the work of Maria so that the recall rate is improved and also the search time becomes minimal compared to existing approaches.

EXISTING SYSTEM

A. Paolucci Algorithm

This section describes the algorithm proposed by Paolucci. It is based on semantic matchmaking of input and output terms of OWL-S services. It uses a greedy approach for matchmaking. It takes an OWL-S *Query* from the user as input and iterates over every OWL-S *Advertisement* in the repository in order to determine a match. An *Advertisement* and a *Query* match if their outputs and inputs both match. The algorithm returns a set of matching advertisements sorted according to the degree of match.For semantic matchmaking, an advertisement *Advt* and query *Query* match if

- For every output parameter in *Query*, there is one output parameter in *Advt*. Let $Query_{out}$ and $Advt_{out}$ represent the list of output concepts of query and advertisement respectively. Matching of outputs exist if

$$\forall c \in Query_{out}, \exists d \in Advt_{out}, s.t. match(c,d) \neq Fail \quad (1)$$

- For every input parameter in *Advt*, there is one input parameter in *Query*. Let $Query_{in}$ and $Advt_{in}$ represent the list of input concepts of query and advertisement respectively. Matching of inputs exist if

$$\forall c \in Advt_{in}, \exists d \in Query_{in}, s.t. match(c,d) \neq Fail \quad (2)$$

Suppose outQ ϵ $Query_{out}$ and outA ϵ $Advt_{out}$ are two concepts, in case of output matching, the match(outQ, outA) function accepts outQ and outA as parameters and returns the degree of match between them. Four degrees of match are defined between them:

- ***Exact***: If outA is an equivalent concept to outQ or outA is a superclass of outQ.
- ***Plugin***: If outA Subsumes outQ.
- ***Subsume***: If outQ Subsumes outA.
- ***Fail***: If none of the above conditions are satisfied.

These four degrees are ranked as: *Exact* > *Plugin* >*Subsumes* > *Fail*.

B. Bipartite graph-based matching

Paolucci's algorithm has drawbacks like it is dependent on the order in which concepts are defined in Query. Therefore, this algorithm may generate false positive or false negative results. To solve this problem, another approach which makes use of bipartite graph matching is introduced. This algorithm introduces a different set of rules for match between concepts, in which Plugin and Subsume levels are interchanged in their degree of match. The assumption of Paolucci that if an advertiser advertises a concept, it would provide all the immediate subtypes of that concept is dropped. Hence, if the query concept is subsumed by the advertisement concept a Subsume match is returned and if the query concept subsumes the advertisement concept, a Plugin match is returned. Plugin is still ranked higher than a Subsume match. The algorithm for matchmaking of output parameters is given below:

Algorithm: PROCEDURE match (outA, outQ)

1: **if** outA = outQ **then**
2: return Exact
3: **else if** outQ superclass of outA **then**
4: return Plugin
5: **else if** outQ subsumes outA **then**
6: return Plugin
7: **else if** outA subsumes outQ **then**
8: return Subsumes
9: **else**
10: return Fail
11: **end if**

A. Exact Filtering

Discovering relevant service is a heavyweight task. Performance of service discovery is significantly reduced when the number of services increases. To overcome this issue, a lightweight process is introduced before discovery mechanism. This process analyses user request in order to extract concepts. Then the service repository is filtered based on the concepts by generating SPARQL queries. The architecture of the existing system is given in Fig. 1.

