An Improved Approach for Semantics Based Filtering and Selection of Services Using Bipartite Matching

P. Mahesh
Computer Science and Engineering, PSN College of Engineering and Technology,
maheshmario@gmail.com

Abstract:

In today's world, a vast number of web services are available, providing a variety of services. As a result service discovery becomes an important task. Large number of services made discovery of web services is a critical one and also discovery process has the scalability issues when the number of services increases. In this work, we address the issue of scalability in web service discovery process by adding a preprocessing stage. Our approach to preprocessing stage involves filtering the services by checking the details of descriptions; semantic-based web service discovery involves Bipartite Matching and semantic enhancement of the service request. Our approach proposes a preprocessing stage based on the details of descriptions that filter service repositories with enhanced service request. We propose a solution for achieving functional level Bipartite Matching based on an Ontology framework. The semantic enhancement of the service request achieves a better matching with relevant services. The service request enhancement involves expansion of additional terms (retrieved from WordNet) that are deemed relevant for the requested functionality. An efficient matching of the enhanced service request with the filtered services is achieved utilizing Bipartite Matching. Our Experimental results validate the effectiveness and feasibility of the proposed work.

Keywords: Automated Web Service Discovery, Bipartite Matching, Semantics-based Filtering

1. INTRODUCTION

A large number of Web Services structure a Service Oriented Architecture (SOA) and facilitate the creation of distributed applications over web. Loose coupling is an important principle underlying Service Oriented Architecture. One aspect of loose coupling is the ability to invoke a service with little (or no) knowledge about it. The *publish-find-bind* architecture is intended to facilitate this process. Service providers create WSDL [9] descriptions and publish them to UDDI [8] registries. Clients search the registry to locate providers of the desired service. Today, in most cases, the WSDL is compiled into client-stubs and the service is invoked. This approach, however, has several limitations.

The WSDL is a specification of the messaging syntax between the client and the provider. It is necessary for a human to interpret the WSDL and then invoke the client-stub with the correct parameters. The search capabilities of UDDI are limited to a syntax-based search. A client can search the registry for a *string* in the service description or it can search the service classification hierarchy (like NAICS [3]) in the TModel. Neither of these techniques is sufficient, for a client, to be able to autonomously choose a service provider and invoke it without human intervention.

In order to overcome these limitations, techniques for semantic description and matchmaking of services have been proposed in recent literature. These techniques use semantic concepts from *Ontologies* to describe the Inputs, Outputs, Pre-conditions and Effects (IOPE) of a service. Current Web service matchmaking approaches check the capabilities of the requested Web service against the capabilities of all advertised Web service advertisements in the registry. As the number of advertised Web services in the registry can be huge, the process of checking all advertised Web services against a single client query can be time consuming.

This work proposes a semantic-based Web service registry filtering mechanism that takes the responsibility of narrowing down the number of Web service descriptions to be checked in detail to the number of only the relevant advertisements to the client request. The proposed filtering mechanism picks the advertisements that are relevant to the client request and ignore, from an early stage and before checking the details of the descriptions, the advertisements that are

IJERT

not able to satisfy the client request. This work makes use of current web technologies like OWL and OWL-S to describe services and define ontologies.

Initially the Semantic Matchmaking Algorithm is proposed by M. Paolucci et al [21]. This work has considerable interest in this algorithm because it has been cited extensively in recent literatures and several subsequent proposals ([13], [22], [16], [14]) are based on it. M. Paolucci's algorithm [21] adopts a greedy approach towards matching the concept-lists. For example, in the case of output matching, for each concept in the $Query_{out,}$, It determines a corresponding concept in $Advt_{out}$ to which it has a maximum degree of match. Once all such max-matchings are computed, the minimum match amongst them is the overall degree of match between the advertisement and the client request. M. Paolucci's algorithm has been tested that the *Greedy* algorithm indeed generates false positive and false negative outcomes. On the other hand, the outcomes of the *Bipartite matching* are identical to that of the *Brute Force* reference model. So, the proposed work utilizes an efficient matchmaking algorithm based on bipartite graphs.

The rest of the paper organized as follows: Section 2 provides background material and Section 3 presents an overview of the proposed approach. Section 4 provides a detailed discussion on semantic enhancement of service request. The detailed description for parameters based service repository filtering is presented in Section 5. Section 6 includes a discussion on Bipartite Matching. The implementation details of the proposed approach and our evaluations presented in Section 7. We present the related work in Section 8. Finally, conclusion and future work are presented in Section 9.

