TELKOMNIKA, Vol.12, No.4, December 2014, pp. 1079~1087
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v12i4.681

OWLS-CSM: A Service Profile Based Similarity Framework for Web Service Discovery

Naji Hasan A. H*, Gao Shu
School of Computer Science, Wuhan University of Technology,
Wuhan, China, 430063
Corresponding author, e-mail: hasanye1985@gmail.com[1], gshu418@163.com[2]

Abstract

This paper presents OWLS-CSM framework, an OWL-S Service Profile based simlairity framework for web service discovery in cloud environment. In the proposed framework, services are presented as advertisements; their concepts are semantically defined and described in a hierarchal ontology to facilitate service matchmaking. In matchmaking process advertisements and query are represented as objects and three levels of similarities are used, based on OWL-S Service Profile, to matching, namely taxonomical similarity, functional similarity and non-functional similarity. Milestone method is adopted in the matchmaking algorithm to match the concepts according to their position in the hierarchal ontology. The results obtained from OWLS-CSM are analyzed and compared with other similar works to prove and evaluate the efficiency of our work.

Keywords*: web service discovery, subsumption hierarchy, OWL-S service profile, taxonomical similarity, milestone*

1. Introduction

The increasing use of web services in cloud environment [1] has raised new vital challenges, such as the finding, the most suitable web services that satisfy users' requirements.

Adding annotations to the services in cloud environment, which then called semantic web services (SWS in short) [2], aim at making web services machine understandable and use-apparent, utilizing semantic web technologies for Web service annotation and processing. The idea is to provide ontology-based descriptions of web services that could be processed by ontology reasoning tools. Thus, intelligent agents can be able to automatically understand what a web service offers and what it requires in order to execute a task and facilitates selecting the suitable service that meets user's needs.

In this paper, we propose an OWL-S Cloud Services Matchmaker (OWLS-CSM) framework, in which user can find the suitable service which meets his needs. In OWLS-CSM, a conceptual model for semantic web services is adopted and WSMO Discovery Framework [3] (WSMO-DF) is modified and utilized for Web service discovery. An OWL-S ontology is built and semantic descriptions are expressed as instances of the profile concept of the OWL-S Service Profile (SP). Semantic descriptions cover functional properties - which can be matched later-such as inputs, outputs, precondition, effects and nonfunctional properties, in order to capture fast a set of candidate Web services that meet user's needs.

This rest of paper is organized as follows. Section 2 presents the related literature includes the conceptions we use in our study. Section 3 introduces the theory of service profile similarities and metrics used in the proposed framework. It shows the three levels of similarities and their algorithms as well. Section 4 introduces the environment that OWLS-CSM works in and analysis the obtained results and compare them with similar previous works in order to show the efficiency of our framework. Finally, there is a section for conclusions and future work.

2. Related Works

OWL-S, SAWSDL, WSDL-S, and WSMO are the major languages for semantic standards of Web service annotation. WSMO-DF defined service descriptions with higher level of details. It provides a similar level of detail can also be used in OWL-S, through the Process model or preconditions and effects [4].

Received September 14, 2014; Revised November 1, 2014; Accepted November 17, 2014

In [5], Web services are described as state transitions. In addition, the Rich Web service representation of WSMO-DF is followed, using domain ontology.

Web service descriptions are defined as CCs in OWL and the matchmaking procedure examine the subsumption relationships [6]. There are lots of frameworks developed in order to perform web services discovery. Fc-Match [7] provides a similarity matching taking advantage of WordNet tool. In the work [8], a framework for describing Web services by the DLs is provided.

OWLS-MX matchmaker [9] used both logic-based reasoning and content-based IR techniques for Web services in OWL-S. iMatcher2 [10] adopted a learning algorithms to find the similarities. Other works, [11],[12], obtained Inputs-Outputs annotations special properties, such as service categorization without considering result filtering (grouping filtering).

In our approach, we obtain the Service Profile model that facilitates retrieving the structural ontology information. In addition, we treat with the general properties of an advertisement and a query. Furthermore, our approach utilizes Profile taxonomies in matching and grouping filtering of the result, which is not available in OWLS-MX matchmaker.

In contrast to OWLS-SLR [13], we adopted Milestone method in matchmaking that performs better results compared with OWLS-SLR.