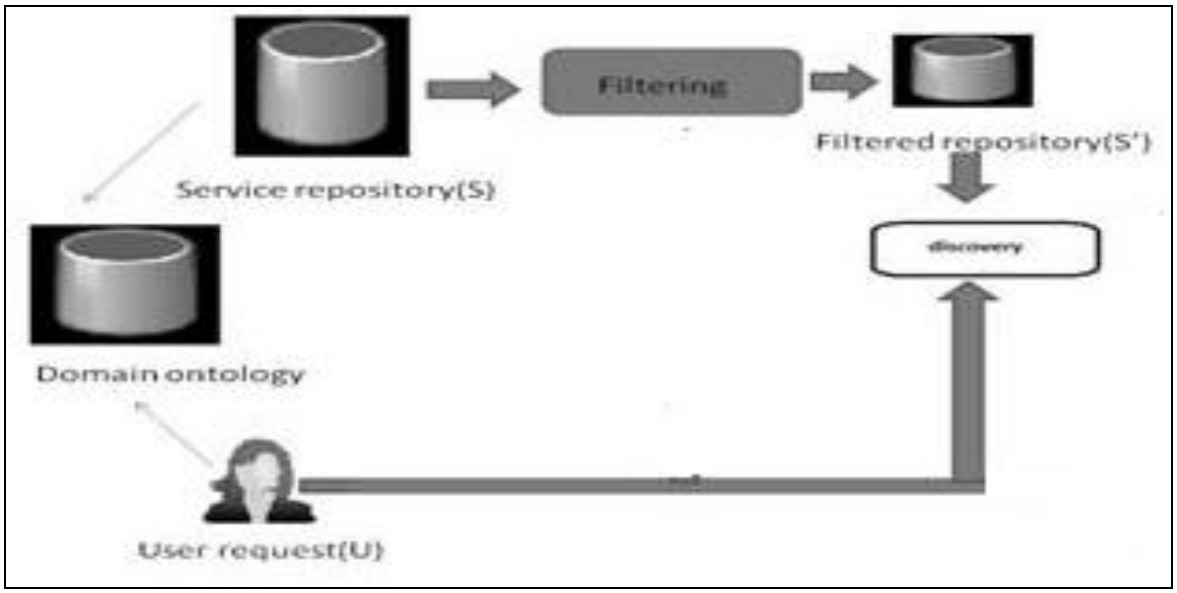

Figure 1. Exact filtering before discovery

Filtering is performed by two SPARQL queries such as Q_{all} and $Q_{some.}$ Q_{all} returns services whose definitions contain *all* the concepts referred by a user request. Q_{some} returns services whose definitions contain some (at least one) concepts referred by a user request, assuming that those services may satisfy its requirements and/or preferences to some extent, despite the missing information.

Given a concrete user request defined using an existing Semantic Web service framework, both filters can be instantiated as SPARQL queries that select services from a service repository, which contains descriptions based on the same Semantic Web service framework.

Example:
Consider a user is searching for services which provide scholarship by government. Here, the input terms are government, academic degree and the output term is scholarship.

U={(I*nputTerm*$_{u1}$,{**Government**}),
(*InputTerm*$_{u2}$,{**Academic degree**}),
(O*utputTerm*$_{u3,}${**Scholarship**)}

Now, a SPARQL query is generated to perform filtering. The generated SPARQL query (Q_{some}) is as follows.

LISTING
SAMPLE Q_{SOME} QUERY

```
select distinct  service1
where {
service: service1
service1
profile:hasInput GOVERNMENT
UNION
service1
profile:hasInput ACADEMICDEGREE
UNION
service1
profile:hasOutput SCHOLARSHIP}
```

In order to reduce the search space, the above query is applied. In exact filtering some relevant services get discarded when filtering is done prior to discovery process. This paves the way for the introduction of semantic-based filtering mechanism.

PROPOSED SYSTEM

Our proposed approach finds relevant services for user's request based on the functional and text-based similarity. This approach is based on the architecture shown in Fig. 2.

