Flexible Matchmaking for RESTful Web Services

Fatma Slaimi, Sana Sellami, Omar Boucelma, and Ahlem Ben Hassine

National School of Computer Science (ENSI), University of Manouba, Tunisia
Aix-Marseille Univ, LSIS, UMR CNRS 7296, Marseille, France
{fatma.slaimi,ahlembh}@gmail.com,
{sana.sellami,omar.boucelma}@lsis.org

Abstract. This paper describes a flexible matchmaking approach that relies on an extensible set of schema/ontology matching techniques and primitives that may be combined in various ways, accordingly with the specification of a service request. The approach has been implemented and tested against a collection of RESTFul web services described in hRESTS microformat.

Keywords: Semantic Web Services Discovery, RESTful Services.

1 Introduction

With the proliferation of web services (WS), WS discovery has been identified as one of the key challenges for achieving efficient service oriented computing. As the number of (publicly) available web services keeps growing, discovering the right service(s) is still an issue although a significant amount of work has been done during the last decade.

With the advent of the Semantic Web, new semantic techniques and systems has been proposed leading a new research topic known as Semantic Web Service Discovery. A number of solutions has been proposed in the literature, and resulted in different semantic matchmaking techniques and systems [1]. Despite commonalities shared with the schema matching research area [1], (semantic) web services discovery techniques have been proposed along a separate and independent pathway. Nevertheless, during the past few years, some schema matching approaches have been proposed to address WS discovery [2].

In this paper, we advocate an "unbundled" approach (versus a monolithic one) for semantic web services discovery. The approach relies on an extensible set of schema/ontology matching techniques and primitives that may be combined in various ways, accordingly with the specification of a service request. Matching techniques have been primarily developed to (semi)automatically allow the discovery of correspondences between (database/XML) schemas, mostly in the context of Information Integration. Matching approaches depend on the description of services and can be applied at the element or structure levels; they may be terminological, structural, or semantic [15].

[1] http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 542–554, 2013.

The contribution of the paper is twofold: (1) a RESTFul matchmaking approach based on schema matching, and (2) an experimental evaluation performed in using the well known hRESTS-TC1 service retrieval test collection [2].

The remainder of this paper is organized as follows: In Section 2, we review the semantic web services concepts that are mandatory for the comprehension of the rest of the paper; in Section 3 we survey some related work, before we detail our approach in Section 4. Section 5 describes and discusses our evaluation environment. Finally we conclude in Section 6.

2 Semantic Web Services

Several languages for semantic annotation of Web services are proposed in the literature. The most known are OWL-S (Web Ontology Language for Web Services), WSMO (Web Service Modeling Ontology), WSML (Web Service Modeling Language), and annotation languages like WSDL-S (Web Service Semantics), SAWSDL (Semantic Annotations for WSDL and XML Schema) and SA-REST . These languages differ in their complexity and expressive power. Languages such as OWL-S and WSMO aim to define an ontology for semantic web service description. However, the annotation languages propose to add annotations as an extension of WSDL description and connect the annotated elements with semantic information. The main goal of semantic annotation approaches is to enhance the syntactic description of web services with semantic information. The most important elements of web services that need to be annotated are functional properties related to the input, output, conditions and effects. These elements are processed during service discovery and composition. Instead of traditional SOAP-based services, which can use different standards to be described both syntactically that semantically, there are no standard for REST based services. Rather, in almost all cases, RESTful services are described with full text documentation which details the functionality of the service and the way to use it. hRESTS (HTML for RESTful Services) [3] is a micro format that enables the creation of machine-processable Web API descriptions based on available HTML documentation. hRESTS is complemented by the MicroWSMO microformat, which supports the semantic annotation of service properties in a SAWSDL-like manner. MicroWSMO introduces additional HTML classes, in order to enable the linking of ontological elements and the provisioning of machinery for transforming data exchanged between two services used in a service composition.

The main components of hRESTS services are:

- service : interface that the client deals with,
- operation : an action that the client can perform,
- resources : the address (URI) to invoke the operation
- method : detect the HTTP method
- input/outputs messages :request and response are the messages sent as input and output of the operation

[2] Available at `http://www.semwebcentral.org/`

– links : in the output messages, make a run-time hypertext graph of related resources.

hRESTS service functional model can be presented by the following Fig.1. This model is very similar to WSDL model [3], hRESTS aims to provide a machine-readable representation of common Web service description.