3. Services Profile Similarity in OWLS-CSM

In this section, we introduce main similarity metrics adopted in OWLS-CSM. Those metrics are the Description Logic Hierarchy (DLH), the Description Logic of Roles (DLR) and the Overall Similarity. Firstly, we present the notion of the object specification for representing services advertisements and query instances in OWL-S CSM and then we introduce the metrics mentioned above in details.

DEFINITION 1. An object specification is a five-fold $<\mathrm{ID},\mathrm{C},\mathrm{I},\mathrm{O},\mathrm{NF}>$, where ID is a unique Profile instance, C is the set of the most specific concepts to where ID belongs to, I and O are the sets of Inputs and Outputs as semantic annotation concepts, respectively, and NF is the set of non-functional values.

The notation A is used to represent an advertisement instance and the Q notation to represent a query instance.

In this paper, we model the semantic web service discovery problem in cloud environment as the process of finding the similarity of objects, $A<ID, C_a, O_a , NF_a>$ and $Q<ID, C_q, O_q , NF_q>$.

The solution of the service discovery problem is based on three similarity levels, namely, Taxonomical Similarity, Functional Similarity and Non-functional Similarity.

- Taxonomical Similarity: This is calculated over the C_a and C_q sets of A and Q specifications and represents their similarity in terms of their taxonomical classification in a Profile subclass hierarchy.
- Functional Similarity: This is calculated over the input (Ia and Iq) and output (O_a and O_q) sets of A and Q specifications (Signature similarity).
- Non-functional Similarity. This is calculated over the values of the popular QoS properties of A and Q specifications.

3.1 Taxonomical Similarity

Taxonomical Similarity exploits Description Logic of Hierarchy (*DLH*) metric which computes the similarity of two semantic concepts by their hierarchical distance in an ontology tree. DLH depends on a concept similarity function *Sim* and a finite set of hierarchical filters F. In our work, we assume the function $\mathrm{Sim}(C_1,C_2)$ represents the similarity of two concepts C_1 and C_2, and the value $\mathrm{Sim}(C_1,C_2) \in [0...1]$. 0 represents no matching and 1 represents the exact matching. DLH metric includes four types of hierarchical filters can be generated between two semantic concepts. We adopted the notation $C_1 \xrightarrow{f} C_2$ to say that C_1 matches to C_2 with one of the following filters:

- Exact (e): The two concepts should be equivalent concepts, that is $C_1 \xrightarrow{e} C_2 \Leftrightarrow C_1 = C_2 \vee C_2 = C_1$.
- Plugin (p): The concept C_2 should subsume concept C_1, that is $C_1 \xrightarrow{p} C_2 \Leftrightarrow C_1 \sqsubseteq C_2$.
- Subsume (su): The concept C_1 should subsume concept C_2, that is $C_1 \xrightarrow{su} C_2 \Leftrightarrow C_2 \sqsubseteq C_1$.
- Sibling (sb): The two concepts C_1, C_2 should be subsumed by a concept T and they should not be disjoint, that is, $C_1 \xrightarrow{sb} C_2 \Leftrightarrow \exists \mathrm{T} : C_1 \sqsubseteq \mathrm{T} \wedge C_2 \sqsubseteq \mathrm{T} \wedge C_1 \sqcap C_2 \sqsubseteq \mathrm{T}$.

Using a set of hierarchal filters F, a relation $C_1 \xrightarrow{f} C_2$ represents that the concept C_1 matches the concept C_2 semantically, if and only if there is at least one filter f in F, that is:

$$C_1 \xrightarrow{F} C_2 \Leftrightarrow \exists f \in F : C_1 \xrightarrow{f} C_2$$

DEFINITION 2**.** Assume there are two concepts A and B, and their Description Logic of Hierarchy (DLH) similarity is defined in the range [0..1] , with regard to a concept similarity function S and a set of hierarchical filter F, as formula 1.

$$DHL(A,B,F) = \begin{cases} Sim(\mathrm{A},\mathrm{B}) \text{ if } \mathrm{A} \xrightarrow{F} \mathrm{B}, \\ 0 \quad \text{otherwise.} \end{cases} \tag{1}$$

Formula (1) can be summarized for *Description Logic of Hierarchy (DLH) similarity* on two sets of concepts S_A and S_B as follows:

$$DHL_{set}(S_A, S_B, F) = \frac{\sum_{\forall B \in S_B} \max_{\forall B \in S_B} [DHL(S_A, S_B, F)]}{|\mathrm{S_B}|} \tag{2}$$