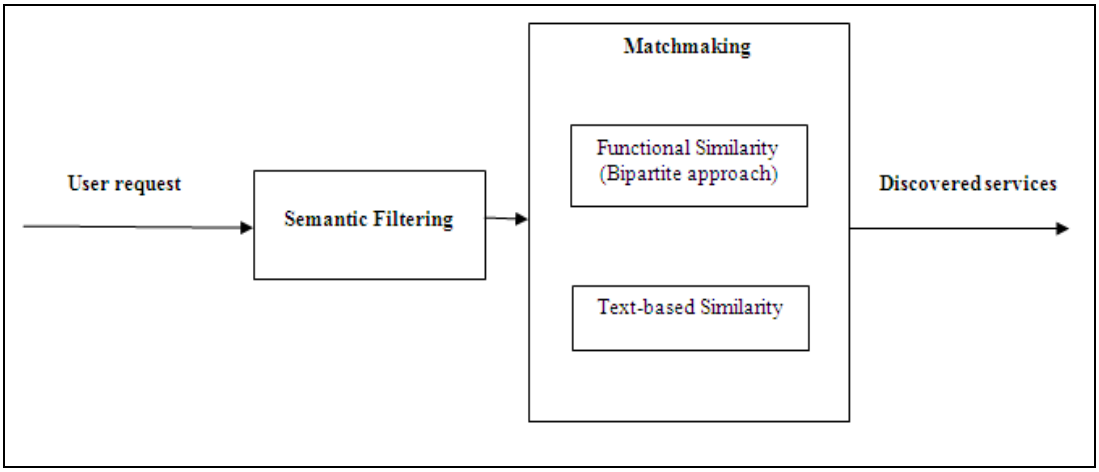

Figure 2. Proposed system architecture

In the proposed approach, we assume that there is a set of Web services described and published in OWL-S. In order to address one of the major limitations of existing approaches, which expects user request to be specified as an OWL-S query, our discovery framework permits user query to be expressed in natural language. Our system offers a simple graphical user-friendly interface for service requester to input query in natural language. It also provides facilities for user to enter input and output parameters of the needed service.Also, one major difficulty that arises with most of the discovery techniques is that as search space increases, the discovery or search time increases proportionally. This problem can be overcome by applying a filtering process before the actual discovery process. A semantic-based filtering is proposed in this work.

A. Semantic-based Filtering

It is better to search all the related words rather than search merely for the given word. That is, finding the meaning is much more important so that the data mined can be more relevant to the user demand. The meaning/related words of the given keyword are obtained using WordNet3.0. The retrieved words are then analyzed further to get more relevant answers. Each word is analyzed further using the WorldNet tool to get related words. The process is repeated for each word found. This can help to get more relevant items and paves way for effective discovery of relevant services.

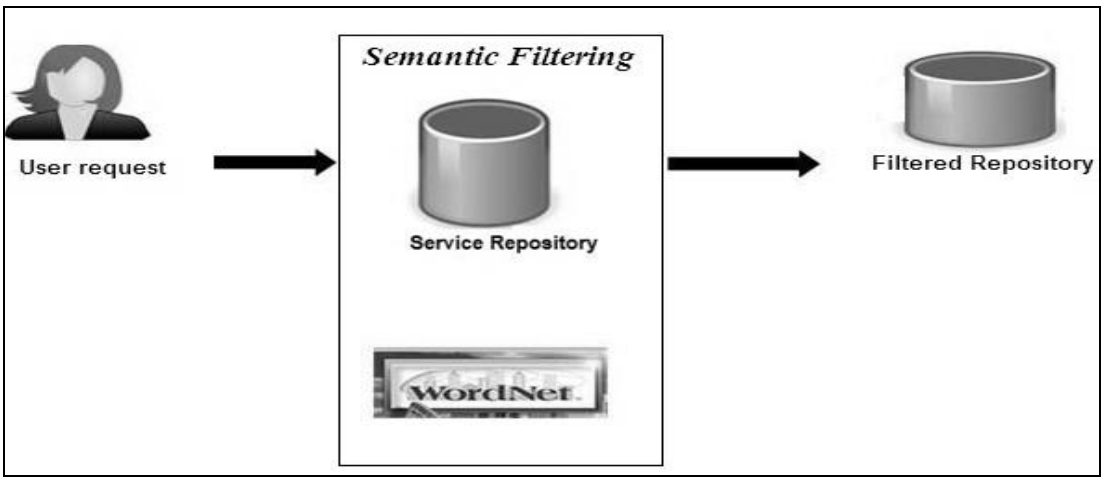

Figure 3. Semantic filtering before discovery

User request is modeled as an OWL-S construct. Similar words found are also included in the input and output of the OWL-S construct. Based on this OWL-S profile containing keywords as well as similar words, the SPARQL query Q_{some} is formed. Fig. 3. illustrates the process of semantic-based filtering performed before discovery. The number of related services discarded during filtering is reduced by this process. Now, this filtered repository can be used during the matchmaking process, which is explained below.

