EFFICIENTLY LOCATING WEB SERVICES USING A SEQUENCE-BASED SCHEMA MATCHING APPROACH

Alsayed Algergawy, Eike Schallehn and Gunter Saake
Computer Science Department, Magdeburg University, 39106 Magdeburg, Germany

Keywords: Web service, WSDL, XML, Level matching, Schema matching, Prüfer sequence.

Abstract: Locating desired Web services has become a challenging research problem due to the vast number of available Web services within an organization and on the Web. This necessitates the need for developing flexible, effective, and efficient Web service discovery frameworks. To this purpose, both the semantic description and the structure information of Web services should be exploited in an efficient manner. This paper presents a flexible and efficient service discovery approach, which is based on the use of the Prüfer encoding method to construct a one-to-one correspondence between Web services and sequence representations. In this paper, we describe and experimentally evaluate our Web service discovery approach.

1 INTRODUCTION

Web services are well-defined, reusable software components that perform specific, encapsulated tasks via standardized Web-oriented mechanisms. They can be discovered, invoked, and composed. The research community has identified two major areas of interest: *Web service discovery* and *Web service composition* (Ma et al., 2008). This paper presents the issue of locating Web services efficiently. As the number of Web services increases, the problem of locating Web services of interest from a large pool becomes a challenging research problem (Wang and Stroulia, 2003; Hao and Zhang, 2007; Bose et al., 2008). In fact, to address this problem, several simple search engines have been developed. However, these engines provide only simple *keyword* search on Web service descriptions. Recently, traditional attribute-based matchmaking algorithms have been proposed. In the Web service discovery context it becomes apparent that keyword search and attribute-based mechanisms are insufficient due to the following reasons. They do not capture the underlying semantic of Web services and/or they partially satisfy the need of user search. This is due to the fact that keywords are often described by a natural language. As a result, the number of retrieved services with respect to the keywords are huge and/or the retrieved services might be irrelevant to the need of their consumers (Ma et al., 2008). More recently, this issue sparked a new research into the Semantic Web where some research uses ontology to annotate the elements in Web services (Atkinson et al., 2007; Nayak and Lee, 2007). Nevertheless, integrating different ontologies may be difficult, while the creation and maintenance of ontologies may involve a huge amount of human effort.

To address these challenges, we propose a new technique for effectively and efficiently locating Web services on the Web. We start by analyzing the Web service document specifications described in WSDL and representing them as service trees. Then, we identify and extract *concrete* and *abstract* parts from each service tree. We notice that the concrete parts from different WSDL documents have the same hierarchical structure, but may have different names. Therefore, we propose a level matching approach to only linguistically compare elements at the same level. However, the abstract parts from different WSDL documents have differences in structure and semantic. For this, we represent service tree abstract parts (operations) as sequences using the Prüfer encoding method (Prufer, 1918), and then apply our sequence-based schema matching approach to the sequence representation. To validate the proposed approach, we conducted a set of experiments using real-world data sets.

Algergawy A., Schallehn E. and Saake G. (2009).
EFFICIENTLY LOCATING WEB SERVICES USING A SEQUENCE-BASED SCHEMA MATCHING APPROACH.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 287-290
DOI: 10.5220/0001969702870290
Copyright © SciTePress

2 OVERVIEW OF THE PROPOSED APPROACH

Our proposed approach is based on the exploitation of the structure and semantic information from WSDL documents. The objective is to develop an efficient approach that measures the similarity between Web services. The measured similarity is used as a guide in locating the desired Web service. To realize this goal, we first analyze WSDL documents and represent them as service trees using Java APIs for WSDL (JWSDL) and a SAX parser for the contents of the XML schema (the *types* element). Then, each service tree is examined to extract its concrete part and its abstract parts. We develop a level matching method to measure the similarity between concrete parts from different service trees. To measure the similarity between abstract parts, we propose a sequence-based matching algorithm. Figure 1 illustrates the outline of the proposed approach.

