International Journal of Soft Computing 8 (4): 313-320, 2013
ISSN: 1816-9503


# A Review on Semantic Web Service Discovery Algorithms

[1]M. Ramakrishnan and [2]M. Suchithra
[1]Department of Information Technology, Velammal Engineering College, Chennai, India
[2]Department of Computer Science and Engineering, Sathyabama University, Chennai, India

**Abstract:** Semantic Web Services (SWSs) are an extension to the Web services, SWSs are the most recent and revolutionary technology developed mainly for machine to machine interaction. With the development of Web service applications how to improve the efficiency of service discovery is an important research work in modern times. There are a lot of approaches and algorithms used for this Web service discovery. The aim of this review study is to bring out various Web service discovery algorithms being used so far and provide an overview of those approaches with their pros and cons.

**Key words:** Semantic Web services, Web service discovery, ontology, algorithm, technology

## INTRODUCTION

The Web is evolving from being a static source of information to a highly dynamic network where resources are shared and information is generated on demand. Web services discovery is an important part of the Web service system architecture. The Web service discovery is the user in some way to search the different types of Web services and can get its all aspects of specific information. Web services are the best-known instantiation of service oriented computing.

In this context, Web services form the main building blocks to construct distributed network-based applications. In order to overcome the limitations of WS discovery, concept of semantics has been introduced with OWL-S in which functionality of a service is described in terms of inputs, outputs, preconditions and effects. Semantic Web services enable the automatic discovery of distributed Web services based on comprehensive semantic representations (Akkiraju *et al*., 2005). Ontology is used to describe the nature of things. In practice, ontology is often a formal vocabularies, its main role is to define a field and the specialized vocabulary of the field and the relationship between these terms. Service ontology Os is defined as (Qiu *et al*., 2007):

$$Os = <c, a, r>$$

Where:

c = A set of name concepts
a = The set of property concepts
r = A set of axioms (this reflects the relationships between concepts)

Ontology is considered to communicate between different entities within the field based on semantic. Semantic Web needs to add to the semantic information in order to realize the function of the logical reasoning. Current SWS frameworks such as WSMO and OWL-S address the allocation of distributed services for a given well-described task but none of the Web services fully solve the issues related to symbolic semantic Web-based knowledge representations. In OWL-S the functional and semantic characteristics of Web services are present but the performance characteristics is absent (Liu *et al*., 2006).

## CHALLENGES IN WEB SERVICE DISCOVERY

The Web service discovery previously was based on WSDL which contains the description of each and every Web service. But a Web service discovery required more intelligible description than the technical details presented in the WSDL description.

So, the semantic technologies are been used in this Web service discovery mechanism. Dynamic discovery based on semantic description of services is an essential aspect of the semantic Web Services integration process (Htoo and Nyunt, 2008). The goal of semantic Web services is the use of richer more declarative descriptions of the elements of dynamic distributed computation including services, processes, message-based conversations, transactions, etc. (Batra and Bawa, 2010). The centralized methods such as UDDI are not adapted or suitable for dynamic, flexible and evolutionary environments. Various solutions to overcome this problem have been suggested.

**Corresponding Author:** M. Ramakrishnan, Department of Information Technology, Velammal Engineering College, Chennai, India

## WEB SERVICE DISCOVERY MECHANISMS

**Service Dependency Graph (SDG):** The Web service discovery mechanism through Service Dependency Graph (SDG) is a very simple and easy approach without much complexity. Service Dependency Graph (SDG) is a directed graph that is constructed dynamically to show all possible input-output dependencies among the Web services registered in some selected service categories. The construction of SDG is based on the service interface descriptions of registered web services written in the Web Services Description Language (WSDL). In WSDL descriptions each operation o is described as a pair (In, Out). "In" denotes the set of attributes of data entities/objects which are the input service operation and "Out" denotes the set of attributes produced:

$$Ino = \{di | di \in Onto(S), o \text{ takes values of di as input}\}$$

$$Outo = \{dj | dj \in Onto(S), o \text{ produces values of dj as output}\}$$

In order to make all the input data to be available before the operation all the input data nodes that are required to perform an operation are connected to that operation node through the directed edges that are logically ANDed and all the operation nodes that can produce values for a particular data node in an SDG are connected to that data node through directed edges that are logically ORed. A dummy node is added to the original AND/OR graph and connects it to all the input data nodes that are known to the requestor with directed edges. This dummy node is called the termination node explained in detail by Liang and Su (2005).