2. BACKGROUND

This section provides a brief background of the methodologies utilized for parameters based service repository filtering and selection of services using Bipartite Matching. We briefly discuss the parameters for ranking the services using Jaccard Similarity in the context of semantic based service repository filtering. We also briefly discuss the Bipartite Matching algorithm used for service selection.

2.1 Jaccard Similarity

Jaccard Similarity is a simple and very intuitive measure of document to document similarity. It is defined as follows:

$$similarity(A,B) = n(A \sqcap B) / n(A \cup B) \quad (1)$$

Using the Inclusion-Exclusion Principle, this reduces to:

$$similarity(A,B) = n(A \sqcap B) / n(A) + n(B) - n(A \sqcap B) \quad (2)$$

Where: $n(A \sqcap B) = \Sigma \min(Ai, Bi)$

$n(A) = \Sigma Ai$

$n(B) = \Sigma Bi$

for i = [0..n-1], where n = number of terms in our term-document matrix.

2.2 Bipartite Graphs and Matching

Bipartite Graph:

A *Bipartite Graph* is a graph G = (V, E) in which the vertex set can be partitioned into two disjoint sets, $V=V0 \cup V1$, such that every edge $e \in E$ has one vertex in V0 and another in V1.

Matching:

A *matching* of a bipartite graph G = (V, E) is subgraph G' = (V, E'), $E' \subseteq E$, such that no two edges $e1, e2 \in E'$ share the same vertex. We say that a vertex v is m*atched* if it is incident to an edge in the matching. Given a bipartite graph G = (V0 + V1, E) and its matching G', the matching is *complete* if and only if, all vertices in V0 are matched

3. OVERVIEW OF PROPOSED APPROACH

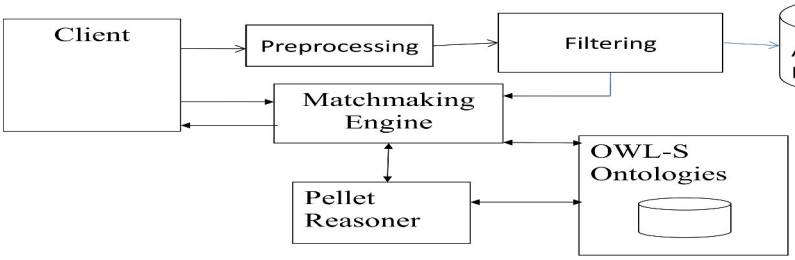

Fig. 1 Automated Service Discovery

Fig. 1 illustrates the key steps of the proposed approach for automated service discovery using bipartite matching. The first step of proposed approach involves enhancement of the service request. This step involves two tasks: 1) enhancement of the web service request with WordNet terms, 2) filtering of the set of web services based on the input, output and description

parameters of the service. The purpose of this filtering is to select a set of services from the available set of advertisements. The next step deals with the selection of web services for the enhanced service request with the filtered set of advertisements.

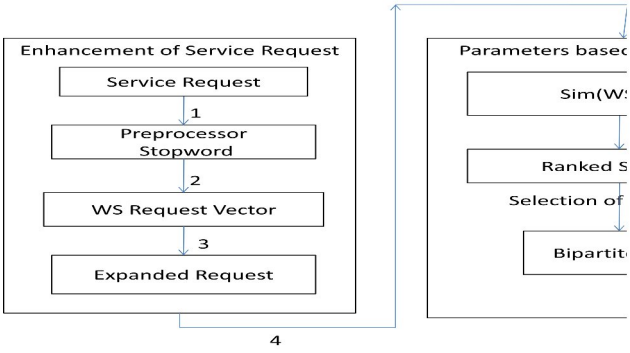

Fig. 2. Automated Service Discovery Components.

Fig. 2 illustrates the main components of the overall system that performs two key steps related to automated service discovery

The WordNet guided enhancement of Service Request, is illustrated in steps 1 to 4, a key part of this process involves enhancing the service request. Step 1 collects the input file from the client. Step 2 demonstrates the initial processing of the service request and its transformation to service request vector. Our approach for enhancement of service request utilizes WordNet to enhance the service request. In step 3 we expand the request vector with relevant terms from WordNet dictionary. Step 4 indicates the formation of enhanced service request vector.

