International Journal of scientific research and management (IJSRM)
||Volume||1||Issue|| 3 ||Pages|| 137-144 ||2013||
Website: www.ijsrm.in ISSN (e): 2321-3418

AN ENHANCED SPARQL - BASED FILTERING AND RANKING OF SEMANTIC WEB SERVICES

Jenifer J, M. Deepa Lakshmi, Dr.Julia Punitha Malar Dhas,

Student, M.E. (C.S.E),
PSN College of Engineering and Technology,
Tirunelveli, Tamil Nadu, India
jeniferjosepha@gmail.com.

Research Scholar,
N.I University, Kumaracoil, Tamil Nadu, India
deepasuresh12@gmail.com

HOD/CSE,
N.I.University, Kumaracoil, Tamil Nadu, India
julaps113@yahoo.com

Abstract---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 scalability issue, a lightweight process is introduced before the discovery mechanism. This process analyses the user request in order to extract the concepts. Then the service repository is filtered based on the concepts by generating SPARQL queries. The unrelated services are discarded during filtering. This filtering will fairly reduce the input for the discovery process. To avoid discarding relevant services during exact filtering, semantic filtering is performed. During this filtering similar words are found using Word Net. Ontology tree is created for the similar words found, from which the relation between the words are found more clearly. These similar words are also included in the automatically generated SPARQL queries. Seven degrees of matching are possible based on the obtained ontology. Based on these degrees of match the services are ordered and stored in the filtered repository. This can provide better efficiency in mining relevant data from the service repository than exact keyword based filtering. Thus an initial set of relevant services are found before the discovery technique which in turn will improve the performance of the matchmaking process.

Keywords: Semantic web services; ontology; SPARQL query; scalability

I. INTRODUCTION

Web services are self-describing, internet-based and platform-independent application components published using standard interface description languages and universally available via standard communication protocols. Web service discovery is the process of finding suitable web services for a given user request. Nowadays, there are huge amount of web services on the Web, which raises a serious problem during search. Several approaches have been proposed for adding semantics to Web service descriptions, including OWL-S, WSDL-S, and WSMO.
Several discovery techniques [1], [7] are available to discover the semantic web services. The following characteristics are considered for judging the computational reliability of discovery techniques. *Efficiency* - as the time required for finding a suitable Web service, *scalability*- as the ability to deal with a large search space of available Web services, and *stability* -as a low variance of the execution time of several invocations.
Current semantic web service discovery techniques do not satisfy all the above characteristics and are also not efficient to handle the large and complex services. Scalability problem is considered in this work. In order to overcome the scalability problem on semantic discovery mechanisms, there are some proposals that provide different techniques to improve the discovery performance, such as indexing or caching descriptions [6], using several matchmaking stages and hybrid approaches [8] that

include non-semantic techniques. In this work, services are filtered using SPARQL queries.

II. BACKGROUND

Semantic Web services serve as foundation tool for discovering and ranking services. Services can be described using OWL-S, WSMO, SAWSDL, WSMO-Lite [3] which defines the features, functionality of the services in terms of input, output parameters, and non-functional aspects. In this work OWL-S service [5], [9] descriptions are considered.

```
<profile:Profile rdf:ID="ACADEMIC-
DEGREEGOVERNMENT_SCHOLARSHIP_PROFILE">
<service:isPresentedBy rdf:resource="#ACADEMIC-
DEGREEGOVERNMENT_SCHOLARSHIP_SERVICE"/>
<profile:serviceName xml:lang="en">
GovernmentAcademicDegreeScholarshipService
</profile:serviceName>
<profile:textDescription xml:lang="en">
It is an attractive service to know about the scholarship offered for
the academic degree by the given government.
</profile:textDescription>
<profile:hasInput  rdf:resource="#_GOVERNMENT"/>
<profile:hasInput  rdf:resource="#_ACADEMIC-DEGREE"/>
<profile:hasOutput rdf:resource="#_SCHOLARSHIP"/>
<profile:has_process rdf:resource="ACADEMIC-
DEGREEGOVERNMENT_SCHOLARSHIP_PROCESS"/>
</profile:Profile>
```