In the implementation of *DLH*, we adopted a concept distance measures *Dis* that compute the distance between concepts included in the hierarchal ontology, and then we have:

$$Sim(C_1, C_2) = 1 - Dis_{milestone}(C_1, C_2) \tag{3}$$

The distance between two semantic concepts in a hierarchal ontology is computed by their respective positions in the concept hierarchy. Some previous works [3], [8] have studied this issue but, we adopted Milestone method [9], in our work. In this method, every node, which represents a concept in hieratical ontology, has a value (called milestone); Milestone is obtained from the following formula:

$$milestone(\mathrm{n}) = \frac{1/2}{k^{l(\mathrm{n})}} \tag{4}$$

where *k* is a preset factor ($k > 1$) that shows the occurrence at which the value decreases along the hierarchal ontology (in our method, $k = 2$), and *l*(*n*) represents the depth of the node *n* in hierarchy (conventionally we select the longest path from the node to the root to measure it). For the root, *l* (*root*) =0. For any two concepts in the hierarchal ontology, there is a *closest common parent CCP*. The distance between two concepts will be defined and obtained by the *milestones* of them and their *closest common parent* is computed as follows:

$$Dis_{milestone}(\mathrm{C_1}, \mathrm{C_2}) = Dis(\mathrm{C_1}, \mathrm{CCP}) + Dis(\mathrm{C_2}, \mathrm{CCP}) \tag{5}$$

$$Dis(\mathrm{C}, \mathrm{CCP}) = milestone(\mathrm{CCP}) - milestone(\mathrm{C}) \tag{6}$$

As we can see from the formula 5 and 6 above, the numerator of the milestone's calculation is predefined to 1/2 so that the distance between the two deepest nodes taking the root as their closest common parent will be 1. Then, the distance between other node pairs will be within 1. However, there is an exception that if the concept of a resource in the hierarchal ontology is a subclass of another concept, the distance will be set to 0, i.e. the similarity between these two concepts will be 1. And this is reasonable because the subclass is always a kind of superclass. We have adopted Milestone method for its intuitiveness and the simplicity of the implementation.

The Edge Counting Distance (*EC*) is implemented over the sub-sumption hierarchy that is computed by the Pellet DL reasoner [10]. An edge exists between two concepts C_1 and C_2 if $C_1 \sqsubseteq_d C_2 \vee C_2 \sqsubseteq_d C_1$, where $C_1 \sqsubseteq_d C_2$ represents that C_1 is a direct subclass of C_2. In the way of implementing the *EC* distance between two concepts, we summarized in the following three r_i rules, where $r_1 > r_2 > r_3$:

$$r_1: \ if \ C_1 = C_2 \vee C_1 \equiv C_2, \ then \ EC(C_1, C_2) = 0$$
$$r_2: \ if \ C_1 \sqcap C_2 \sqsubseteq \perp, \ then \ EC(C_1, C_2) = 1$$
$$r_3: \ \text{others: } EC(C_1, C_2) = Dis_{milestone}(C_1, C_2)$$

For instance, if there is a hierarchical relationship between two concepts C_1 and C_2 (r_3), then the EC distance is equal to the distance that can be computed using Milestone method according to the formula 5.

3.2 Functional Similarity

Functional Similarity (*FS*) depends on the DHL_{set} similarity of the Inputs-Outputs sets of two specifications, with respect to the following points:

- All the advertisement inputs are satisfied by the query inputs.
- All the query outputs are satisfied by the advertisement outputs (this is called signature matchmaking).

DEFINITION 3. The Functional Similarity between two specifications A and Q is normalized within the range $[0..1]$, with respect to the Web service filter W_f, that is:

$$FS(A, Q, W_f) = \sqrt{DLH_{set}(I_q, I_a, \mathrm{F}_I) * DLH_{set}(O_a, O_q, \mathrm{F}_O)} \qquad (7)$$

In the above formula, the geometric mean is used, instead of the arithmetic mean, as a Web service should be excluded if either of its inputs or outputs similarity equal to zero.