The enhancement of service request is followed by service filtering from the relevant group of services. This is achieved by parameters based service filtering as illustrated in steps 5 and 6 of Fig. 2. Parameters based service filtering includes narrowing the set of appropriate services matching the service request based on parameters, i.e., input, output and description. The filtered set of services is then matched against an enhanced service request as part of Selection of Web Services, as illustrated in steps 7 and 8 of Fig. 2. Parameters based filtering of web services begins with the representation of the web service parameters as a vector in which each entry records the terms of the operations' input and output. The set of related web services is represented by a collection of such vectors. Next we rank this web service collection by applying Jaccard Similarity, as illustrated in step 6.

The filtered set of web services is then matched against an enhanced service request as part of Selection of Web Services, as illustrated in steps 7 and 8. Our approach for bipartite matching utilizes ontology linking to the filtered services. We provide ranking of services according to the terms of input and output parameters. However, considering both the input and output parameters we prioritize output matching as the outputs of a service are more important for client of a matchmaker. The following equation is used to for this weight calculation:

$$S_{final} = W_{output} * S_{output} + W_{input} * S_{input} \quad (3)$$

In the above equation, S_{input} and S_{output} represent the similarity score considering only the input parameters and output parameters respectively. W_{input} and W_{output} represents the weights for the input and output similarity score, where they are fixed to 0.4 and 0.6 after several runs of matchmaker considering the expected outcome. Finally, S_{final} represents the final score of similarity considering both the input and output parameters.

4. ENHANCEMENT OF SERVICE REQUEST

In our approach we begin with the enhancement of web service request wherein we combine the WordNet terms, following the service request vector building process. For each term in the service request vector, a collection of similar words added from the WordNet dictionary. Additional terms are added based on semantic relationship between the terms in the request vector.

4.1 Web Service Request Preprocessing

The service request OWL-S file is parsed and preprocessed. Preprocessing includes: the removal of markups, translation of upper case characters in to lower case, punctuation removal and white space used as a term limiters and stopword removals. The outcome of this preprocessing is in a term vector.

4.2 Service Request Expansion

To achieve semantic enhancement we utilize an approach that augments the WordNet noun database [24],[15]. The add step extends each service vector by additional WordNet elements. The generateEnhancedRequest Algorith (Fig. 3) gives the details. Step 2 of Fig. 2 executes the modification of the web service request vector.

IJERT

Algorithm: generateEnhancedRequest

Input: Web Service Request (OWL-S)

Output: Enhanced Service Request SR_e

1: begin

2: for each Web Service Request SR do

3: /* Preprocess Service Request SR */

4: // Removal of Stopwords

5: end for

6: for each Web Service Vector wsi ϵ WS do

7: for each term tj ϵ ti' && tj ϵ T do

8: Extract WordNet Element ej

9: append terms in ej to SR_e

10: end for

11: end for

12: return SR_e

13: end

5. PARAMETERS BASED SERVICE FILTERING

The next step is service filtering from the relevant category of services using Parameters Based Service Filtering. Web service parameters, i.e., input, output and descriptions, aid service filtering through narrowing the set of appropriate services matching the enhanced service request. Sections 5.1 and 5.2 provide a detailed discussion of each of the steps for parameters based service filtering.

5.1 Service Parameters Retrieval

As discussed earlier, the web service description is provided in the OWL-S document. For retrieving the relevant service parameters, the corresponding OWL-S document is processed to extract the associated operation parameters by retrieving all the terms under <profile:hasInput> and <profile:hasOutput> tags. The OWL-S preprocessing also includes stoplist removal. Final outcome of this step is Service Description Vector for the published services using OWL-S.

5.2 Service Collection Pruning

A large number of services available in the Registry which are relevant and irrelevant to the Service request. So irrelevant services need to be discarded from the search registry. The pruning of the Service Collection is based on eliminate the services that have zero ranking between the enhanced request. This is illustrated in the PruneAdvertisements algorithm.

Algorithm: PruneAdvertisements

Input: Advertisement Collection present in the Domain registry

Output: Relevant Service Description Set

1: begin

2: for Advertisement in Domain registry do

3: /* form service description vector after stoplist removal*/

4: Rank $_{advt}$= SIM(Advt, SR_e)

5: if Rank $_{advt}$ >0

6. append Advt to pruneSet

7: end for

8: end

6. SELECTION OF SERVICES

The parameters-based filtered set of web services is then matched against an enhanced service request as part of Semantic Bipartite Matching. Search() procedure accepts a Query as input and tries to match it with each advertisement in the repository. A match is computed for both, output and input concepts. If the match is not a *Fail*, it appends the advertisement to the result set. Finally the sorted result set is returned to the client.