The bottom-up search strategy is used algorithm to be presented starts at the termination node which connects to all the known data entities and operations and ends at the starting node if a solution can be found. The graph used in this algorithm can handle cycles (Liang and Su, 2005). There is a cost assignment scheme in which a unit cost c is assigned to all operation nodes and the AND nodes are created for inter attribute constraints. The algorithm shown below is used by Linag and Su (2005).

**Algorithm:**

```
/**
* Bottom-up AND/OR graph search
*
* The graph has a finite number of nodes. It may have loops.
* t: the termination node.
* s: the starting node.
* K: a list of nodes to be explored.
* g(G' '•SG(n), n) is updated with the cost of a better solution graph as G' ' is expanded.
* A known node: a node whose solution has been reached by the algorithm.
* An optimized node: a node whose minimal solution graph has been found and the node has been removed from K.
**/
Find{
K is initialized with the termination node t ; //step (a) Until K is empty {
Remove the next node n in K with the smallest g (G".SG (n) , n); //step(b)
If n is the starting node s
Break;
J = expand (n); // J is a set that consists of the child nodes of n
If (J != null) {
for each j ∈ J {
if j is an OR node{
if g(G".SG(j), j) > E(j) + g(G".SG(n) , n){//step(c)
//Updating the cost of the solution graph of j, if a
//better solution has been found
g(G".SG(j), j) = E(j) + g(G".SG(n) , n);
//Mark n as the parent of j in the minimal
// solution graph
MarkedParent (j) = n;
}
if (j has not been visited and j ∉ K) {
add j to K;// step(e)
Label j as known; // G" is expanded
}
}
else {
if all j's parents are known { // step (d)
g(G".SG(j) , j) = sum over j's Parent p(g(G".SG(p),
p)) + E(j);
if (j has not been visited and j ∉ K){
add j to K;// step(e)
Label j as known; // G" is expanded
}
}
else
Record that one more parent of j (i.e., n) is known;
}
}//for each child of n
} //if J! = null
n is optimized (solved); // This statement is reached if either n has no child
// or n has been processed.
} //end of until
if s is optimized (solved), the minimal cost of the solution graph of s is g(G".SG(s), s),
// Step (f)
else report that no solution can be found;}
```

If the SDGs become larger, the number of operations becomes more in number and thus the formation of SDGs consumes more time.

**Service clustering:** Based on the service ontology that is generated by domain modeling towards the user's common needs, cluster, i.e., categorize the services which realize same function but have different QoS values into different service clusters. This method will not consider the irrelevant services towards certain request thus making the service discovery efficient. From the study (Liu *et al*., 2011), SO1 represents the specific type of service ontology and WS2 represents the service instance for the similarity calculation. Sim (SO1, WS2) can be calculated using the weighting method. Capability similarity is calculated by using an algorithm.

**Algorithm:**

```
SimPre(Prec1p, Prec2p)
Input: Prec1p = entity1p: state1p, Prec2p = entity2p:
state2p
Output: The similarity between Prec1p and Prec2p
If match (entity1p, entity2p) _ Fail Then
If state1p = state2p Then
return 1/2*(match(entity1p, entity2p)+1)
Else
If exists the status path from state1p to state2p and the status path length is
Len, Then
return 1/2*( match(entity1p, entity2p)+1/(Len+1))
Else
return 0
```

Experiments have been done to compare the service clustering time and accuracy with other approaches. Though the accuracy and service clustering time are evaluated there is a pitfall to set the threshold automatically in this approach.

**Filtering the web service registry:** Two problems of keyword searching have been identified by Pilioura *et al.* (2003); low precision and reduced recall. To enhance the performance of Web service matching approach, a framework for filtering the Web service descriptions registry is employed when receiving a client request. By filtering the registry, researchers apply the matching algorithm (Brogi *et al*., 2004) only to candidate descriptions that are relevant to the client request. The filtering stage aims at narrowing down the number of advertisements to be checked in detail. There is an extension to the OWL-S service profile ontology that enriches the OWL-S upper ontology with a global category property that refers to global well known categories defined in global category ontology. A set of the filtered advertised descriptions is a potential set of descriptions that may satisfy the client query. Web service advertisers and requesters will use concepts from the Global Concepts ontology to categorize the Web service they offer and request, respectively. Indeed, filtering the Web service descriptions will be based on these concepts. Two checks are performed when a Web service registers itself to the advertising entity: Consistency checks, Global category check. To determine whether or not an advertisement is relevant to the client request 'filtering concepts taxonomy' is applied in the study (Khdour and Fasli, 2010) (Fig. 1).