Listing 1: OWL-S service profile example

OWL-S service description allows for the description of web service in terms of a *profile*, which tells “what the service does”, a *process model*, which tells “how the service works”, and a *grounding*, which tells how to access the service. Service profiles describe the service functionality in terms of inputs, outputs, preconditions and results. Listing 1 describes the service profile for the service which returns the scholarship offered for the academic degree by the given government.

Service descriptions define the functional and non-functional properties of services using concepts from the domain ontologies. For example the service description of the academic_degree_government_scholarship_service will contain Input/Output terms (functional properties) that refer to concepts like government, academic degree or scholarship for instance

2.1. Querying Semantic web services

For querying semantic web services, three approaches are available. They are graph based, rule based, and DL based query languages. Graph based query languages fetch RDF triples based on matching triple patterns with RDF graphs. Rule based query languages provide logical rules to define queries. DL based query languages are used to query Description Logics ontologies described in OWL-DL.

There are several graph based query languages but SPARQL [10],[4] (Simple Protocol And RDF Query Language) is the only W3C recommended language. SPARQL [2] defines standard query language and data access protocol for use with RDF data model. It works for any data source that can be mapped to RDF. For querying the service repositories SPARQL has four different types of queries such as SELECT, CONSTRUCT, DESCRIBE and ASK. SPARQL has facilities to:

1. Extract RDF sub graphs
2. Construct a new RDF graph using data from the input RDF graph queried
3. Return ‘‘descriptions’’ of the resources matching a query part
4. Specify optional triple or graph query patterns.

The general format of a SPARQL query is:

- PREFIX-Specification of a name for a URI (like RDQL’s USING)
- SELECT-Returns all or some of the variables bound in the WHERE clause

- CONSTRUCT-Returns a RDF graph with all or some of the variable bindings
- DESCRIBE-Returns a “description” of the resources found
- ASK-Returns whether a query pattern matches or not
- WHERE-list, i.e., conjunction of query (triple or graph) patterns
- OPTIONAL-list, i.e., conjunction of optional (triple or graph) patterns
- AND-Boolean expression (the filter to be applied to the result)

III. EXISTING SYSTEM

3.1 Filtering

To overcome the scalability issues the services are filtered before the discovery mechanism as in figure 1. Filtering is performed by two SPARQL queries such as Q_{all} and $Q_{some.}$