The match() procedure accepts two concept-lists as inputs and constructs a bipartite graph using them. It then invokes a hungarian algorithm [17], [20] to compute a *complete matching* on the graph. The match() procedure is invoked twice in search(). The order of Query and Advertisement in each call is however swapped.

The computeWeights() function computes the values of w1,w2,w3, depending on the number of concepts in V0. It uses the formulae presented in the section *"Computation of Edge Weights"* to compute the values. The degreeOfMatch() function is a call to the reasoned in order to determine the relationship between the two concepts a and b.

Algorithm: Search(Query)

1: Result = Empty List

IJERT

2: **for** each Advt in Repository **do**
3: outMatch = match(Queryout, Advtout)
4: **if** (outMatch = Fail) **then**
5: Skip Advt. Take next Advt.
6: **end if**
7: inMatch = match(Advtin, Queryin)
8: **if** (inMatch = Fail) **then**
9: Skip Advt. Take next Advt.
10: **end if**
11: Result.append(Advt, outMatch, inMatch)
12: **end for**
13: return sort(Result)

Algorithm: Match(List1, List2)
1: Graph G = Empty Graph (V0 + V1,E)
2: V0 ← List1
3: V1 ← List2
4: (w1, w2, w3)← computeWeights(|V0|)
5: **for** each concept a in V0 **do**
6: **for** each concept b in V1 **do**
7: degree = degreeOfMatch(a, b)
8: **if** degree 6= Fail **then**
9: Add edge (a, b) to G
10: **if** (degree = Exact) **then** w(a, b) = w1
11: **if** (degree = Plugin) **then** w(a, b) = w2
12: **if** (degree = Subsume) **then** w(a, b) = w3
13: **end if**
14: **end for**
15: **end for**
16: GraphM = hungarianMatch(G)
17: **if** (M = null) **then**
18: No *complete matching* exists
19: **return** Fail
20: **end if**
21: Let (a, b) denote Max-Weight Edge in G
22: degree ←degreeOfMatch(a, b)
23: **return** degree

7. EXPERIMENTAL EVALUATION

For evaluating the performance of the algorithms which are proposed in previous sections, we semi-automatically built a test collection of 1007 OWL-S service descriptions which use a reference ontology with 4694 concepts. Most of the structure of the taxonomy of the referenced ontology is also based on the hierarchy of the words in WordNet. We then used *Recall* and *Precision* metrics for evaluating the results. *Recall* is the extent to which the method retrieves *all* of the similar services while *Precision* is the extent to which the method retrieves *only* the real similarities.

The effectiveness of the Compound Similarity is shown by conducting three set of experiments: (1) Greedy Match Making algorithm by Paolucci; (2) The search using the Text Based Matching algorithm with the available service descriptions only; (3) The search using Bipartite matching. The results are shown in the below table.

IJERT

Method	Precision	Recall
Paolucci Matching	51.4%	82.3%
Text Based Matching	100%	29.5%
Bipartite Matching	56.25%	88.2%

8. RELATED WORK

Our approach has similarities to existing approaches [4], [1], [2], [19] of natural language processing techniques that address the text part of the challenge in content-based image retrieval (CBIR). These approaches, however, were used in isolation to one another.

In [12], Sassen et al. describe the SeCSE approach for architecture time service discovery that is based on ontologies that are validated, easy to use, complete, and widely accepted in domains. In contrast to this, our approach begins with the description of an ontology framework that includes upper ontologies, e.g., SUMO and more descriptive domain and application related ontologies. We propose a linked

ontology structure for a wide-ranging description of domain semantics.

Phatak [22] adds *ontologymappings* and QoS constraints to the algorithm from [21]. Choi [13] expands the search scope of [21] by the use of analogous terms from an ontology server. It also makes use of a rule-based search in order to apply user restrictions and to rank search results. It computes fine-grained rankings by the use of *concept similarity* (horizontal and vertical closeness between concepts). Jaeger [16] extends the work from [21] by using matching over the *properties* and over the *Service Profile* hierarchy. It offers a better (fine-grained) ranking scheme as compared to [21].

