[Post print version, please cite as] Fatma Ezzahra Gmati, Nadia Yacoubi Ayadi, Salem Chakhar: Parameterized Algorithms for Matching and Ranking Web Services - (Short Paper). OTM Conferences 2014: 784-791. DOI: 10.1007/978-3-662-45563-0_50

Parameterized Algorithms for Matching and Ranking Web Services

short paper

Fatma Ezzahra Gmati[1], Nadia Yacoubi-Ayadi[1], and Salem Chakhar[2,3]

[1] National School of Computer Sciences, University of Manouba, Tunis, Tunisia
{fatma.ezzahra.gmati,nadia.yacoubi.ayadi}@gmail.com
[2] Portsmouth Business School, University of Portsmouth, Portsmouth, UK
[3] Centre for Operational Research and Logistics, University of Portsmouth, Portsmouth, UK
salem.chakhar@port.ac.uk

Abstract. The paper presents two parameterized and customizable algorithms for matching and ranking Web services. Given a user query and a set of available Web services, the matching algorithm performs a logic-based semantic matchmaking to select services that functionally match the query and maintains those which fully verify the constraints specified by the user. The ranking algorithm takes as input the matching Web services, assigns to each one a score in the range 0-1 and finally rank them based on the score values. The algorithms have been implemented, evaluated and compared to iSEM and SPARQLent. Results show that the algorithms behave globally well in comparison to these frameworks.

Keywords: Web Service, Service Composition, Semantic Similarity, Matchmaking, Service Ranking

1 Introduction

Web service matchmaking is a crucial issue within Web service composition [2][3]. It is related to the selection of the most appropriate one among the different candidate Web services. In this paper, we propose two algorithms for matching and ranking Web services. The matching algorithm performs a logic-based semantic matchmaking to select services which fully verify the functional requirements and the constraints specified by the user. The ranking algorithm first assigns to each matching Web service a score in the range 0-1. The scores, along with some user's preference information, are then used to rank the selected Web services.

We developed and implemented a prototype supporting both algorithms. We used the Semantic Matchmaker Evaluation Environment to evaluate the performance of the algorithms and compared them to iSEM [5] and SPARQLent [7]. Results show that our algorithms behave globally well in comparison to iSEM and SPARQLent.

The paper is structured as follows. Sections 2 and 3 detail the matching and ranking algorithms. Section 4 presents the performance analysis. Section 5 discusses some related work. Section 6 concludes the paper.

2 Matching Web Services

2.1 Similarity Measure

The similarity measure quantifies the semantic distance between the two entities participating in the match. The similarity measure, μ, of two service attributes is a mapping that measures the semantic distance between the conceptual annotations associated with the service attributes. It is defined as follows [4]:

$$\mu : A \times A \rightarrow \{\text{Exact, Plug-in, Subsumption, Container, Part-of, Fail}\},$$

where A is the set of all possible attributes. The definitions of semantic measures are given in [2][3][4]. A preferential total order is established on the above mentioned similarity maps: Exact $\succ$ Plug-in $\succ$ Subsumption $\succ$ Container $\succ$ Part-of $\succ$ Fail, where $x \succ y$ means that x is preferred over y. In this paper, we implemented the idea of [1] to compute the similarity degrees.

2.2 Functional Conjunctive Matching

A service S is defined as a collection of attributes that describe the service. Let $S.A$ denotes the set of attributes of service S and $S.A_i$ denotes each member of this set. Let $S.N$ denotes the cardinality of this set. Let S^R be the service that is requested, and S^A be the service that is advertised. A first customization of functional conjunctive matching is to allow the user to specify a desired similarity measure for each attribute. A second customization is to allow the user specifying which attributes should be utilized during the matching process, and the order in which they are considered. These two customizations can be supported through the concept of Criteria Table [4]. A Criteria Table, C, is a relation consisting of two attributes, $C.A$ and $C.M$. $C.A$ describes the service attribute to be compared, and $C.M$ gives the *least* preferred similarity measure for that attribute. Let $C.A_i$ and $C.M_i$ denote the service attribute value and the desired measure in the ith tuple of the relation. Let $C.N$ be the number of tuples in C.