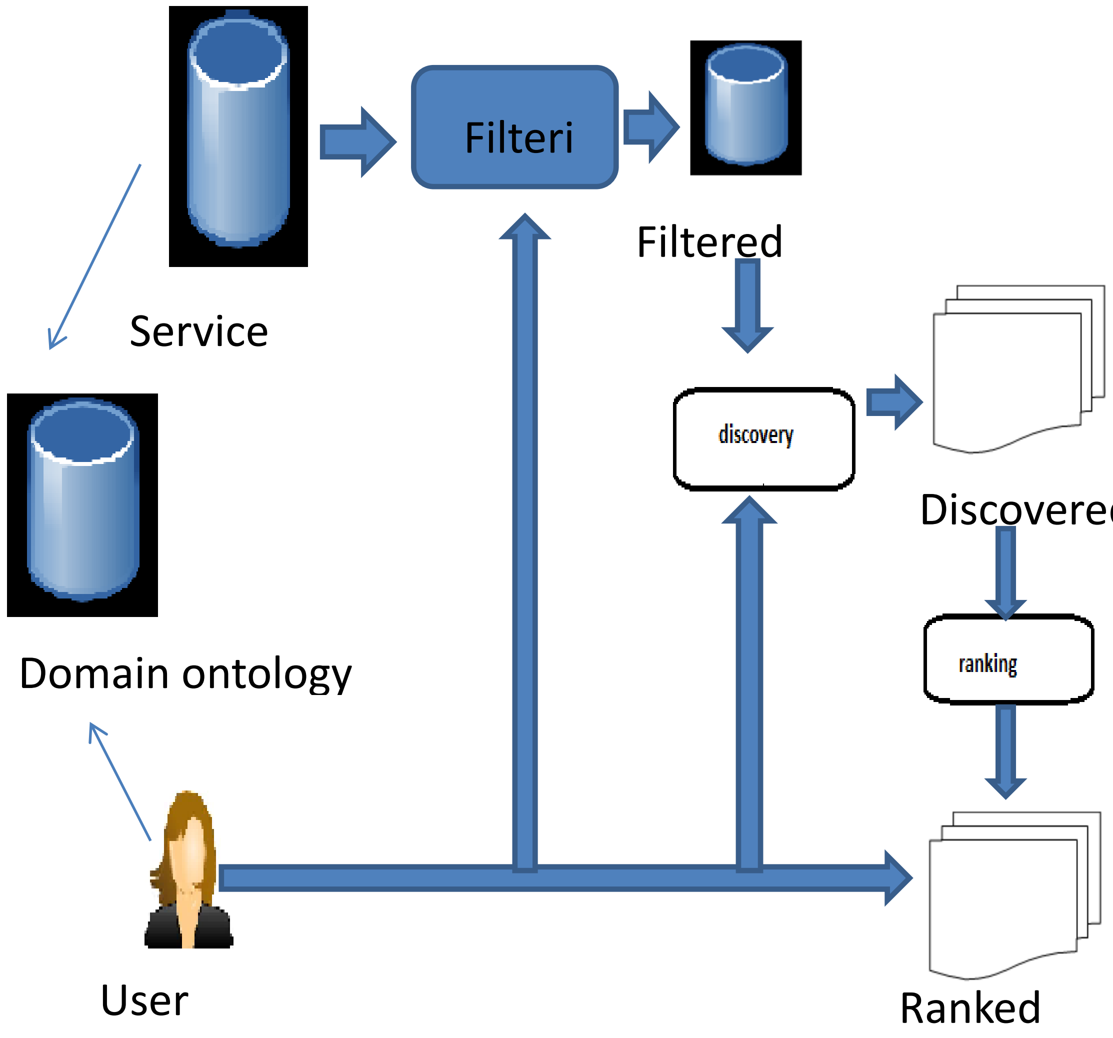

Figure 1: Service discovery including filtering

Q_{all} returns only the services whose definitions contain *all* the concepts referred by a user request. It assumes that services have to fulfill every term of the request in order to be useful for the user. Q_{some} selects service definitions that refer to *some* (at least one) of the concepts referred by a user request, assuming that those services may satisfy its requirements.

PROPOSED SYSTEM

In the available exact filtering mechanism some of the relevant services are discarded. To overcome this problem, filtering is performed semantically as in Figure2. Semantic means the meaning of the words. Finding the meaning is much more important so that the data mined can be more relevant to the user demand.

For example consider that a user is searching for a common keyword "vehicle", however he may be searching for any vehicle, may it be a car, motorbike, a cycle, truck or anything. So it is better to search all the related words rather than searching merely for the given word. Hence, the meaning/related words of the given keyword are obtained using Wordnet3.0 by finding the related terms of each word in an iterative manner. Services obtained during this filtering are ranked based on different degrees of matching. This filtering mechanism can provide better efficiency in mining accurate data from the service repository.