In order to manage various degrees of relaxation during Inputs-Outputs matchmaking, the Functional Similarity uses a Web service filter W_f which sets the values of the hierarchical filter sets F_I and F_o in formula(7). The hierarchical filters we use are Exact (W_e), Plugin (W_p), Subsume (W_{su}), and Sibling (W_{sb}). The following is the relationship of F_I and F_o filter sets:

- $W_e \rightarrow F_I = F_O = \{e\}$. In this filter, two specifications are matched only if they have the same or to equivalent concepts of their Inputs-Outputs.
- $W_p \rightarrow F_I = \{e, p\} \wedge F_O = \{e, su\}$. This filter represents that is a specification A can be used instead of a Q specification. That means all the inputs of the advertisement should be equivalent or sub-classes of the query's inputs, and all the outputs of the query should be equivalent or super-classes of the advertisement outputs. In other words, two specifications are matched only if they have similar or to equivalent concepts in their Inputs-Outputs.

- $W_{su} \rightarrow F_I = F_O = \{e, p, su\}$. This filter relaxes even more the matching level and the advertisement is allowed to have: more general inputs than the query and more general outputs than the query.
- $W_{sb} \rightarrow F_I = F_O = \{e, p, su, sb\}$ .This is the most relaxed filter that allows the existence of sibling relationships among Inputs - Outputs concepts. The priority of Web service filter's relaxation is $W_e < W_p < W_{su} < W_{sb} \rightarrow F_I = F_O = \{e, p, su, sb\}$.

3.3 Non-Functional Similarity

In our work, the Non-functional Similarity (NFS) is defined using a function denote dt for computing the similarity of two data type values, where dt (a, b) is normalized in the range [0..1].

DEFINITION 4. Assume a set T_d of the common data type of A and Q specifications with $T_d \neq \varnothing$. The Non-functional Similarity is normalized in the range in the range $[0..1]$, with respect to the function *dt* as follows:

$$NFS(A,Q) = \frac{\sum_{\forall d \in T_d} dt\left(ID_a.d, ID_b.d\right)}{|T_d|} \quad (8)$$

If $T_d \neq \varnothing$, then $NFS(A,Q) = 1$. The *dt* function calculates the similarity of two data type value sets (V_A and V_Q) by comparing directly their values. For example in String, the Jaro-Winkler similarity measure [12] can be used and its value is normalized in the range $[0..1]$ and the value of two strings can be computed as:

$$dt\left(V_A, V_Q\right) = \max_{\substack{\forall V_a \in V_A \\ \forall V_b \in V_B}} [\text{str}(v_a, v_q)] \quad (9)$$

Then the most similar matching combination value can be extracted. In the other data types, the two value sets can be compared directly and a value in the range will be returned using the following formula:

$$dt\left(V_A, V_Q\right) = \frac{|V_A \cap V_Q|}{|V_A \cup V_Q|} \quad (10)$$

In our study the name of services will be the values we are going to be compared in order to obtain the Non-functional Similarity using Jaro-Winkler similarity measure.

3.4 The Overall Similarity

The *TS*, *FS*, and *NFS* similarities together are considered as the overall similarity Sim_{all} of the two A and Q specifications.

DEFINITION 5. Assume there are two specifications A and Q . Their similarity Sim_{all} is the triple $\langle TS, FS, NFS \rangle$ of their Taxonomical, Functional and Non-functional similarities, then,

$$Sim_{all}\left(A, Q, F_T, W_f\right) = \left\langle TS\left(A, Q, F_T\right), FS\left(A, Q, W_f\right), NFS\left(A, Q\right)\right\rangle \quad (11)$$

The aggregation of the triple similarity into a single value is computed as the weighted mean Sim_{weight} of the three similarities according to user requirements, as follows:

$$\mathrm{Sim}_{\mathrm{weight}} = \frac{\mathrm{a.\ TS} + \mathrm{b.\ FS} + \mathrm{c.\ NFS}}{\mathrm{a} + \mathrm{b} + \mathrm{c}} \quad (12)$$

where *a, b*, and *c* are normalized weights in the range $[0..1]$.

The overall matchmaking algorithm of a *Q* specification with a set of *A* specifications is described in Algorithm 1. The algorithm introduces the complete set of the advertisements, implementing a two-phase filtering based on the taxonomical and functional requirements. The motivation is to exclude first the advertisements that do not taxonomically satisfy with the required query, to find the highest functional similarity procedure to be applied on a fewer set of advertisements.