B. Bipartite with Text-based matching

The matchmaking process involves two steps namely: performing functional similarity (using bipartite graph-based matching) based on input and output parameters of user request and available OWL-S services and textual similarity (using Jaccard similarity) based on user request in natural language and text description of OWL-S services.

The functional similarity using bipartite graph-based approach is computed as discussed in previous section. The textual similarity is computed as follows. OWL-S service provides specification description using <profile:textDescription> tag for a web service. Usually, a textual description of a service provides a brief functional description of what it is. The following NLP (Natural Language Processing) steps are carried out for computing textual similarity:

Term Vector representation

Text description of each service is represented as a term vector, where each component represents the frequency of a word in the description. Using this model, services can be represented as vectors s_k = $(w_{1k}, w_{2k}, \ldots, w_{nk})$, where w_{ik} is a weight for term i in the description of service s_k. Terms are extracted from the <profile:textDescription> tag of OWL-S service description. The user query is also represented as a term vector.

Stop word removal

Stop word removal aims to reduce words which act poorly as index terms. For example, those words can be "a", "the", "and" etc. An external stop word list is used to filter out those words.

Stemming

Stemming is a process to replace words with their root / stem forms by removing suffixes or prefixes. Words such as: *intersected*, *intersecting*, *intersection* is stemmed as *intersect*. This process reduces not only the variety of words, but also computational cost. The Porter Stemming Algorithm [24] is used to conduct stemming process.

Vector Enhancement

The modified term vectors are further enriched using WordNet and SUMO (Suggested Upper Merged Ontology). This process appends relevant ontology concepts and deletes irrelevant terms from the term vector based on the ranking of semantic relationships among the terms.

After performing the above steps, Jaccard similarity [25] measure is used to calculate the similarity between Query term vector and Advertisement term vectors. Based on this measure, the retrieved services are sorted. The text similarity is calculated as follows:

$$Text_{sim}(Advt, Query) = Advt_T \cup Query_T \div [(Advt_T + Query_T) - (Advt_T \cup Query_T)] \quad (3)$$

Finally, the proposed bipartite with text-based matching approach computes the compound similarity of two services, say S_1 and S_2 as a linear combination of functional similarity and text-based similarity as follows:

$$Sim(S_1, S_2) = [FS(S_1, S_2) + TS(S_1, S_2)] \div 2 \quad (4)$$

To summarize, our proposed work performs a preprocessing step (semantic filtering) before the actual discovery process so that it can retrieve relevant services in a limited search time by accepting user request even in natural language. This proposed approach is highlighted below.

C. Proposed Bipartite with text-based matching after performing Semantic-based Filtering

In this approach, the service request is obtained from user and the initial service repository is filtered using "Semantic-based Filtering" approach. Thus the search space will be reduced considerably. Next, the "Bipartite with Text-based" approach is used to match the Input / Output parameters and the text descriptions of the Query and the Advertised services. The result will be a set of required services relevant to the user request and also within a limited time.

IMPLEMENTATION AND EXPERIMENTAL RESULTS

For semantic matchmaking, OWL-S services are selected from OWLS-TC2 [26], which is a publicly available collection of OWL-S services used to evaluate and compare different matchmaking algorithms. It comprises 1007 services, which uses reference ontology with 4694 concepts from seven different domains. Pellet reasoner [27] is used to classify the loaded ontologies. Jena API is used for reasoning concept relationships. The Input / Output parameters and text-description are considered for the matchmaking process.

For example, our algorithms were tested with a user query for Hospiltal_Investigating_Service which has one input and one output parameter namely Hospital and Investigating respectively and a text description as "Returns investigating of Hospital"

Experimental Results:

When the search time is computed for the above algorithms, it is seen that as the search space increases, execution time increases proportionally. This is illustrated in Fig. 4.It is evident that this will be an issue if the number of advertised services increases. This problem can be overcome by applying a filtering process before the actual discovery process.