According to this approach if no result found for a search, the relevant services obtained do not exactly satisfy the user's requirement.

**Preconditions and effects:** A simple syntax based matching can produce many false positives since nature of service is not captured in the service description. Concepts of semantics in OWL-S have been introduced to overcome this limitation (Martin *et al.*, 2004). An algorithm for matching a service discovery query with a service description based on the precondition and effect parts of the description is used, preconditions are necessary at service provider's end and effects are needed at client's end.

The algorithm uses the notions of Satisfiability Modulo Theory to produce polynomial time solutions to what is essentially an NP complete problem. The 4 degrees of matching introduced by Paolucci *et al.* (2002) are being used in this algorithm. Each precondition has two types of expressions.

**Simple expression:** Atomic expressions in which there is a single operator and two operands.

**Complex expression:** specifies disjunctive, conjunctive relation between two or more clauses and negation relation associated with one or more clauses each of which can be complex or simple expressions. Condition matching involves three different phases:

```
<owl:Ontology rdf:about=""/>
<owl:ObjectProperty rdf:ID="globalcategory">
<rdfs:domain rdf:resource="http://www.daml.org/services/owl-s/1.1/
ServiceProfile.owl#Profile" />
<rdfs:range rdf:resource="http://localhost/GlobalConcepts.owl"/>
</owl:ObjectProperty>
<owl:Class rdf:about=" http://www.daml.org/services/owl-s/1.1/
ServiceProfile.owl #Profile">
<rdfs:comment>A profile can have one global category</rdfs:comment>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#globalcategory" />
<owl:cardinality rdf:datatype="http://www.w3.org/2001/
XMLSchema#nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Fig. 1: An extention of OWL-S service profile: The global cetegory property (Khdour and Fasli, 2010)

- Parameter compatibility
- Condition equivalence
- Condition evaluation

Preconditions are checked just before invocation of the service at the client side. The algorithms used are shown below, for more reference refer (Bellur and Vadodaria, 2010).

**Algorithm 1:**

```
MatchCondition (QConditions, A Conditions)
Graph G = Empty Graph (V0 + V1, E)
V0 = QConditions
V1 = AConditions
for each query_condition in QConditions do
for each advt_condition in AConditions do
      check for parameter compatibility
      check for condition equivalence
      edge (query_condition, advt_condition) in G = lowest
      degree of matching among all the terms
end for
end for
Graph R = HungarianMatch(G)
if (R = null) then
No complete matching exists
return Fail
end if
Let (Q, A) denote Min weight edge in R
degree = DegreeOfMatch (Q, A)
return degree
```

**Algorithm 2:**