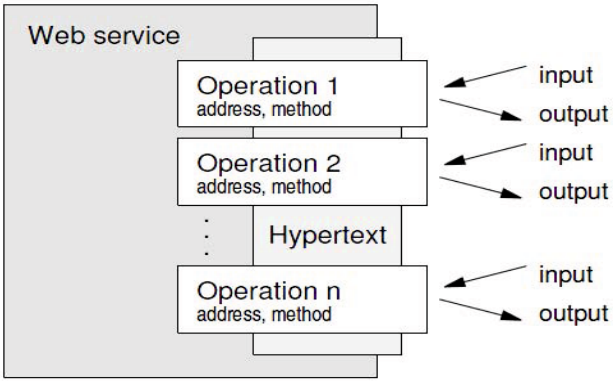

Fig. 1. hRESTS Service Model ([3])

hRESTS web service description represent only syntactic information, it doesn't include semantics. To semantically annotate this description, the MicroWSMO microformat can be used. It has been proposed in [3]. It define links to semantic concepts (ontology represented in the Web Ontology Language) to describe web service components.

3 Related Work

During the last decade, several Semantic Web Service Discovery approaches have been proposed in the literature. Most of them have been devoted to the matchmaking of SAWSDL[4] [5] [6] and OWL-S services [7] [8]. These matchmakers can be logic-based, non logic-based or hybrid, i.e., a combination of logic and non logic according to the classification proposed in [9].

Discovering RESTful services approaches has been recently proposed in the literature [10] [11] [12]. The first work XAM4SWS [10] compares operation, input, output and service similarities. These similarities are combined in the single one for each pair of operations. Semantic matchmaking considers hRESTS characteristics and determines verb (get, post, put, delete) similarity values. The second work [11] proposed a graph-theoretic approach for matching RESTful services. This approach matches RESTful web services functionalities based on their WADL(Web Application Description Language) elements. It uses linguistic knowledge and domain-specific heuristics.

[3] //www.w3.org/TR/wsdl20/

4 SR-REST: A Matchmaking Approach for RESTful Services

We propose SR-REST[4] a matchmaking approach for RESTful web services. Given a repository of web services and a user's request our aim is to automatically perform matching of this query with services as a means to find out a mapping indicating the services' elements corresponding to the underlying request. The result of the matching process is a set of mapping elements of both the schema of matched elements and the plausibility of their correspondence. The latter is defined using a similarity value between 0, i.e. strong dissimilarity; and 1, i.e strong similarity. The overall similarity between two services should be computed based on the degree of matching of their underlying operations.

The main idea of the approach consists in comparing the services with the requested service, i.e., operations, inputs and outputs, based on several SWS matchmaking strategies. Our aim is to take leverage of existing matchers [9] [13] such as, logic, syntactic, structural and semantic matchers, in providing suitable combination of these latters, leading to a flexible and efficient matchmaking algorithms. The approach is tested against RESTful web services encoded in hRESTS. The proposed matchmaker, SR-REST, exploits information from all levels of the hRESTS Web service description in the matching process, i.e., service, operation and input/output levels.

4.1 Logic-Based Matchers

Usually, the inputs and the outputs are highly important in the service description due to the fact that they are considered as the only annotated part of the service enabling the use of the semantic relations between services' parameters. Therefore, SR-REST performs a logic matching based on the subsumption reasoning. It computes the logic match between inputs (respectively outputs) of the service offer and the service request using different logic filters proposed by Paolucci and al. [14]. Given two concepts C (from the service request) and D (from the service offer), the degrees of logic match (DoM) are defined as bellow:

- **exact:** if $C \equiv D$
- **plug-in:** if $C \sqsubseteq D$
- **subsume:** if $C \sqsupseteq C$
- **fail:** if C is disjoint with D

These filters allow us to exploit subsumption relations e.g. generalization, specification between concepts of an ontology. However, there are other semantic relations between concepts that could be useful in the matchmaking process (fig. 2) such as neighborhood relations. For that reason, we propose a new filter to detect such kind of relation called *intersect*. Assuming that we have two sets LG and LN of direct generalization and direct neighbors respectively, where:

[4] Service (Entity) Resolution RESTful web services

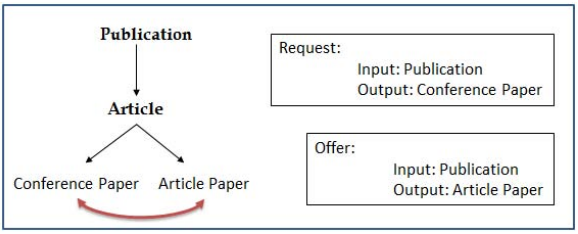

Fig. 2. Example of sibling relation between concepts

- **LG (C)**is the set of all direct generalization C' of the concept C, i.e. C is a sub concept of C'.
- **LN (C)**is the set of all direct neighbors C' of the concept C, i.e. LG (C) = LG (C')
- ∀(A, B) pair of concept, A *intersect* B, (denoted intersect(A, B)) if and only if ∃ a concept C where C ∈ LN (A) and C ∈ LN (B).

As a means to rank the previously mentioned filters, SR-REST attributes a quantitative equivalence (as in [5]) for each DoM relation based on the distance between concepts in the ontology. So, we mapped the pre-cited discrete DoMs into a constant numerical scale, varying from zero (no similarity) to one (exact similarity). Formally, we define PL(C,D) as the shortest path between two concepts C and D in an ontology and E the numerical equivalent for the DoM filter (exact, plugin, subsume, intersect, fail). Then, the logic similarity Ls(C,D), between the two concepts C and D is given by the following measure:

$$Ls(C, D) = E/PL(C, D). \tag{1}$$

4.2 Syntactic-Based Matchers

SR-REST can perform syntactic matching by computing the similarity between names, operation and descriptions of the service. For each pair of, service offer operation and request operation, it measures the similarity between texts based on functions proposed in simpack[15] such as: the Extended Jaccard , the Loss-of-Information , the Jensen-Shannon and the Cosine similarity measures.

Name Similarity. Names of services are defined by programmers. Usually, programmers respect some conventions, the names of variables and methods generally have meanings. A service name can summarize the functionality of the service. Using a tokenizer, some available similarity measure methods can be used to determine the similarity value between an offered service name and a requested service name, i.e., levensthein edit distance, average string, Dice's coefficient, Jaro coefficient, TF-IDF. SR-REST deploys some string based measurements described in Simpack [15] to compute the similarity value between service names.

The performance of names based matching depends on the qualities of service names. Note that, some service names describe perfectly the functionalities of their services while others cannot.

Text Similarity. Text description consists of a description of the web service written by developers in natural language using simple phrases and technical terms in an easily understandable way. Text description uses simple sentences instead of complicated ones leading to an easy process. Two similar services may use, in their text descriptions, similar terms, abbreviations, domain specific terminology, and phrases. Thus, it is possible to discover similar services by matchmaking their text descriptions. Hence, several web services matchmakers proposed in the literature [6] [16] consider text descriptions of Web services as a crucial information for the discovery process. These matchmakers use Tokenizer, they first convert words into lowercases, then use a porter stemmer for stemming and finally filter words using special characters, e.g. stop words, separators, etc. Text description of a hRESTS Web service document is represented by a vector where each dimension expresses a pair: the term and its frequency in the document. This matchmaker supports various vectors based on the similarity measurement functions implemented by Simpack.

4.3 Semantic Structural-Based Matcher

Semantic Structural based matchmakers are graph algorithms that consider the inputs as labeled graphs. They analyze the position of nodes in the graph. The importance of the structural similarities can be vindicated by the fact that same elements may appear in divers contexts and hence need to be differentiated to ensure a correct matching. In this way, SR-REST is also based on structural matching to determine the structural similarities between Input/Output elements. This similarity is based on the context elements and on the hypothesis that two elements are structurally similar if theirs structural neighbors are similar. The context includes: ancestor, sibling, immediate Child and leaf nodes.

The problem of semantic relatedness in an ontology or taxonomy is well known in the literature [17]. Generally, existing approaches represent the ontology as a graph for the semantic structural measure where nodes represent the different concepts and the edges represent relationships between concepts. However, the distance between concepts (graph nodes) can be used to compute the degree of their similarity. One of the most used approach for the semantic relatedness measure is Resnik [18], Resnik's measure function calculates the semantic similarity between ontology's concepts. It is based on the information content of a concept, which uses the instance probability. Using Resnik, different concepts in the ontology are annotated with values of the probability of their appearance in the service description.

$$sim_{Res}(A, B) = \max_{c \in S(A,B)} -logp(c) \tag{2}$$

where S(A,B) is the set of concepts that subsume both A and B in the ontology.