Based on the concept of Criteria Table, a sufficient functional conjunctive match between services is defined as follows:

$$\begin{array}{l} \forall_i \exists_{j,k} (C.A_i = S^R.A_j = S^A.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i \\ \Rightarrow \text{SuffFuncConjMatch}(S^R, S^A) \quad 1 \leq i \leq C.N. \end{array} \tag{1}$$

This basic matching rule is extended in [2][3] to support functional disjunctive, functional generic, service-level and quality of service-based matching.

2.3 Matching Algorithm

The matching algorithm that follows from Sentence (1) is given in Algorithm 1. It loops on the available Web services set U and for each service S^A it proceeds as follows: (i) scans the Criteria Table and for each attribute identifies the corresponding attributes in S^R and S^A; (ii) computes the similarity degrees between

the corresponding attributes; and (iii) appends the service S^A and the corresponding similarity degree in the list `mServices` of matching Web services. The output of Algorithm 1 is the list `mServices`. Algorithm 1 applies to functional conjunctive matching. It can be extended to apply to other types of matching.

Algorithm 1: Matching

Input : S^R: Requested service; C: Criteria Table.
Output: `mServices`: Matching services.

```
1  U ⟵ {list of potential advertised services};
2  for (each service S^A in U) do
3      while (i ≤ C.N) do
4          while (j ≤ S^R.N) do
5              if (S^R.A_j = C.A_i) then
6                  Append S^R.A_j to rAttrSet;
7              j ⟵ j + 1;
8          while (k ≤ S^A.N) do
9              if (S^A.A_k = C.A_i) then
10                 Append S^A.A_k to aAttrSet;
11             k ⟵ k + 1;
12         i ⟵ i + 1;
13     bol ⟵ true;
14     while (t ≤ C.N ∧ bol) do
15         mu ⟵ μ(rAttrSet[t], aAttrSet[t]);
16         if (mu) ⪰ C.M_t) then
17             simDegrees[t] ⟵ mu;
18         else
19             bol ⟵ false;
20         t ⟵ t + 1;
21     if (bol) then
22         Append (S^A, simDegrees[1], ··· , simDegrees[C.N]) to mServices;
23 return mServices;
```

Inferring $\mu(\cdot,\cdot)$ by ontological parse of pieces of information into facts and then utilizing fast rule-based engines leads to $O(|R||F||P|)$ where $|R|$ is the number of rules, $|F|$ is the number of facts, and $|P|$ is the average number of patterns in each rule [4]. Hence, the complexity of Algorithm 1 is $O(nC.N \times S^A.N) + O(|U||R||F||P|) \simeq O(|U||R||F||P|)$ (since, as underlined by [4], the process of computing μ is the most "expensive" step of the algorithm).

3 Ranking Web Services

3.1 Computing of Web Services Scores

The scores of the Web services are computed based on the similarity measures of the matching attributes specified in the Criteria Table. We need to assign a numerical weight to every similarity degree as follows: Fail: w_1, Part-of: w_2,

Container: w_3, Subsumption: w_4, Plug-in: w_5 and Exact: w_6. In this paper, we assume that the weights are computed as follows:

$$w_1 \geq 0, \tag{2}$$

$$w_i = (w_{i-1} \cdot p) + 1, \quad i = 2, \cdots, p; \tag{3}$$

where p is the number of attributes. This way of weights computation ensures that a single higher similarity degree will be greater than a set of p similarity degrees of lower weights taken together. Indeed, the weights values verify the following condition: $w_i > pw_j$, $\forall i > j$. Then, the initial score of an advertised service S^A is computed as followers:

$$\rho(S^A) = \sum_{i=1}^{i=p} w_i. \tag{4}$$

The scores given by Equation (4) are not in the range 0-1. The following equation can be used to normalize the scores (U' is the set of matching Web services):

$$\rho'(S^A) = \frac{\rho(S^A) - \min_{K \in U'} \rho(S^K)}{\max_{K \in U'} \rho(S^K) - \min_{K \in U'} \rho(S^K)}. \tag{5}$$

3.2 Ranking Rule

The basic information used for ranking Web services are the scores. However, the use of the scores only may lead to the problem of ties. To avoid this problem, we may exploit the user's preference information given in the Criteria Table. Two measures can be used to avoid the problem of ties:

- *Difference between the desired similarities.* Let s be a similarity degree and denote by $Ord(s)$ the ordinal rank of s in respect to the other predefined similarity degrees. The definition of $Ord(\cdot)$ is as follows: Fail: 1, Part-of: 2, Container: 3, Subsumption: 4, Plug-in: 5 and Exact: 6. Let S^A be an advertised service and let $(s_1, \cdots, s_p)$ be its similarity degrees for attributes $A_1, \cdots, A_p$. Then, we define the function $Diff(\cdot, \cdot)$ as follows:

$$Diff(S^A, C) = \sum_{k=1}^{k=p} \{Ord(s_k) - Ord(C.M_k)\}. \tag{6}$$

- *Lexicographic selection.* Let S_i^A and S_j^A be two services and let $(s_1, \cdots, s_p)$ and $(s'_1, \cdots, s'_p)$ be their similarity degrees for attributes $A_1, \cdots, A_p$. Then, the lexicographic rule is defined as follows:

$$Lex(S_i^A) > Lex(S_j^A) \Leftrightarrow (\exists l)(\forall r < l)(s_r = s'_r) \wedge (s_l > s'_l). \tag{7}$$

The ranking rule is then stated as follows: use the score-based ranking and if there is a tie, apply the order-based ranking; if another tie occurs, apply the lexicographic selection; and if a further tie occurs, select the first service. Formally,

```
RR: if ρ'(S_i^A) > ρ'(S_j^A) then S_i^A ≻ S_j^A
    else if ρ'(S_i^A) = ρ'(S_j^A) then
            if Diff(S_i^A, C) > Diff(S_j^A, C) then S_i^A ≻ S_j^A
            else if Diff(S_i^A, C) = Diff(S_j^A, C) then
                 if Lex(S_i^A) > Lex(S_j^A) then S_i^A ≻ S_j^A
                 else if Lex(S_j^A) = Lex(S_i^A) then
                      else Select the first service
```

Assume that the matching algorithm has identified the services given in Table 1. This table also shows the initial and normalized scores computed through Equations (4) and (5), respectively. Table 2 summarizes the ranking process of services given in Table 1. The final ranking has been obtained by using successively the scores, the similarity difference and the lexicographic selection.

Table 1. Web services identified by the matching algorithm and their scores

Service S_i	Input	Output	Service category	$\rho(S_i)$	$\rho(S_i)'$
S_1	Exact	Subsume	Subsume	147	0.5
S_2	Plug-in	Exact	Subsume	174	1
S_3	Plug-in	Plug-in	Plug-in	120	0
S_4	Plug-in	Subsume	Exact	174	1
S_5	Subsume	Exact	Subsume	120	0.5

Table 2. Ranking process

Rule	Test	Ranking
Scores	$\rho'(S_i^A) > \rho'(S_j^A)$	$S_2 = S_4 \succ S_1 = S_5 \succ S_3$
Order difference	$\rho'(S_i^A) = \rho'(S_j^A) \wedge Diff(S_i^A, C) > Diff(S_j^A, C)$	$S_2 = S_4 \succ S_1 = S_5 \succ S_3$
Lexico. selection	$Diff(S_i^A, C) = Diff(S_j^A, C) \wedge Lex(S_i^A) > Lex(S_j^A)$	$S_2 \succ S_4 \succ S_1 \succ S_5 \succ S_3$

3.3 Ranking Algorithm

The proposed ranking process is given in Algorithm 2. It takes the list of matching services `mServices` as input and proceeds as follows: (i) computes, for each matching service, the initial score using function `ComputeScore`, which implements Equation (4); (ii) computes the minimum and maximum scores values and then, for each matching service, it computes the normalized score using Equation (5); and (iii) uses the ranking rule `RR` to sort Web services. The output of Algorithm 2 is an ordered list `mServices` of matching Web services. We note that $N + 2$ in Algorithm 2 indicates the index of the score value in `mServices`.

The complexity of the first *while* loop of Algorithm 2 is $O(pC.N)$. The computing of the minimum or maximum values of a list of n values is $O(n)$. The complexity of the two last *while* loops, which correspond to a comparison-based sorting algorithm, is at best $O(n \log n)$ and in worst case, it makes $O(n^2)$. Hence, the

overall complexity of Algorithm 2 in best case is $O(pC.N) + O(2n) + O(n \log n)$ and in worst case is $O(pC.N) + O(2n) + O(n^2)$.