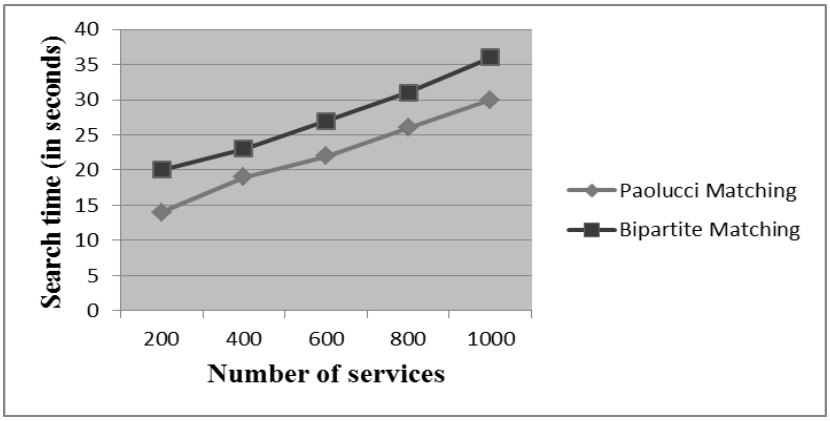

Figure 4. Search time (Without Filtering)

When filtering is applied before the actual discovery process, the search space is reduced considerably. The following TABLE 1 illustrates the reduction of the initial search space which consists of 1007 services to just 35 and 53 services by applying the existing exact filtering method and proposed Semantic filtering method respectively.

Table 1. relevant services retrieved by exact and semantic filtering

Filtering technique	No. of relevant services	No. of retrieved relevant services
Exact filtering	55	35
Semantic filtering	55	53

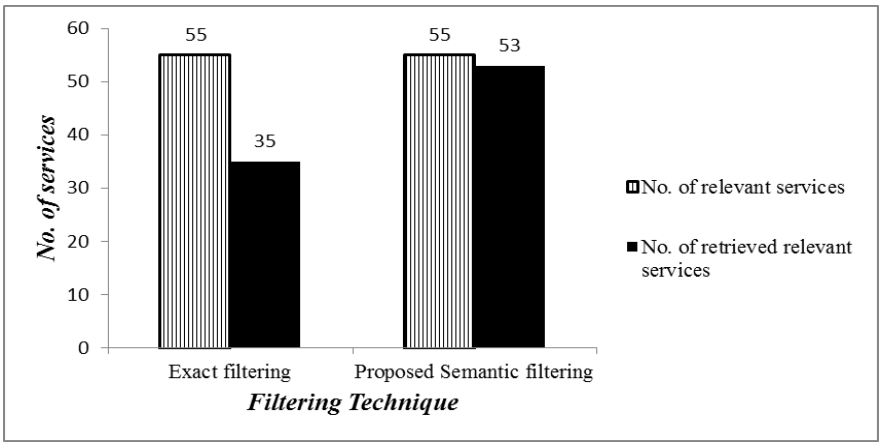

Figure 5. Performance of exact and semantic based filtering

For the experimented query there are totally 55 relevant services in the test collection. By applying exact filtering 35 relevant services were only retrieved and by applying the proposed semantic filtering 53 relevant services were retrieved. The results show that the semantic filtering has considerably high recall rate than exact filtering (Fig. 5).

This reduced search space can now be used with our proposed "Bipartite with text-based" matching algorithm. The efficiency of this combined "Semantic Filtering and Bipartite with Text-based" matching in terms of search time is illustrated in the following TABLE 2.

Table 2. Search Time

Algorithm	No. of services to be searched	Search time (in sec.)
Proposed Bipartite with text-based matching (without filtering)	1007	43
Proposed Bipartite with text-based matching (with semantic filtering)	55	14

A graphical chart illustrating the performance of the proposed approach without filtering and with filtering in terms of search time is shown in Fig. 6.