Algorithm.1. Overall Matchmaking Algorithm

Input: Q Query specification, Cloud C contains a set of the advertised services S_A, Taxonomical Filter set F_T, Web service filter W_f, weights *a, b, c,* l_t, f_t are threshold in the range $[0..1]$.

Output: Sets that contains the matched services categorized in each of the 16 grouping filters.

1. *matchedSet*←*null*
2. *for* every A_i in S_A in the cloud *C do*
3. t_S←TS(*Ai,Q,*F_T) // calculating taxonomical similarity
4. *if* t_S=0 or t_S<l_S *then*
5. *continue*;
6. *else*
7. f_S←FS(*Ai,Q,*W_f) // calculating functional similarity
8. *If* f_S=0 or fs<l_f *then*
9. *continue*;
10. *else*
11. w_f =*getW*$_f$(A_i,*Q)*
12. *nfs=NFS*(A_i,*Q)* // calculate non-functional similarity
13. *end if*
14. *end if*
15. Sim_{weight}=(*a.*t_S+*b.fs+c.nfs)/a+b+c* // calculating overall similarity
16. *matchedSet*←*matchedSet* $\cup\{\langle A_i, Sim_{weight}, w_f \rangle\}$
17. *end for*
18. *if matchedSet is not null*
19. $Array[16] \leftarrow [\varnothing, \varnothing, \ldots \varnothing]$ // an array with 16 sets.
20. *for* all t← $\langle A_k, Sim_{weight}, w_f \rangle \in matchedSet$ do
21. *x-*w_f ←*getClassifiedFilter* (A_i,*Q)*
22. $Array[x\_w_f] \leftarrow Array[x\_w_f] \cup \{t\}$
23. end for
24. for *i* ← 1 to 16 do
25. $Array[i] \leftarrow Sort_{sim,desc}(Array[i])$ // sort the result and put it in groups
26. end for
27. return *Array*

In algorithm above, for every advertisements A_i included in the set S_A in the cloud C set (in line 2), it calculates first the TS (in line 3). Then if TS equals to 0 or less than the threshold I_t (at line 4), then A_i will be passed over and the algorithm will be continued with the next advertisement. The I_t threshold sets the minimum required similarity value of the taxonomical concepts provided by the user and it used to define and modify the degrees of taxonomical relaxation. The algorithm calculates the FS (at line 7) by utilizing the Web service filter W_f and if *FS* equals to 0 or less than the threshold I_f , which does similar work as I_t but in FS, (at line 8), then A_i will be passed over and the algorithm will be continued with the next advertisement. If the FS value is acceptable, then the result is the Web service filter W_f that meets the A and Q specifications (at line 11).

The NFS will be calculated (at line 12), and then the Overall Similarity Sim_{weight} value will be calculated (at line 15) and it will be added to the set matches of the matched specifications as a triple of the matched specification, the Sim_{weight} value and the Web service filter W_f (at line 16).

The matched advertisements are returned according to the classified filter (at line 21). In other words, each triple of the matchedSet will be added to *Array* array (as 21- 22 lines) that contains 16 sets, one set for each of the 16 grouping filters we have mentioned early (xW_e, xiW_e, etc.). At the end, the Simweight of the triples (at line 25) orders all sets and *Array* will be returned beck.

4. Experiment and Evaluation

In order to prove the efficiency of OWLS-CSM, we test and compared its results to two similar matchmakers OWLS-SLR [13] and OWLS-MX [9], using the OWLS-TC version 2.2 revision 2 collection [13] included 1,007 OWL-S advertisements and 29 queries.

OWLS-SLR matchmaker has been chosen to compare with our agent due it has been widely used to match services described in owl ontology and OWLS-MX matchmaker as it is a popular matching agent, which has been exploited with the OWLS-TC collection. We used the M4 configuration of OWLS-MX as the best configuration according to the study in [4]. We sat high user preferences for functional and non-functional matching; a= 0.9, b= 1 and c= 0.9. The experiments ran on a Windows 7 PC with i7 3.4 GHZ processor, maximum JAVA heap size of 1200 MBs.

4.1 Precision and Recall

In our study, OWLS-CSM uses relevance sets of the collection in order to accomplish the recall and the precision comparisons. A taxonomy-based collection is built in order to test the performance with regards to the Taxonomy Similarity. As the study [12], *OWLS-MX* can handle only direct Profile instances. Figure.1 illustrates the average precision and figure 2 the average recall of all queries for *OWLS-CSM, OWLS-SLR and OWLS-M4*, according to the Web service filter that was adopted. We have ignored the subsumed-by filter of *OWLS-MX* for presentation purpose.