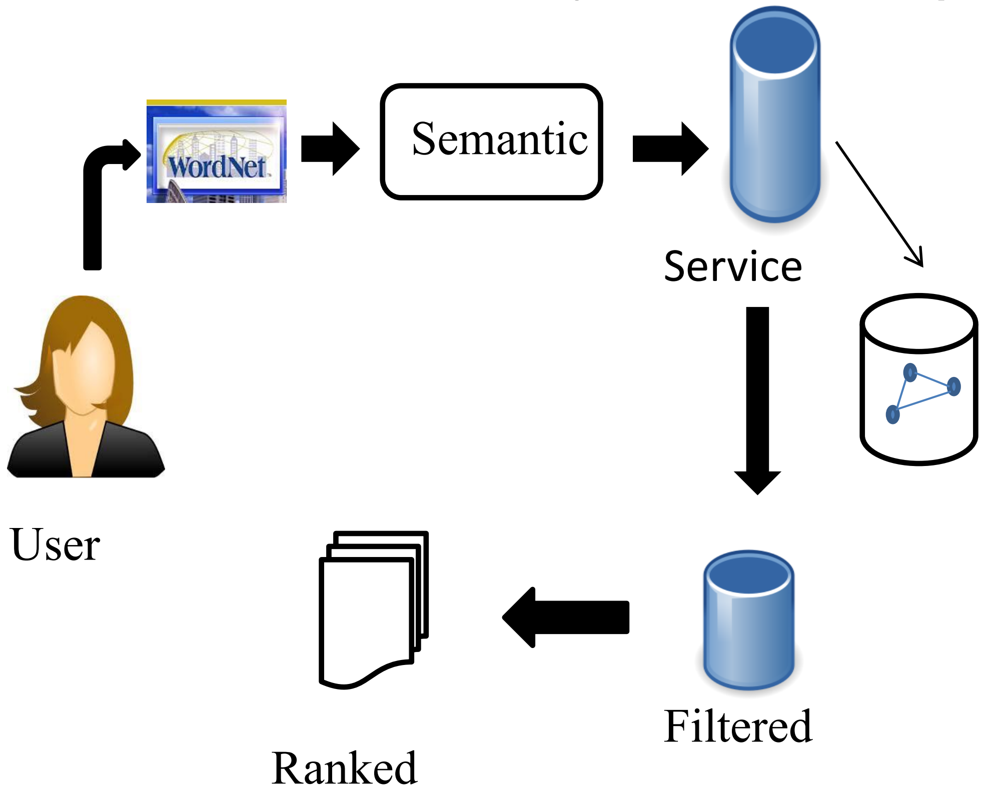

Figure 2: Proposed System architecture

Semantic Filtering

4.1 Finding similar or related words

The seed (also called as key) is obtained from the user as the input. The seed may be a single word, or can be even more than one word. The seed is now processed by the system. The system first tries to find

the related words of the given keyword. This helps to retrieve more relevant data.
The retrieved words are then analyzed further to get more relevant answers. Each word is analyzed further using the WorldNet tool to get the related items/ words. The process may be repeated for each word found. This can help to get more relevant items and paves way for discovering the relevant services.

4.2. Querying

Filtering is performed based on the similar words found by Word Net. SPARQL query Q_{some} filter is used to perform filtering. $\mathbf{Q_{some}}$ returns all services from the repository that contains at least one or more concepts referred by the user request. It uses UNION keyword.

Listing 1.$Q_{some\ query}$

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

Based on the OWL-S profile containing the keywords as well as the similar words, the SPARQL query Q_{some} is formed. Listing 1 shows a sample Q_{some} query generated using SPARQL.

ii. Ranking of the services

The services obtained from the above filtering process are ordered according to seven degrees of match based on the ontology tree constructed as shown in figure 4. The possible degrees of match are Exact, Direct subclass, Subclass**,** Direct super class, Super class, Sibling, Fail. Based on these seven degrees the services are ordered.

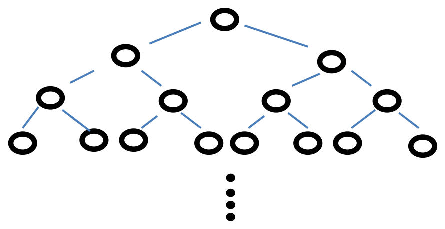

Figure3:Ontology tree

The services which are having exact match are listed first then direct subclass, subclass and so on. All the services are ordered in this manner. Finally the ordered services are stored in the filtered repository. It contains only the ordered relevant services. Thus the input for the discovery process is reduced fairly. This will improve the performance of the discovery process.