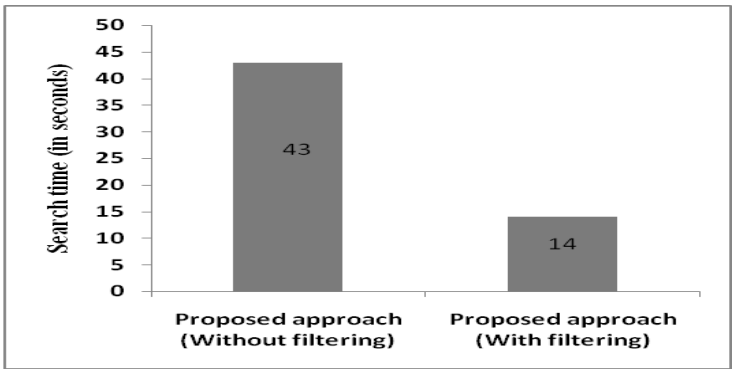

Figure 6. Performance of matchmaking algorithm in terms of Search time

Our proposed bipartite with text-based matching with semantic filtering approach retrieves 8 more services compared to existing bipartite matching, out of which 3 services are related to the user request. Additional services retrieved are: CheckHospitalAvailability, CheckPersonnelAvailability, Select Other Hospital, **Biopsy Availability,** InformHospital, **ODGCService, ODGService,** PatientTransport. All the highlighted services are related to the user request. The precision rate of the various matchmaking algorithms are presented in TABLE 3 and their graphical representation is shown in Fig. 7.

Table 3. Precision rates

Method	Precision
Paolucci Matching	48.6%
Bipartite Matching	80%
Proposed Bipartite with text-based similarity Matching	76%

Our proposed system provides the following advantages: A natural language interface for ease of use, considerable reduction in search time by applying filtering and reasonable precision. The experimental results show the effectiveness of our proposed approach when compared to existing discovery techniques

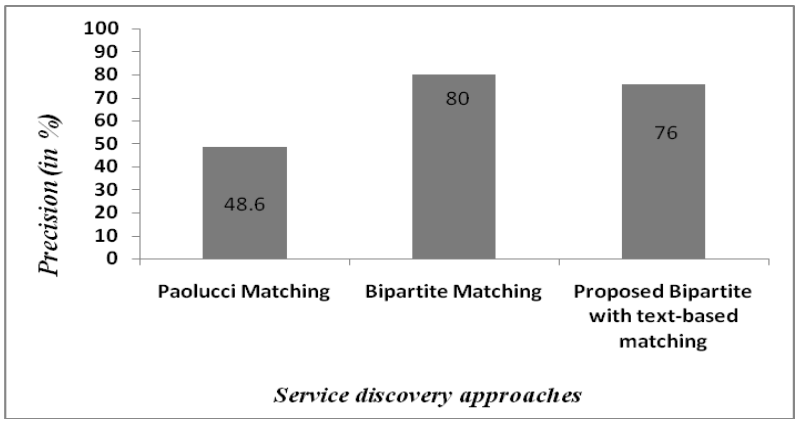

Figure 7. Precision rates of the three algorithms

CONCLUSION

In this work we have introduced a Compound Similarity (combination of functional similarity and their text similarities) of OWL-S annotated web services. Functional similarity is determined using bipartite matching of I/O parameters of services. To measure textual similarity, we utilize Jaccard similarity. Increased search time is a major problem in discovery of services.

In order to overcome this issue, the search space to be used during the discovery process has to be reduced. This is done by applying a semantic-based SPARQL filter before the actual discovery process. The proposed bipartite and text-based similarity using semantic filtering approach retrieves most relevant services related to user request, which can be specified in natural language. The experimental results show the effectiveness of our proposed approach.