The usage of synonyms does not capture the overall semantics of the domain and application functionality. However, our approach utilizes concepts extracted from domain ontology. These extracted concepts account for relationships between the domain objects and provide a comprehensive coverage for the underlying semantics for both the domain and the application functionality. Our approach appends the syntactic service description with relevant semantic terms. This enables uniform combination of syntactic and semantic matching rendering our approach more generalizable for overall service matching and requiring minimal human interaction.

9. CONCLUSION AND FUTURE WORK

The proposed approach presents an integrated approach for automated service discovery. Specifically, the approach addresses two major aspects related to semantics based discovery: Semantics based filtering and selection based on bipartite matching. For semantics based filtering, this work proposes a WordNet guided filtering of advertisements leads to reduce the scalability issues in MatchMaking process. For Selection based on Bipartite matching, we employ ontology linking to discover services in functional level. Our experiments show that this lead increased precision levels, recall levels and the relevance scores of the retrieved services.

In future, proposed work can be extended to web service composition. Typically, multiple services have to be discovered so that they together match a service request. It should be possible to utilize ontologies, and explicitly return the sequence of individual service invocations to be performed in order to achieve the desired composite service. When no full match is possible, a flexible matching approach could be created to return partial matches and/or suggest additional inputs that would produce a full match by capturing the dependencies among the matched services. This has several interesting research issues. Another avenue for future work is to create an interactive, intelligent service composer that is semantically guided to locate the target service components step by step. We also intend to extend our ontology framework and investigate additional mapping tools to better express a service request to search for relevant concepts.

REFERENCES

[1] Adcock, J.; Girgensohn, A.; Cooper, M.; Liu, T.; Wilcox, L. and Rie, E.(2004). FXPAL Experiments for TRECVID. Proc. TRECVID.

[2] Agrawal, R.; Imielinski, T. and Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. Proc. ACM SIGMOD Int'l Conf. Management of Data.

[3] North American Industry Classification System, http://www.naics.com/.

[4] http://www.musclenoe.org/research/sci_deliv_pub/D5.1_WP5_SoA_RevisedVersion_sept05.pdf, 2012.

[5] Pellet: An OWL DL Reasoner. *http://pellet.owldl.com/*.

[6] Protege: Ontology Editor and Knowledge-base framework.*http://protege.stanford.edu/*.

[7] RacerPro: OWL Reasoner and Inference Server for the Semantic Web. *http://www.racer-systems.com/*.

[8] Universal Description Discovery and Integration (UDDI), http://uddi.org/.

[9] Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl/.

[10] Ankolekar, A.et al. (2002) DAML-S Coalition. DAML-S: Web Service Description for the Semantic Web. *ISWC*.

[11] Antoniou, G. et al. (2003) Web Ontology Language: OWL. *Handbook on Ontologies in Information Systems*.

[12] http://idcrue.dit.upm.es/biblioteca/mostrar.php?id=2154, 2012.

[13] Choi, O. et al. (2005). Extended Semantic Web Services Model for Automatic Integrated Framework. *NWESP*.

[14] Guo, R. et al. (2005). Capability Matching of Web Services Based on OWL-S. *Proceedings of 16th International Workshop on Database and Expert Systems Applications*.

[15] http://reliant.teknowledge.com/DAML/SUMO.owl, 2008.

[16] Jaeger, M. et al. (2004). Ranked Matching for Service Descriptions using DAML-S. *Proceedings of CAiSE'04 Workshops*.

[17] Kuhn, H. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly*.

[18] Martin, D. et al. (2004) OWL-S: Semantic Markup for Web Services. *Technical Report, Member Submission, W3C*.

[19] Kher, M.; Brahmi, D. and Ziou, D. (2004). Combining Visual Features with Semantics for a More Efficient Image Retrieval. Proc. 17th Int'l Conf. Pattern Recognition (ICPR '04).

[20] Nedas, K. (2005) Implementation of Munkres-Kuhn (Hungarian) Algorithm. *http://www.spatial.maine.edu/ kostas*.

IJERT

[21] Paolucci, M. et al. (2002). Semantic Matching of Web Service Capabilities. *Springer Verlag, LNCS, International Semantic Web Conference*.

[22] Phatak, J. et al. (2005). A Framework for Semantic Web Services Discovery. *WIDM*.

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

[24] Niles, I. and Pease, A. (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. Proc. IEEE Int'l Conf. Information and Knowledge Eng. (IKE '03).

IJERT