```
Match (Query, Advt)
QInputs = All input terms of Query
AInputs = All input terms of Advt
QOutputs = All output terms of Query
AOutputs = All output terms of Advt
Degree_Inputs = Match (QInputs, AInputs)
Degree_Outputs = Match (AOutputs, QOutputs)
Degree_1 = min (Degree_Inputs, Degree_Output
QPreconditions = All preconditions of Query
APreconditions = All preconditions of Advt
QEffects = All effects of Query
AEffects = All effects of Advt
Degree_Precondition = MatchCondition(QPrecondition, APreconditions)
Degree_Effect = MatchCondition (QEffects, AEffects)
Match = Degree1 * 100+Degree_Precondition*
10+Degree_Effect
return Match
```

From three different phases of matchmaking process, a 3-digit number is created that indicates overall degree of matching which is used to discover services more easily. In this approach, it is demonstrated that the performance of the algorithm is better and it is possible to add preconditions and effect matching to simple input and output term matching without runaway search times.

**Web service similarity:** The two aspects of Web services similarity measure: function similarity and process similarity. They both are based on the concept of semantic similarity which is explained by Xie *et al.* (2011). A Petri net is one of several mathematical modeling languages for the description of distributed systems. A Petri net consists of places, transitions and directed arcs that connect a place to a transition or a transition to a place. Petri net is a graphical tool that can accurately describe the system structure and process. Petri net is increasingly applied to the Web service process modeling and analysis. Web service process can be transformed into Petri net through next steps:

**Step 1:** If the process is an atom process, go to step 2: else go to step 3.

**Step 2:** Analyze the atomic process and the relationship with other atomic processes and describe with Petri net. Turn to next process and go to step 1 if there is not process, exit.

**Step 3:** Split process. Use Petri nets to describe the process control construct and split it into sub processes, go to step 1.

Transition is the atom process which can be invoked. Places save the state of the transition. In order to calculate the process similarity, it is necessary to transform the Petri net to concept string. The core of the Petri net is transition which is shown by the concept of atomic process. So, the string consisted with the concepts can show the web service process. Some researches use the edit distance between strings to calculate the similarity of the process.

The edit distance between two strings is the minimum changing times of one string to another. The changes include insert, remove and replace. But the edit distance requires an accurate comparison between the concepts and not related to semantic information. This study is based on the sequential patterns similarity, uses the results of concept semantic similarity to calculate the process similarity.

Let Astr, Bstr as the process concept string of Web Service A and B. Suppose BStr is longer than AStr. AStri means the ith concept in the string. The process similarity is:

$$\text{Process\_Sim}(\text{WSA},\text{WSB}) = \frac{1}{L}\max\sum_{i=1}^{L}\text{Sim}(\text{AStr},\text{BStr}_{i+k})$$

$$L = \min\left(\text{Length}(\text{AStr}),\text{Length}(\text{BStr})\right);$$

$$0 \le i+k \le \text{Length}(\text{BStr})$$

If there is condition information in the string, it is necessary to judge whether the condition is satisfied then calculate the concept similarity. In the process similarity algorithm if the two Web services control constructs are different, the similarity is low. Because the match is according to the processes sequence, it is effective to identify the different structures of Web service process. The web service similarity given as:

$$\text{Service\_Sim}(WSA,WSB)=\lambda_1 \text{Function\_Sim}(WSA,WSB)+\lambda_2 \text{Process Sim}(WSA,WSB)$$

Where:

$\lambda_1$ = The function similarity weight

$\lambda_2$ = The process similarity weight

So, researchers can set different weights to adjust the Web service similarity. Generally we set $\lambda_1>\lambda_2$ that means the function similarity is more important than process similarity in the Web service.