4.4 Structural-Based Matcher

A hRESTS Web service document can be given the structure of a labeled tree with the service at the root. Thereby, the problem of structural matching can be converted into a tree matching problem. Based on an XML schema, we create a subtree for each operation. The root of a subtree is the operation itself where the other nodes describe the inputs and outputs parameters. We use a WSDL Analyzer [19] to find the structural similarities between Input/output elements. This analyzer measures the similarities and the differences between two Web services. In addition, it exploits different types of data in the WSDL schema (names, type of information, structure). It proceeds first by identifying the similarities and differences over a set of operations, then by examining the structure of the Input/Output parameters. In the tree adopted depiction of a hRESTS description Web service document, the leaf nodes are the basic data types given by the XML schema.

– Definition (Service Tree): A labeled tree TS = (N,E,R,F) is an acyclic, connected graph where N = {n_1, n_2,..., n_p } a set of nodes and E a set of edge. The function F : N $\longrightarrow$ L assigns a label to each node with basic data types DT $\in$ L (the set of labels) and R is the root of the tree TS.

The measurement of the similarity between two trees of services TS1 and TS2 begins from the root and recursively pass through the tree. For every node c $\in$ N_{TS1} and d $\in$ N_{TS2}.

$$Sim(c,d) = \begin{cases} wn * simn(F(c), F(d)) \\ +ws * Max \oplus i, jF(c), F(d) \notin D \\ simt(F(c), F(d))F(c), F(d) \in D \end{cases}$$

where (c, n_i)$\in$ E_{TS1} and (d,m_j)$\in$ E_{TS2}, $\bigoplus i, j$ is the sum of pairs sim(n_i,m_j) for $1 \leq i < card(n)$ and $1 \leq j < card(m)$, every n_i and m_j occur a once in the sum. If card(n) = card(m), there are some nodes that cannot be matched. Weights w_n and w_s are employed to reflect the effect of the structural or name similarities. simt is a function to measure the Similarity between types, it is based on the use of a compatibility table which assigns a degree to each pair of basic data types. The labels similarity, simn can be measured with different functions: edit distance, substring containment or similarity in WordNet (measure the semantic proximity).

4.5 SR-REST Algorithm

SR-REST takes into consideration all web service components (service, operation, inputs and outputs) during the matchmaking process. The underlying approach applies multiple matching strategies for each component: syntactic matching for service level and text description (when available) and operations names, logic and structural matching for inputs and outputs parameters. For this reason, we propose four different functions, Sim_{Ser} (Service similarity), Sim_{Op}

(operations similarity), Sim_{in} (inputs similarity), and Sim_{out}(outputs similarity). These similarities are combined using weights in an aggregated function *simA* defined as follows:

$$\text{simA(a,b)} = \text{sim}_{Ser}\text{(a,b)} * \text{w}_{Ser} + \text{sim}_{Op}\text{(a,b)} * \text{w}_{Op} + \text{sim}_{in}\text{(a,b)} * \text{w}_{in} + \text{sim}_{out}\text{ (a,b)} * \text{w}_{out}$$

$$\text{where } w_{Ser} + w_{Op} + w_{in} + w_{out} = 1 \tag{3}$$

```
program Service Matcher (Output)
  {
   var
     reqServ, offServ: Services
   begin
     Similarity(reqServ.ser,offServ.ser);

  ForEach (OpR[i] In reqServ.operations)
    { ForEach(OpO[j] in offServ.operations)
    {   simOp(i,j) = Similarity(OpR [i], OpO [j]);
        simIn(i,j) = matchParameters(OpR [i].inputs , OpO [j].inputs);
        simOut(i,j) = matchParameters( OpR [i].outputs , OpO [j].outputs);
        simA(i,j) = wSer * simSer(i,j) + wOp * simOp(i,j)
                    + wIn * simIn(i,j) + wOut * simOut(i,j);
        MSim(k)= simA(i,j);
        }
     }
      simg =SUM( MSim )/| reqService.operations|;
      return simg
    }
  }
end.
```

Fig. 3. *SR-REST Algorithm*

5 Evaluation

In this section, we present a preliminary evaluation of our approach. The hRESTS-TC1[5] test collection has been used to assess SR-REST Matchmaker. This collection is derived from SAWSDL-TC1 collection [10] and consists of 25 queries (from different domains: communication, food, economy, medical, travel and education), 895 services and 24 ontology used to semantically annotate the inputs and outputs parameters of web services. A relevance set is provided for each query, which is used to evaluate the precision of the matchmaker. SR-REST is implemented in Java and Pellet 2.0. We used SME2 the Semantic Matchmaker Evaluation Environment [6] to evaluate the performance of SR-REST.