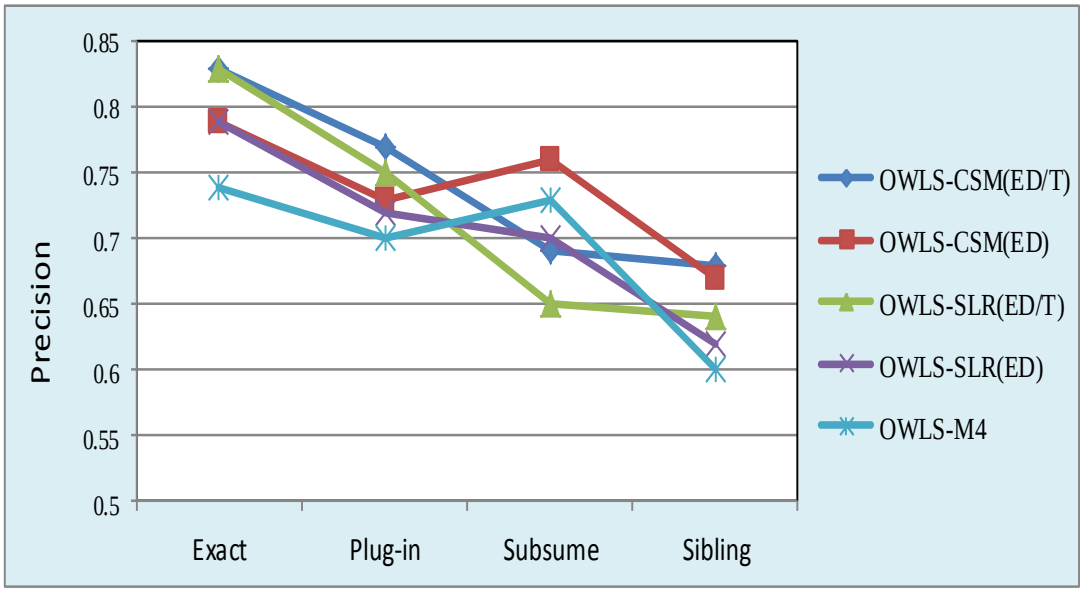

Figure 1. Precision

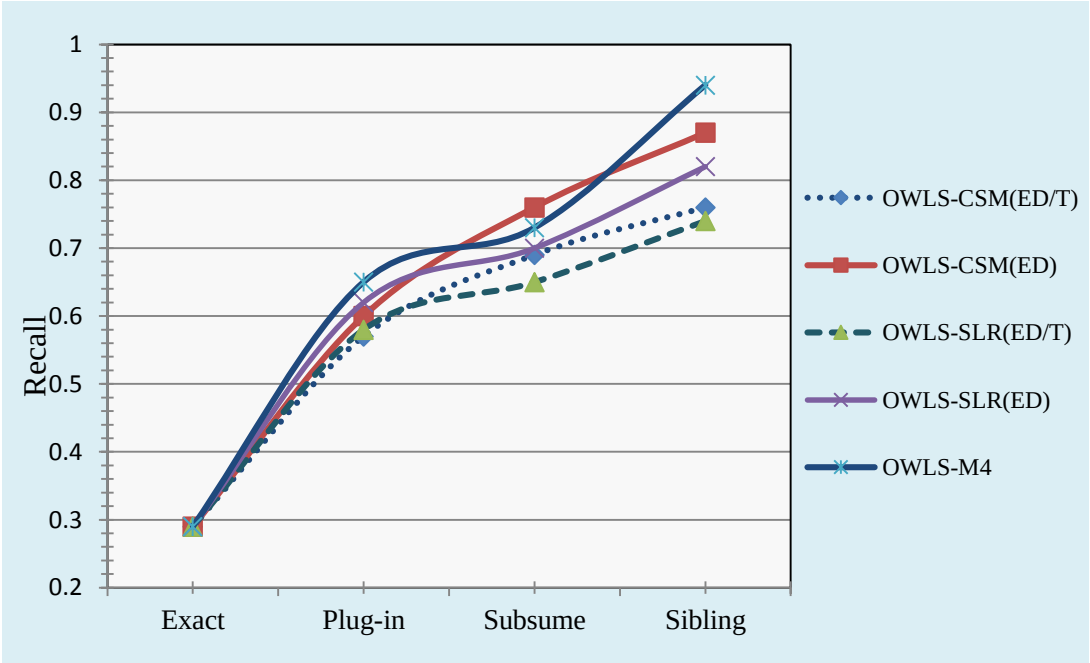

Figure 2. Recall

As we can see in figure 1, *OWLS-M4* generally has better precision than *OWLS-CSM* and *OWLS-SLR* in the collection without Profile taxonomy, proving that the filter definitions that it follows, fits better to the specific collection. However, by performing domain-oriented discovery, *OWLS-CSM* shows better performs than *OWLS-SLR* and *OWLS-M4*, explaining the advantage of a domain-oriented approach to Semantic Web Services discovery.

The recall of *OWLS-CSM* and *OWLS-SLR* with the Profile taxonomy are a little bit lower than the one without taxonomy as some results do not pass the $F_{T=\{e\}}$ filter set.

5. Conclusions

In this work, we present OWLS-CSM, which is an OWL-S Service Profile based framework for web service discovery in cloud environment. In OWLS-CSM, concepts of services are described and distributed in a hierarchal ontology, and that in turn facilitates service matchmaking.

The matchmaking process goes through three types of similarities, namely taxonomical similarity, functional similarity and non-functional similarity. The core method adopted in the matchmaking algorithm of OWLS-CSM is called Milestone method that computes the similarity of concepts of services according to their position in the hierarchal ontology; it also offers better results than previous works. The overall similarity, which is computed along with the aggregated value of the three mentioned similarities, represents the final values of the matching of the advertisements with the required query.

References

[1] Borko Furht. *Handbook of Cloud Computing*. Springer. London. 2010.
[2] Wei, Dengping, Ting Wang, Ji Wang, Abraham Bernstein. SAWSDL-iMatcher: A customizable and effective Semantic Web Service matchmaker. *Web Semantics: Science, Services and Agents on the World Wide Web.* 2011; 9(4): 402-417.