Algorithm 2: Ranking

```
   Input  : mServices: Services list; N: Nb. of attributes; C: Criteria table; w: weights vector.
   Output: mServices: Ranked list of matching services.
 1 r ⟵ length(mServices);
 2 while (t ≤ r) do
 3 |   row ⟵ tth row in mServices;
 4 |   s ⟵ ComputeScore(row, w);
 5 |   Append s to row;
 6 |   Append row to mServices;
 7 a ⟵ min(mServices);
 8 b ⟵ max(mServices);
 9 while (l ≤ r) do
10 |   ns ⟵ (mServices[l, N + 2] − a)/(b − a);
11 |   mServices[l, N + 2] ⟵ ns;
12 while (i ≤ r) do
13 |   while (j ≤ r) do
14 |   |   row_i ⟵ row i in mServices ;
15 |   |   row_j ⟵ row j in mServices ;
16 |   |   if (RR(row_i[1], row_j[1], C, w)) then
17 |   |   |   tmp ⟵ row_j;
18 |   |   |   row_j ⟵ row_i;
19 |   |   |   row_i ⟵ tmp;
20 return mServices;
```

4 Performance Analysis

We implemented a prototype called PMRF (Parameterized Matching-Ranking Framework) that supports both the matching and ranking algorithms. We used OWLS API to parse the published Web services list and the user request. The similarity between the concepts inferred using Jena API. In the current version, PMRF supports only Input and Output attributes. The open source tool SME2 (Semantic Matchmaker Evaluation Environment) has been used for performance evaluation and comparison using OWLS-TC collections. The implemented Plugin for the experiment requires a precise interface. Hence, we assigned to the Criteria Table the default values (Input: Fail, Output:Fail).

We compared PMRF with SPARQLent [7] and iSEM [5] frameworks. The results of analysis are shown in Fig. 1. The test collection used is OWLS-TC4, which consists of 1083 service offers described in OWL-S 1.1 and 42 queries. Fig. 1.*a* shows that PMRF recall is significantly better than iSEM logic-based and SPARQLent. This means that PMRF is able to reduce the amount of false positives. Fig. 1.*b* indicates that PMRF has a more accurate precision than iSEM logic-based and SPARQLent. This is due to the use of the score-based ranking, which gives a more coarse evaluation than degree-based aggregation. Fig. 1.*c* attests that PMRF is faster than SPARQLent and slightly less faster

than iSEM. SPARQLent has a high query response time if the query include preconditions/effects. PMRF and iSEM have close query response time because both consider only direct parent/child relations. Finally, Fig. 1.*d* reveals that PMRF consumes less memory than iSEM logic-based and SPARQLent.