[5] http://semwebcentral.org/projects/hRESTS-tc/
[6] http://projects.semwebcentral.org/projects/sme2/

For our evaluation, we use the Precision and Recall defined as follows:

$$\text{Precision}= \frac{|A \cap B|}{B}, \text{ Recall}= \frac{|A \cap B|}{A}$$

where A is the set of all relevant services for a request and B the set of all retrieved services for a request.

SME2 evaluates the retrieval performance of matchmakers by measuring the average-precision (AP) and the macro-averaged precision[20]. Average precision AP is the average of precisions computed at each point of the relevant documents in the returned ranked list. However Macro-Average corresponds to the precision values for answer sets returned by the matchmaker for all queries in the test collection at standard recall levels. As the evaluation of a single query is not sufficient to make a significant observation, macro average precision is computed over many queries giving equal weight to each user query. Ceiling interpolation is used to estimate precision values at each standardized recall level, since each query likely has a different number of relevant services.

The Macro-Average precision is defined as follows:

$$\text{Prec}_i = \frac{1}{Q}. \sum q \in Q \ \max\{P_o | R_o \geq Rec_i \wedge (R_o, P_o) \in O_q\}$$

where O_q is the set of observed pairs of recall/precision values for query q when scanning the ranked services in the answer set for q stepwise for true positives in the relevance sets of the test collection.

5.1 Matcher Configuration

SR-REST combines different matchers: logic, syntactic and structural. We have performed different evaluation runs using several settings. Based on the work proposed in [10], we applied three configurations of level weights : interface, operations and parameters($w_{interface}$,$w_{operation}$,w_{inputs},$w_{outputs}$). The first configuration *config 1*(0, 0,0.5, 0.5) considers only inputs and outputs in the matching process. The second configuration *config 2* (0.25, 0.25, 0.25, 0.25) assigns the same weight to the different parameters. In the third configuration *config 3* (0.1, 0.1, 0.4, 0.4), we attribute a weight of 0.1 to the interface and operation and 0.4 to the inputs and outputs.

We assigned numerical equivalents for each DoM matching filter, exact=1, plugin=0.5, sub=0.5, intersect=0.25 and fail =0.

5.2 Experimentation

Table 1 summarizes the average precision (AP) for all previously described configurations of SR-REST. We assessed two versions of SR-REST. The first one without semantic structural match of parameters and the second one using Resnik for structural similarity measure.

Table 1. Averaged AP values

	config 1	config 2	Config 3
average.AP	0.70	0.73	0.75
	config 1+resnik	config 2 +resnik	Config 3+resnik
average.AP	0.71	0.754	0.77

We compare SR-REST with XAM4SWS [10] which is the only matchmaker to work on hRESTS services. SR-REST and XAM4SWS apply the same matchers: logic and syntatic ones. They attribute numerical equivalent for each DoM relation and aggregate the different measures in a global function. Unlike XAM4SWS, SR-REST deploys a semantic structural matching of input and outputs parameters using Resnik measure and considers the textual description of the service.

Table 2. SR-REST vs XAM4SWS Average Precision

Level weights	Num Dom Equivalents	AP SR-REST	AP XAM4SWS
config 1(0,0,0.5,0.5)	(1,0.5,0.5,0)	071	0.718
config 2 (0.25,0.25,0.25,0.25)	(1,0.5,0.5,0)	075	0.725
config 3(0.1,0.1,0.4,0.4)	(1,0.5,0.5,0)	0.77	0.747

5.3 Discussion

Based on the experimental results, we can make the following remarks:

- We used three different configurations to evaluate the performance of SR-REST approach. As illustrated in Fig. 4, in taking into account the service and operation levels in the matchmaking process, we did improve both precision and recall. This can be explained by the fact that, in some cases, a service name can reflect the functionality of the service and thus some pertinent services can be discovered while taking into account the name and the textual description of the service.
- Semantic structural matching (Resnik) improves the precision and recall of SR-REST (Fig. 5). This matcher considers the different relations between concepts of the ontology. So, indirect subsumption relations between concepts can be identified and consequently some relevant results can be retrieved.
- The structure of hRESTS documents may play an important role in the matchmaking process. In some cases, false positives and false negatives can be caused by the difference in the cardinality of input and output parameters sets. Then, we used WSDL-Analyzer to determine structural matching of hRESTS documents based on a labeled tree matching. In fact a hRESTS document can be considered as a WSDL file after elimination of its concretes parts (binding, ports, methods).