[3] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D.L. Mcguinness, E. Sirin, N. Srinivasan. Bringing Semantics to Web Services with OWL-S*. World Wide Web.* 2007; 10(3): 243-277.
[4] G.Meditskos, N. Bassiliades. Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S*. IEEE transactions on knowledge and data engineering.* 2010; 22(2): 278-290.
[5] Chantal Cherifi, Vincent Labatut, Jean-François Santucci. On Flexible Web Services Composition Networks. *Digital Information and Communication Technology and Its Applications.* 2011; 166: 45-59.
[6] P. Wang, Z. Jin, L. Liu, G. Cai. Building Toward Capability Specifications of Web Services Based on an Environment Ontology. *IEEE Trans. Knowledge and Data Eng*. 2008; 20(4): 547-561.
[7] M. Li, B. Yu, OF. Rana, Z. Wang. Grid Service Discovery with Rough Sets. *IEEE Trans. Knowledge and Data Eng.* 2008; 20(6): 851-862.
[8] D. Bianchini, VD. Antonellis, M. Melchiori, D. Salvi. *Semantic-Enriched Service Discovery*. Proc. Int'l Conf. Data Eng. Workshops. 2006: 38-44.
[9] M. Klusch, B. Fries, K. Sycara. OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services. *Web Semantics: Science, Services, and Agents on the World Wide Web.* 2009; 7(2): 121-133.
[10] H. Dong, FK. Hussain, E. Chang. Semantic Web Service matchmakers: state of the art and challenges. *Concurrency and Computation: Practice and Experience*. 2013; 25(7): 961–988.
[11] Naji Hasan.A.H, Gao Shu, Al-Gabri Malek, Jiang Zi-Long. An Efficient Approach based on Hierarchal Ontology for Service Discovery in Cloud Computing. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(4): 2905-2913.
[12] Chantal Cherifi, Vincent Labatut, Jean-François Santucci. On Flexible Web Services Composition Networks. *Digital Information and Communication Technology and Its Applications.* 2011; 166: 45-59.
[13] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz. Pellet: A Practical OWL-DL Reasoner. *J. Web Semantics*. 2007; 5(2): 51-53.