This study focuses on the Web service similarity which may affect the accuracy directly and several influence factors are also considered in the algorithm. Though the algorithm can cluster the Web services more accurately the efficiency of discovery mechanism is not been demonstrated. Literature (Lu *et al*., 2012) proposes a Web services discovery algorithm based on the similarity of the functional description of the Web service but does not calculate the semantic similarity of the description of the input/output parameters.

**Personalization and contextualization approach:** It is believed that the fundamental problems of Information Retrieval (IR) in general and services discovery in particular are about the representations of the semantics pertaining to the queries and the target resources and the prediction of the relevance of the target resources (services) with respect to a user query (Rong *et al*., 2008). A novel design and development of an agent and ontology based service discovery and personalization framework is illustrated by So *et al*. (2009). As user queries are usually short, e.g., only 2x words long on average (Jansen and Spink, 2006), query personalization and contextualization is essential for effective Web service discovery. To eliminate the need to manually adapting the cluster deployment context each time they are deployed, a contextualization technique is applied for enabling dynamic creation of functioning virtual constructs aware of their context in cluster deployment (Keahey and Freeman, 2008).

The system architecture for service personalization and contextualization system (SPA) is depicted in detail by So *et al*. (2009) (Fig. 2). For user profile creation and

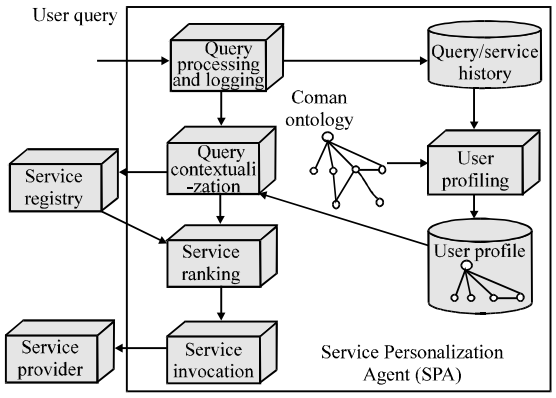


Fig. 2: General systems archiecture of SPA (So *et al*., 2009)

updating, the SPA System utilizes the ODP ontology to establish the ontological user profiles. When a new user profile is created, a copy of the ODP ontology will be instantiated. Each concept node is described by a set of terms selected based on their TFIDF weights derived from the background web documents attached to a specific ODP category. In addition each node is assigned an Interest Score (IS) which represents the user's specific service interests. At the initialization time each node is assigned a raw interest score 1 that is the user may have the same preference for each ODP category. After service ranking, the user may select to invoke particular Web services. The description of an invoked service is matched with the nodes in the ontological user profile to identify which domain (context) the user is really interested in. In particular, the matching process is conducted using the cosine similarity function (Salton and McGill, 1983).

**Domain specific nodes:** The domain specific nodes discovery mechanism is a networking infrastructure of a distributed Web service search system and its ontology-based routing method. An ontology for semantic web annotation in which semantic relations and hierarchy between terms is defined is selected.

To overcome the problems that may arise because of using semantic annotation of web services a CODS (Collection of Domain Specific nodes) with a P2P network (peer to peer network is established in order to communicate with each and other nodes) that uses domain-specific Web service discovery sub-systems which are built around ontology is used. Network topology and ontology based routing is being used. Communication among CODS nodes is handled by simple text messaging. Basic operatio in network topology are:

- Join to network: The new nodes are joined to the network by inserting the details of the new CODS node to one of the peer nodes. Node checks its database and either it ignores the message if it is already exists or adds the new node to the network
- Leave from network: A peer can leave the network in two ways: implementing a resign procedure or without informing any other peers
- Related node routing: If there is no result in the current ontology then the information is transferred to related nodes

The ontology routing (Canturk and Senkul, 2012) CODS System is designed to use ontology for routing the requests in between CODS peer nodes. Ontological routing is used both in domain specific crawler layer for forwarding an unrelated Web service found during crawling and in service discovery layer for routing service search request to the CODS nodes. Instead of complete ontology which is costly a subset of the ontology can be used in routing.