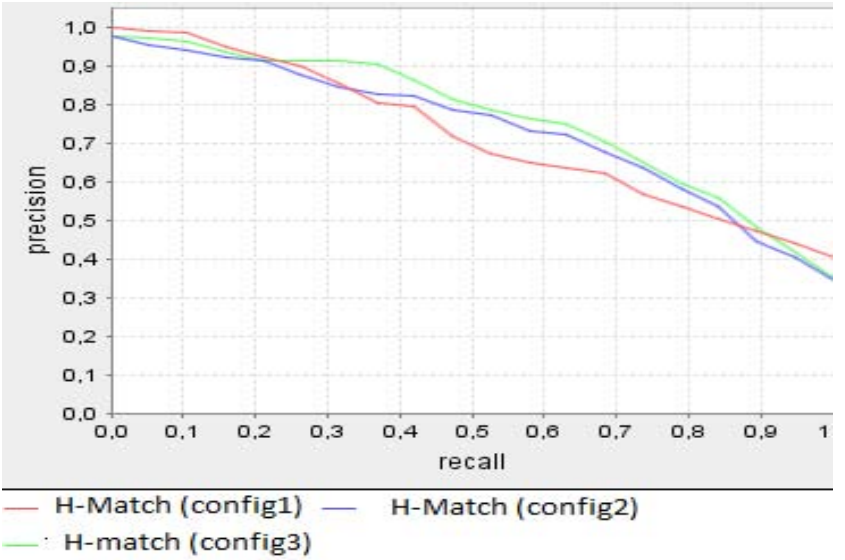

Fig. 4. Recall/Precision Macro Averaged

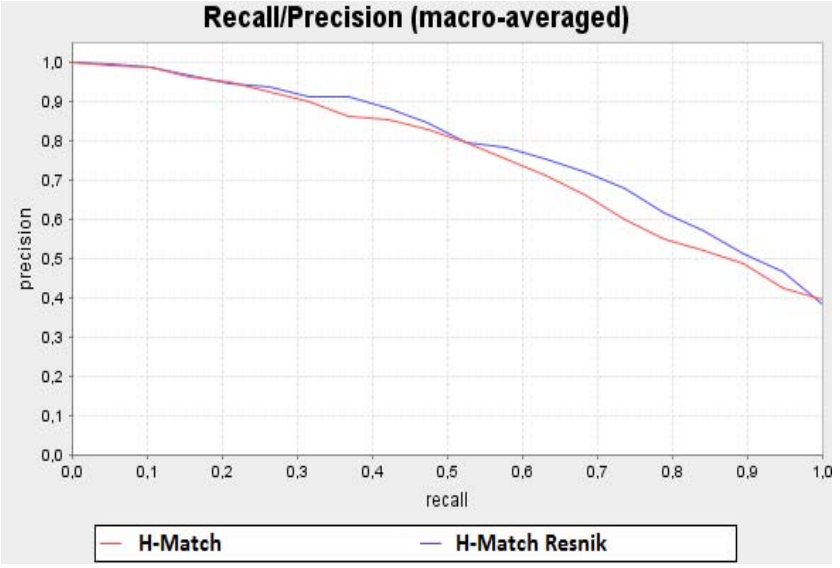

Fig. 5. Recall/Precision Macro Averaged(config3)

– Matcher tuning is a hard task and is highly dependent on the collection tests. While the combination of logic, syntactic, structural and semantic matchers yield to good results, an improvement of these results can be done in using

other matchers or techniques. Most of the existing matchmakers use machine learning or approximate matching techniques to this aim. Then, we believe that we can improve SR-REST in considering more features like synonym relations and in applying an approximative logic matching of annotated parameters to compute the approximated concept subsumption relations and the corresponding approximated signature matching. In addition a learning machine approach can be used to aggregate results of different matchers. As in XAM4SWS, we could use an estimator such as OLS (Ordinary Least Squares) that maps the logic filter to its numerical counterpart.