V. ILLUSTRATION

Let D = (O, S, U) be a 3-tuple that represent a discovery scenario. Here, O is a set of domain ontologies such as O ={O_1, $O_{2,}$…… O_n}, S is a set of service descriptions. S_i is defined by several terms t_{ij} . Each term refer to a set of concepts C_i defined in the ontology O_{Si} . Thus service term Si = {(t_{i1}, C_{i1}), . . , (t_{in}, C_{in}) : C_{i1} U … U C_{in} subset of S_i}. U is a user request which contains requirements in the form of terms that refer to some subset of concepts from domain ontology O_u subset of O**.** *U = {(t_1, C_1), . . . , (t_n, C_n) : C_1 U ... U C_n subset of O_u}.*

For example, consider the user is searching for scholarship provided by the government. The concepts are extracted from the user request to model the user request as an OWL-S construct. In OWL-S, service profile contains inputs, outputs, preconditions, and results of the service. Based on these constructs the services are filtered. For simplification only the input and output terms are considered for filtering. The user request is:

U={(*inputTerm*$_{u1}$,{**Government**}),(*InputTerm*$_{u2}$,{**Academic degree**}),(*outputTerm*$_{u3,}${**Scholarship**)}

The related words of the input and output terms are found iteratively using Word Net. Are as follows:

Iteration 1

Government={authority, regime, politics} Academic degree={grade, level}
Scholarship={funding, learnedness}

Iteration 2

Authority={authorization, government, agency, dominance}
Funding= {financial support, financing, financial backing

}

Likewise the similar words for the above found words are also found using Word Net. This process is repeated until there are no new similar words found. After that the ontology tree is created for the obtained similar words.

After including similar words,

U={(*inputTerm*$_{u1}$,{**Government**}), (*inputTerm*$_{u2}$,{authority}), (*inputTerm*$_{u3}$,{regime}), (*inputTerm*$_{u4}$,{politics}), (*inputTerm*$_{u5}$,{authorization}), (*inputTerm*$_{u6}$,{dominance}), (*inputTerm*$_{u7}$,{**Academicdegree**}), (*inputTerm*$_{u8}$,{grade}), (*inputTerm*$_{u9}$,{level}), (*inputTerm*$_{u10}$,{class}), (*outputTerm*$_{u11,}${**Scholarship**}), (*outputTerm*$_{u12,}${funding}), (*outputTerm*$_{u13,}${learnedness}) *(outputTerm*$_{u14,}${financial backing})}

Consider a subset of the Service Repository, which contains the following services related to academic domain.

S$_1$={(*inputTerm*$_{11}$,{Government}), *(inputTerm*$_{12}$,{Academicdegree}), (*outputTerm*$_{13,}${Lending})}
S$_2$={(*inputTerm*$_{21}$,{Government}),*(inputTerm*$_{22}$,{Academic degree}),(*outputTerm*$_{23,}${Scholarship})
S$_3$={(*inputTerm*$_{31}$,{Government}), (*inputTerm*$_{32}$,{Academicdegree}), (*outputTerm*$_{33,}${Funding})}
S$_4$={(*inputTerm*$_{41}$,{Authority})(*inputTerm*$_{42}$,{level}), (*outputTerm*$_{43,}${Financialsupport})}
S$_5$={(*inputTerm*$_{51}$,{Academicitemno}), (*outputTerm*$_{52}$,{publication}),(*outputTerm*$_{53,}${Author})}
S$_6$={(*inputTerm*$_{61}$,{Authority}),(*inputTerm*$_{62}$,{level}),(*outputTerm*$_{63,}${scholarship})}