In ontology routing, decision is based on the similarity of service to the target domain. Generally nodes share a common ontology and try to find the optimum number of the ontology terms kept in the local index of the node but this enhanced approach can find the optimum number of ontology terms in local index by not sharing an ontology.

**An unstructured peer based approach:** The approaches based on P2P approach offers a self-organized and decentralized environment and the interactions between web services are dynamic. More researched made on Web service P2P computing has been told by Gharzouli and Boufaida (2010, 2011) and Barhamgi *et al.* (2007). In this P2P approach two main concepts were used:

- An epidemic discovery
- A composition table

The discovery algorithm used here offers a distributed solution to the semantic Web service in unstructured Peer to Peer networks and the composition table offers a distributed to preserve and publish description of the Web services (Fig. 3).

This particular approach also goes a step ahead and paves way for Web service composition when there is a requirement. The architecture that is the base for the approach used here is presented in Fig. 4.

The JXTA is used because of the peer to peer infrastructure interoperability, platform independence ubiquity. The JXTA allows two peers to interact with each other even if the environment is protected by firewall.

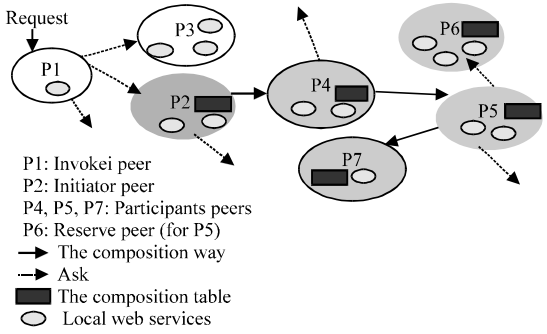


Fig. 3: Example of an unstructured P2P network (Gharzouli and Boufaida, 2010)

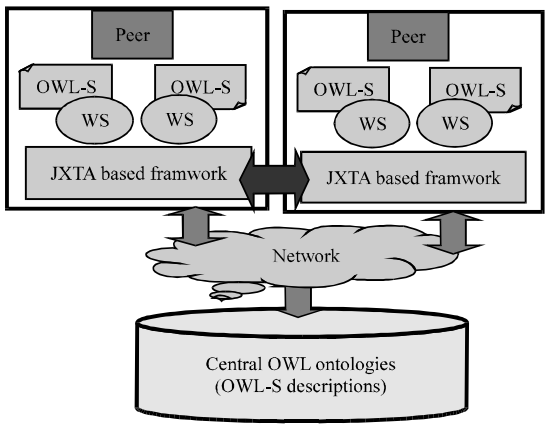


Fig. 4: A reference architecture (Gharzouli and Boufaida, 2010)

**Begin:**
Receive the request (init-Input, the init-Output and the goal).
Search a local Web service where [(WS-input = Init-Input) and (WS-output = Init-Output) and (WS-goal = Goal)]
If (there is a local WS) then execute WS; send the response; go to END
If (there is not a local WS) then
Compose a local Wwb service [(LocCWS-input-Inpit-Input) and (LocCWS-Output-Init-Output) and (LocCWS-goal = Goal)];
if (there is a LocCWS) then execute LocCWS; send the response; go to END;
If (there is not a LocCWS) then
Search-in-Composition-table (Init-Input, Init-Out-put, Goal);
If (there is not P2PCWS in the composition table) then
Launch-a-New-P2P-composition();
End.

The 'Epidemic Discovery Algorithm' is based on input/output matching. In this context, an example of a universal OWL ontology and a collection of OWL-S descriptions for a variety of Web services is generated manually by Ganjisaffar and Saboohi (2006) which contains more than 240 semantic services descriptions.

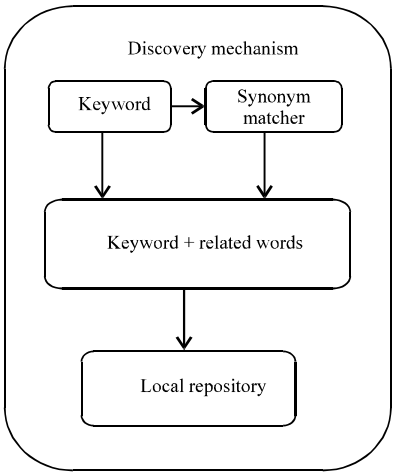


Fig. 5: Proposed architecture

## PROPOSED ARCHITECTURE

Figure 5 shows the proposed architecture, instead of having a central repository that contains all the registered services, it is proposed to have a which that central repository is categorized based on their similarity called "local repository". A "Synonym Matcher" that contains various related words that yield all the related words to the keyword/s which is taken as the input. In this approach, the use of Synonym Matcher enables the discovery of all related Web services when the exact service is unavailable due to problems like non existence or service not in use.

## CONCLUSION

In this study, researchers have presented the various approaches used in Web service discovery and are explained in a brief manner. The architecture used for each methodology is studied well and shown how each method differs from the other. It is clearly seen that as there is a need for better performance and accurate results, the methods used for Web service discovery also have been developing along with the needs. From graph match making to domain specific matching or any such approach each approach aims to provide better results though there may be few drawbacks. Those drawbacks are rectified in the future approaches introduced. Future approach will definitely come up to meet the requirements of the users.

## REFERENCES

Akkiraju, R., J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth and K. Verma, 2005. Web service semantics-WSDL-S. November 7, 2005. http://www.w3.org/submission/wsdl-s/.

Barhamgi, M., P.A. Champin, D. Benslimane and A.M. Ouskel, 2007. Composing data-providing web services in P2P-based collaboration environments. Proceedings of the 19th International Conference on Advanced Information Systems Engineering, June 11-15, 2007, Trondheim, Norway, pp: 531-545.

Batra, S. and S. Bawa, 2010. Review of machine learning approaches to semantic web service discovery. J. Adv. Inform. Technol., 1: 146-151.