REFERENCES

1. http://uddi.org/.
2. http://www.w3.org/TR/wsdl.
3. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P. (2002). Semantic Matching of Web service Capabilities. *Springer Verlag, LNCS, International Semantic Web Conference*, pp. 333 – 347.
4. Hepp M, Hoffman J, Stollberg M. (2007). A Caching Mechanism for Semantic Web service discovery. K. Aberer, et al. (Eds.), *ISWC/ASWC, Vol. 4825 of LNCS, Springer*, pp. 480–493.
5. Kaufer F, Klusch M. (2009). WSMO-MX: A hybrid Semantic Web service matchmaker. *Web Intelligence and Agent Systems*.
6. http://en.wikipedia.org/wiki/OWL-S
7. Antonio Ruiz-Cortes, David Ruiz, Jose Maria Garcia . (2012). Improving Semantic Web services discovery using SPARQL-Based Repository Filtering, *Journal of Web Semantics*.
8. Umesh Bellur, Roshan Kulkarni. (2007). Improved Matchmaking Algorithm for Semantic Web services Based on Bipartite Graph Matching. *IEEE International Conference on Web Services*, pp. 86-93.
9. G. Antoniou et al. (2003) Web Ontology Language: OWL. *Handbook on Ontologies in Information Systems*.
10. http://wordnet.princeton.edu/
11. JENA: Java Framework for Building Semantic Web Applications. http://jena.sourceforge.net/
12. Lei Li and Ian Horrocks. (2003). A software framework for matchmaking based on Semantic Web technology. *Proceedings of the 12th Int. Conf. on WWW*, pp. 331-339 , ACM New York
13. D. Bianchini, V. D. Antonellis, M. Melchiori, D. Salvi. (2006). Semantic-enriched service discovery. *Proceedings of IEEE ICDE 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI06)*, Atlanta, USA.
14. S. Lamparter, A. Ankolekar. (2007). Automated selection of configurable Web services. 8. *Internationale Tagung Wirtschaftsinformatik. Universitaetsverlag Karlsruhe*, Germany.
15. H. Peng, Z. Shi, L. Chang, and W. Niu. (2008). Improving Grade Match to Value Match for Semantic Web service discovery. *The IEEE International Conference on Natural Computation (ICNC)*, IEEE Computer Society, Jinan, China, pp. 232 -236.
16. Gongzhen Wang, Donghong Xu, Yong Qi, Di Hou. (2008). A Semantic Match Algorithm for Web services based on Improved Semantic Distance. IEEE, *4th International Conference on Next Generation Web Services Practices*.
17. A. B. Bener, V. Ozadali, and E. S. llhan. (2009). Semantic Matchmaker with Precondition and Effect Matching Using SWRL. *Expert Systems with Applications*, vol. 3 6, issue 5.
18. M. Liu, Q. Gao, W. Shen, Q. Hao, and I. Van. (2009). A Semantic-Augmented Multi-level Matching Model of Web services," *Service Oriented Computing and Applications*, vol. 3, issue 3, pp. 2 05-215.
19. K. Klusch M, Fries B, Sycara. (2009). OWLS-MX: A Hybrid Semantic Web service matchmaker for OWL-S services. *Journal of Web Semantics: Science, Services and Agents on the WWW*; 7(2), pp. 121–133.
20. Georgios Meditskos and Nick Bassiliades. (2010). Structural and Role-Oriented Web service discovery with Taxonomies in OWL-S. *IEEE Transaction on knowledge and data engineering*, vol. 22, no. 2, pp. 278 – 290.
21. Golsa Heidary, Kamran Zamanifar, Naser Nematbakhsh. (2010). A Three phase Semantic Web Matchmaker. *International Journal of Smart Home* Vol. 4, No.3.
22. Yang Zhang, Fagui Liu, Nan Zhang. (2011). Toward Fine Grained Matchmaking of Semantic Web services based on Concept Similarity. *Journal of Information & Computational Science*. 8: 2. pp. 377-384.
23. M. Cai, W. Y. Zhang, and K. Zhang. (2011). ManuHub: A Semantic Web System for Ontology-Based Service Management in Distributed Manufacturing Environments, *IEEE Trans. on Systems, Man & Cybernetics-Part A: Systems & Humans*, Vol. 41, No. 3.
24. http://ccl.pku.edu.cn/doubtfire/NLP/Lexical_Analysis/Word_Lemmatization/Porter/ Porter Stemming Algorithm.htm
25. trac.research.cc.gatech.edu/ccl/export/184/SecondMindProject/SM/SM.WordNet/Paper/WordNetDotNet_Semantic_Similarity.pdf
26. OWL-S Service Retrieval Test Collection. Version 2.1. http://projects.semwebcentral.org/projects/owls-tc/.
27. E. Sirin et al. (2005). Pellet: An OWL DL Reasoner. *Journal of Web Semantics*.