The global domain ontology is considered as the set of concepts involved in previous descriptions:

O={government,funding,lending.authority,funding,academicitemnumber,level,publication,author}

The result after execution of Q_{some} query is:

$$Q_{some}(S, U) = \{S_1, S_2, S_3, S_4, S_6\}$$

The services $S_{4,}$ S_6 which contains the similar words is also included in the returned list of services. These services are left out in exact filtering. Thus the result is improved in this filtering. The obtained services are ordered based on the ontology created. The service S_2 has exact matching for all the terms and so is ranked first, then the services S_1,S_6 next since they posses direct subclass match followed by the service $S_{4,}$ which has subclass relation with the concepts in the request. Finally the service S_3 is listed since it has exact matching for two terms and fail for one term.

VI. ANALYSIS AND EVALUATION

6.1 Implementation

The proposed work is carried out using Java. The filtering process is done using SPARQL queries. The output of the filtering process is analyzed and the experimental results prove the soundness of the proposed semantic filter.

6.1.1 Experimental Scenario

To evaluate the proposed work, a test collection should be used. For this work OWL-S Service Retrieval Test Collection is used. It consists of 1083 OWL-S services. There are nine domains available in the collection such as Education, Medical care, Communication, Food, Travel, Economy, Weapon, Geography, Simulation. Using the proposed filters the services are filtered based on the user request. To prove the effectiveness, the recall value is calculated. It is the measure of the ability of the system to retrieve the relevant services.

$$\textit{Recall} = \frac{\textit{No. of relevant services retrieved}}{\textit{No. of relevant services in the collection}}$$

Recall value is calculated in the case of both existing and the proposed approach.

6.2 Analysis

Filtering is done before the discovery technique. So it is necessary that the filtering technique should not discard the relevant and related services. However, in exact filtering some related and relevant services gets discarded when the filtering process is done prior to the discovery process. This paves the way for the introduction of the semantic based filtering.
The semantic filtering proves to be the best way to filter the unrelated services and avoid filtering the relevant services. Wordnet3.0 is used for adding

semantics to the filtering process. To analyze the performance of both filters recall rate is calculated. The number of relevant services retrieved by applying both the filters is listed in the Table1 .

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

Table 1. Relevant services retrieved by exact and semantic filtering

For the experimented query there are totally 55 relevant services in the test collection. By applying exact filtering 35 relevant services are retrieved and by applying semantic filtering 53 relevant services are retrieved. By using these values, recall rate is calculated. Figure 3 shows the performance of semantic and the exact filtering techniques in terms of recall rate.

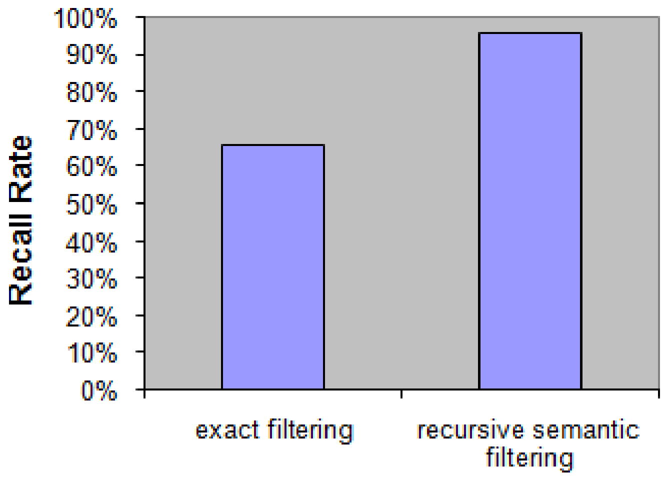

Figure 3: Performance of exact and semantic based filtering

The results show that the semantic filtering has considerably high recall rate than the exact filtering. The calculated recall rates for exact and semantic filtering are 66% and 96% respectively. It is evident that applying semantic filtering can provide an efficiency of 20% to 30% greater than exact filtering. Thus, the usage of semantic search can be more practical in nature.