Bellur, U. and H. Vadodaria, 2010. On extending semantic matchmaking to include preconditions and effects. Proceedings of the IEEE International Conference on Web Services, September 23-26, 2008, Beijing, China, pp: 120-128.

Brogi, A., S. Corfini and R. Popescu, 2004. Flexible matchmaking of web services using DAML-S ontologies. Proceedings of the 2nd International Conference on Service-Oriented Computing, November 15-19, 2004, New York, USA.

Canturk, D. and P. Senkul, 2012. Ontology-based routing of web services in distributed service discovery system containing domain specific nodes. Proceedings of the IEEE Symposium on Computers and Communications, July 1-4, 2012, Cappadocia, Turkey, pp: 283-288.

Ganjisaffar, Y. and H. Saboohi, 2006. Semantic web central: Project sws-tc. http://projects.semwebcentral.org/projects/sws-tc/.

Gharzouli, M. and M. Boufaida, 2010. A distributed P2P-based architecture for semantic web services discovery and composition. Proceedings of the 10th Annual International Conference on New Technologies of Distributed Systems, May 31-June 2, 2010, Tozeur, Tunisia, pp: 315-320.

Gharzouli, M. and M. Boufaida, 2011. PM4SWS: A P2P model for semantic web services discovery and composition. J. Adv. Inform. Technol., 2: 15-26.

Htoo, N.Z.C. and T.T.S. Nyunt, 2008. Semantic web services offer discovery using OWL-S IDE. Proceedings of the 2nd International Conference on Signal Processing and Communication Systems, December 15-17, 2008, Gold Coast, Australia, pp: 1-5.

Jansen, B.J. and A. Spink, 2006. How are we searching the world wide web? A comparison of nine search engine transaction logs. Inform. Process. Manage., 42: 248-263.

Keahey, K. and T. Freeman, 2008. Contextualization: Providing one-click virtual clusters. Proceedings of the 4th IEEE International Conference on eScience, December 7-12, 2008, Indianapolis, IN., USA., pp: 301-308.

Khdour, T. and M. Fasli, 2010. A semantic-based web service registry filtering mechanism. Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, April 20-23, 2010, Perth, WA., USA., pp: 373-378.

Liang, Q.A. and S.Y.W. Su, 2005. AND/OR graph and search algorithm for discovering composite web services. Int. J. Web Serv. Res., 2: 46-64.

Liu, C., Y. Peng and J. Chen, 2006. Web services description ontology-based service discovery model. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, December 18-22, 2006, Hong Kong, China, pp: 633-636.

Liu, J.X., K.Q. He, J. Wang and D. Ning, 2011. A clustering method for web service discovery. Proceedings of the IEEE International Conference on Services Computing, July 4-9, 2011, Washington, DC., USA., pp: 729-730.

Lu, G., T. Wang, G. Zhang and S. Li, 2012. Semantic web services discovery based on domain ontology. Proceedings of the World Automation Congress, June 24-28, 2012, Puerto Vallarta, Mexico, pp: 1-4.

Martin, D., M. Burstein, J. Hobbs, O. Lassila and D. McDermott *et al*., 2004. OWL-S: Semantic markup for web services. November 22, 2004. http://www.w3.org/Submission/OWL-S/.

Paolucci, M., T. Kawamura, T.R. Payne and K. Sycara, 2002. Semantic matching of web services capabilities. Proceedings of International Semantic Web Conference, June 9-12, Sardinia, Italy, pp: 333-347.

Pilioura, T., A. Tsalgatidou and A. Batsakis, 2003. Using WSDL/UDDI and DAML-S in web service discovery. Proceedings of the WWW 2003 Workshop on E-Services and the Semantic Web, May 20, 2003, Budapest, Hungary, pp: 1-14.

Qiu, Q., Q. Xiong, Y. Yang and F. Luo, 2007. Study on ontology-based web services discovery. Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design, April 26-28, 2007, Melbourne, Australia, pp: 641-645.

Rong, W., K. Liu and L. Liang, 2008. Towards personalized ranking in web service selection. Proceedings of the IEEE International Conference on e-Business Engineering, October 22-24, 2008, Xi'an, China, pp: 165-172.

Salton, G. and M.J. McGill, 1983. Introduction to Modern Information Retrieval. McGraw Hill Inc., New York, USA., ISBN-10: 0070544840, pp: 448.

So, C.F., C.C.L. Lai and R.Y.K. Lau, 2009. Ontological user profiling and language modeling for personalized information services. Proceedings of the IEEE International Conference on e-Business Engineering, October 21-23, 2009, Macau, China, pp: 559-564.

Xie, L.L., F.Z. Chen and J.S. Kou, 2011. Ontology-based semantic web services clustering. Proceedings of the IEEE 18th International Conference on Industrial Engineering and Engineering Management, September 3-5, 2011, Changchun, China, pp: 2075-2079.