6 Conclusion

In this paper, we described SR-REST a flexible matchmaking approach that allows the combination of several (schema) matching/matchmaking techniques to achieve semantic web service discovery. The approach has been implemented and successfully tested against a collection of RESTFul web services described in hRESTS.

Although we used the hRESTS description of web services, SR-REST is generic enough to be applied for the discovery of other web resources (e.g., Cloud APIs) that maybe (more or less) semantically annotated.

Finally, applicability and importance of the approach could be also discussed with respect to the availability of a significant number of hRESTS web services. One should notice, that an important number of REST services are being made available (see *programmableweb* website[7] for instance). Hence, assuming one can (semi)automatically annotate those REST APIs, the SR-REST approach described in this paper could be tested and evaluated against REST services listed in the *programmableweb* directory.

References

1. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping. Springer (2011)
2. Algergawy, A., Nayak, R., Siegmund, N., Köppen, V., Saake, G.: Combining schema and level-based matching for web service discovery. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 114–128. Springer, Heidelberg (2010)
3. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML microformat for describing RESTful web services. In: Web Intelligence, pp. 619–625. IEEE (2008)
4. Klusch, M., Kapahnke, P., Zinnikus, I.: Adaptive hybrid semantic selection of sawsdl services with sawsdl-mx2. Int. J. Semantic Web Inf. Syst. (2010)
5. Schulte, S., Lampe, U., Eckert, J., Steinmetz, R.: Log4sws.kom: Self-adapting semantic web service discovery for sawsdl. In: The 6th World Congress on Services, SERVICES 2010, Miami, Florida, USA, July 5-10, pp. 511–518. IEEE Computer Society (2010)

[7] `www.programmableweb.com`

6. Wei, D., Wang, T., Wang, J., Bernstein, A.: Sawsdl-imatcher: A customizable and effective semantic web service matchmaker. Web Semantics: Science, Services and Agents on the World Wide Web (2011)
7. Klusch, M., Kapahnke, P.: Adaptive signature-based semantic selection of services with owls-mx3. Multiagent and Grid Systems 8(1), 69–82 (2012)
8. Klusch, M., Kapahnke, P.: iSeM: Approximated reasoning for adaptive hybrid selection of semantic services. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 30–44. Springer, Heidelberg (2010)
9. Klusch, M.: Semantic web service coordination. In: CASCOM: Intelligent Service Coordination in the Semantic Web, pp. 59–104 (2008)
10. Lampe, U., Schulte, S., Siebenhaar, M., Schuller, D., Steinmetz, R.: Adaptive matchmaking for restful services based on hrests and microwsmo. In: Proceedings of the 5th Workshop on Emerging Web Services Technology, WEWST 2010, Ayia Napa, Cyprus, December 1. ACM International Conference Proceeding Series, pp. 10–17. ACM (2010)
11. Khorasgani, R.R., Stroulia, E., Zaïane, O.R.: Web service matching for restful web services. In: 13th IEEE International Symposium on Web Systems Evolution, WSE 2011, Williamsburg, VA, USA, September 30, pp. 115–124. IEEE (2011)
12. John, D., Rajasree, M.S.: Restdoc: Describe, discover and compose restful semantic web services using annotated documentations. International Journal of Web and Semantic Technology (IJWseT) 4(1) (2013)
13. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
14. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
15. Bernstein, A., Kaufmann, E., Kiefer, C., Bürki, C.: SimPack: A generic Java library for similiarity measures in ontologies. Technical report, Department of Informatics, University of Zurich, Zurich, Switzerland (2005)
16. Becker, J., Müller, O., Woditsch, M.: An ontology-based natural language service discovery engine - design and experimental evaluation. In: Alexander, P.M., Turpin, M., van Deventer, J.P. (eds.) ECIS (2010)
17. Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 241–257. Springer, Heidelberg (2003)
18. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11(1), 95–130 (1999)
19. Klusch, M., Kapahnke, P., Zinnikus, I.: Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 550–564. Springer, Heidelberg (2009)
20. Raghavan, V.V., Bollmann, P., Jung, G.S.: Retrieval system evaluation using recall and precision: Problems and answers. In: Proceedings of the 12th International Conference on Research and Development in Information Retrieval, SIGIR 1989, Cambridge, Massachusetts, USA, June 25-28, pp. 59–68 (1989)