CONCLUSION

Scalability is a major problem in the discovery of the semantic web service. In order to enhance scalability, the number of services to be used during the discovery process has to be reduced. This is done by applying a SPARQL filter query which is enhanced with semantics, before the actual discovery process. The query enhanced using Word Net is automatically generated from the user request to filter the irrelevant services. The obtained services during filtering are ranked based on their degrees of match.

The filter can be applied to any Semantic Web Service framework because it is based only on domain concepts referred by service descriptions and user requests. Also the proposed semantic based preprocessing mechanism reduces the input for the discovery techniques by discarding the unrelated services and retaining a ranked collection of all relevant services.

References

[1] Alireza Zohali and DR.Kamran Zamanifar, (2005), 'Matching Model For Semantic Web Services Discovery', Journal of Theoretical and Applied Information Technology.

[2] Antonio Ruiz-Cortes, David Ruiz, Jose Maria Garcia ,(2012), 'Improving Semantic Web Services Discovery Using SPARQL-Based Repository Filtering', Journal of Web Semantics.

[3] Belaid D, Chabeb Y, Tata S, (2009) ' Toward an Integrated Ontology for Web Services', ICIW, IEEE Computer Society, pp.462–467.

[4] Bernstein A, Kiefer C, Stocker M, (2007), 'The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks', in: K. Aberer, et al. (Eds.), ISWC/ASWC, Vol. 4825 of LNCS, Springer, pp. 295–309.

[5] Burstein M, Hobbs J , Lassila O, Martin D, McDermott, (2006), 'OWL-S: Semantic Markup for Web Services', Tech. Rep. 1.2, DAML.

[6] Hepp M, Hoffman J, Stollberg M, (2007), 'A Caching Mechanism for Semantic Web Service Discovery', in: K. Aberer, et al. (Eds.),ISWC/ASWC, Vol. 4825 of LNCS, Springer, pp. 480–493.

[7] Horrocks I, Li L, (2003), 'A software framework for matchmaking based on semantic web technology',in: WWW, ACM Press

[8] Kaufer F, Klusch M, (2009), 'WSMO-MX: A hybrid Semantic Web service matchmaker', Web Intelligence and Agent Systems.

[9] McGuinness D.L,van Harmelen F, (2004), 'OWL Web Ontology Language Overview, Recommendation', W3C.

[10] Prud'hommeaux E, Seaborne A, (2008), 'SPARQL Query Language for RDF'

ABOUT THE AUTHORS

Ms. J. Jenifer, is a PG Student of Computer Science and Engineering in PSN College of Engineering and Technology, Tirunelveli. She is working in the area of Semantic Web Services under the guidance of Mrs. M.Deepa Lakshmi. She received her Bachelor's degree from Anna University, Chennai. Her research interests include web mining and mobile computing. (e-mail: jeniferjosepha@gmail.com)

Mrs. M.Deepa Lakshmi, is a Research Scholar in Noorul Islam University, Kumaracoil, under the supervision of Dr. Julia Punitha Malar Dhas. Her research is centered on the semantic web services. She is working as Associate Professor in the Department of Computer Science and Engineering at PSN College of Engineering and Technology, Tirunelveli District, TamilNadu with more than 13 years of teaching experience. Her other areas of interest are programming and web design. (e-mail: deepasuresh12@gmail.com)

Dr. Julia Punitha Malar Dhas, received PhD degree in Computer Science in 2010 from Dr. M.G.R. University. She received her M.E. (CSE) degree from Madras University and B.E. degree from Bharathiyar University. She is currently working as the Professor and Head of the Department of CSE at Noorul Islam University, Kumaracoil, K.K.District, TamilNadu with more than 13 years of teaching experience. Her research interests include Wireless Networks, Mobile computing.(e-mail: julaps113@yahoo.com)