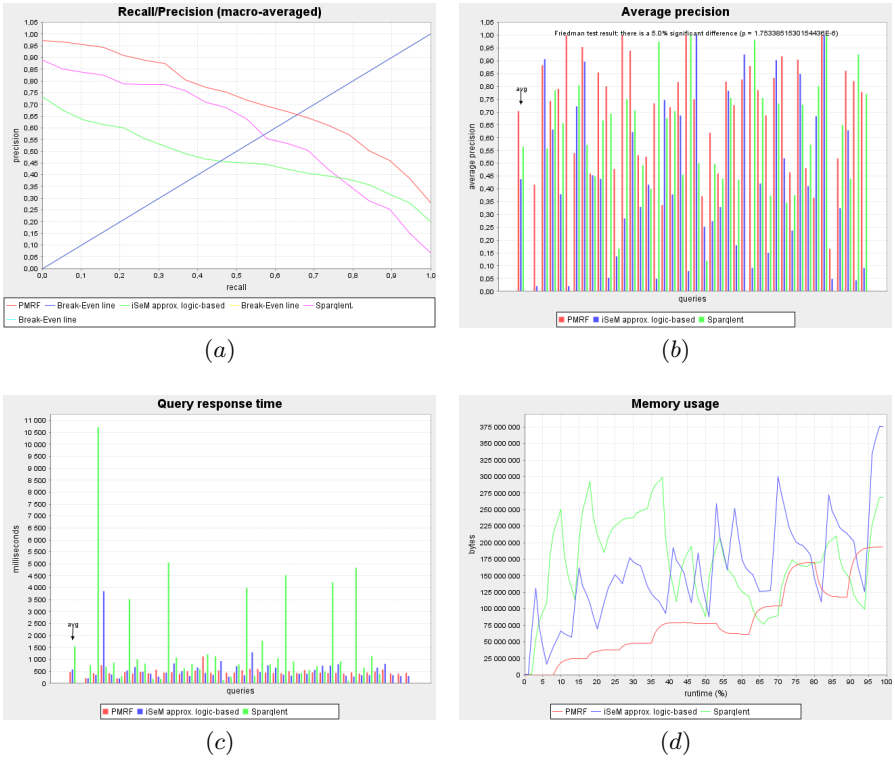

Fig. 1. Results of performance analysis

5 Related Work

A parameterized semantic matchmaking framework that enables the user to specify the matched attributes and the order in which attributes are compared is proposed in [4]. In [1], the authors propose a greedy-based algorithm that relies on the concept of matching bipartite graphs. The frameworks iSEM [5] and SPARQLent [7] consider preconditions and postconditions of Web service. In [2][3], we extended the work of [4] by relaxing matchmaking conditions.

Different ranking algorithms have been proposed. In [6], the authors propose a ranking method that combines the quality of service and fuzzy logic. The authors in [8] provide a context based method where the final rank list is obtained based

on the proximity of the context of similar Web services. In [9], the authors use the multicriteria analysis to sort Web services based on the dominance scores.

In this paper, we improved our previous matching algorithms given in [2][3] and added a new algorithm for ranking Web services. Although our approach relies entirely on logical techniques, it integrates a scoring technique, which provides a ranked list of discovered services.

6 Conclusion

We presented two parameterized algorithms for Web service matching and ranking. The matching algorithm takes as input a user query and performs a logic-based semantic matchmaking in order to select the services that match the user's constraints. Then, the ranking algorithm sorts the matching services based on their scores and the user's preference information. Both algorithms have been implemented and the performance analysis shows that the algorithms behave globally well in comparison to SPARQLent [7] and iSEM [5]. In the future, we intend to: (i) finalize the development of PMRF to support all attributes and different matching types, (ii) use other sorting algorithms, and (iii) extend our work to quality of service-based matching.

References

1. Bellur, U., Kulkarni, R.: Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In: IEEE International Conference on Web Services, pp. 86–93. IEEE Computer Society, Los Alamitos, California, USA (2007)
2. Chakhar, S.: Parameterized Attribute and Service Levels Semantic Matchmaking Framework for Service Composition. In: Fifth International Conference on Advances in Databases, Knowledge, and Data Applications, pp. 159–165. IARIA - International Academy, Research, and Industry Association, Wilmington, USA (2013)
3. Chakhar, S., Ishizaka, A., Labib, A.: QoS-Aware Parameterized Semantic Matchmaking for Web Service Composition. In: The 10th International Conference on Web Information Systems and Technologies, pp. 50–61. SciTePress - Science and Technology Publications, Barcelona, Spain (2014)
4. Doshi, P., Goodwin, R., Akkiraju, R., Roeder, S.: Parameterized Semantic Matchmaking for Workflow Composition. IBM Research Report RC23133, IBM Research Division (2004)
5. Klusch, M., Kapahnke, P.: The iSEM matchmaker: A Flexible Approach for Adaptive Hybrid Semantic Service Selection. Journal of Web Semantics 15, 1–14 (2012)
6. Manoharan, R., Archana, A., Cowla, S.N.: Hybrid Web Services Ranking Algorithm. International Journal of Computer Science Issues 8, 83–97 (2011)
7. Sbodio, M.L., Martin, D., Moulin, C.: Discovering Semantic Web Services Using SPARQL and Intelligent Agents. Journal of Web Semantics 8, 310–328 (2010)
8. Segev, A., Toch, E.: Context Based Matching and Ranking of Web Services for Composition. IEEE Transactions on Services Computing 3, 210–222 (2011)
9. Skoutas, D., Sacharidis, D., Simitsis, A., Sellis, T.: Ranking and Clustering Web Services Using Multicriteria Dominance Relationships. IEEE Transactions on Services Computing 3, 163–177 (2010)