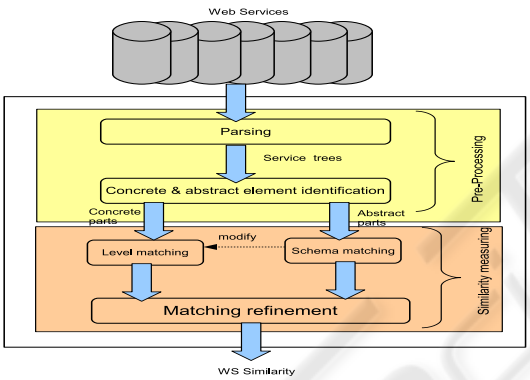


Figure 1: Web services similarity measure framework.

3 SIMILARITY MEASURING

3.1 Level Matching

Once obtaining the concrete part of each service tree, we apply the level matching approach on every concrete part pair. The proposed approach linguistically compares only nodes at the same level, as shown in Figure 2(a). The level matching approach considers only semantic information of concrete elements. It measures the elements' (tag names) similarity by comparing each pair of elements at the same level based on their names, assuming that the same names bear the same semantic meaning.

To compute the name similarity between two element names represented as strings, we first break each string into a set of tokens T_1 and T_2 through a customizable tokenizer using punctuation, upper case, special symbols, and digits, e.g, getDataService $\rightarrow$ {get, Data, Service}. We then determine the name similarity between the two sets of name tokens T_1 and T_2 as the average best similarity of each token with a token in the other set. The output of this stage for any two service trees $ST1$ & $ST2$ are 3 ($n' \times m'$) name similarity matrices, $NSimM$, where n' is the number of concrete elements of $ST1$ and m' is the number of concrete elements of $ST2$ per level

3.2 Schema Matching

To compute the similarity between abstract parts (operations), we should exploit both semantic and structural information of service trees. To achieve this goal, we propose a sequence-based matching approach. The proposed approach consists of two stages: *Prüfer Sequence Construction* and *Matching Algorithms*.[1]

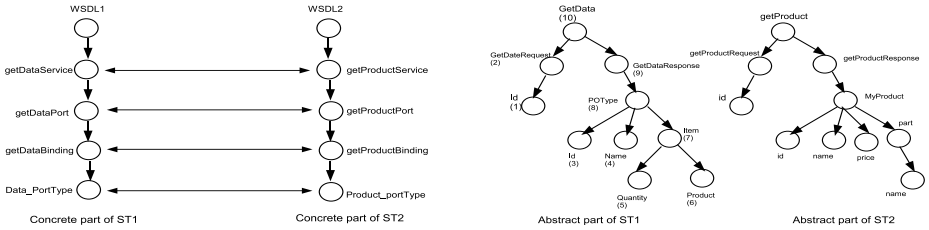
3.2.1 Prüfer Sequence Construction

In this stage, we aim to represent abstract parts of service trees as sequence representation using the Prüfer sequence method, which constructs a one-to-one correspondence between service trees and sequences. We capture the semantic information of service trees in Label Prüfer Sequences (LPSs), and the structural information of them in Number Prüfer Sequences (NPSs). The two sequences form so-called a Consolidated Prüfer Sequences ($CPS = (NPS, LPS)$) (Tatikonda et al., 2007). They are constructed by doing a *post-order traversal* that tags each node in the abstract part of the service tree, as shown in Figure 2(b), with a unique traversal number. NPS is then constructed iteratively by removing the node with the smallest traversal number and appending its parent node number to the already structured partial sequence. LPS is constructed similarly but by taking the node labels of deleted nodes instead of their parent node numbers. Therefore, NPS that is constructed from unique post-order traversal numbers gives tree structure information and LPS gives tree semantic information.

3.2.2 Matching Algorithms

In this stage, we aim to compute the similarity between abstract parts of service trees (operations). This task can be stated as follows: Consider we have two Web service document specifications $WSDL1$

[1]For more details about our sequence-based schema matching approach, see (Algergawy et al., 2008)



(*a*) Concrete parts of $ST1$ & $ST2$. (*b*) Abstract parts of $ST1$ & $ST2$.

Figure 2: Concrete & abstract parts of $ST1$ & $ST2$.

and $WSDL2$, each containing a set of operations. $OpSet1 = \{op_{11}, op_{12}, ..., op_{1k}\}$ represents the operation set belonging to $WSDL1$, while $OpSet2 = \{op_{21}, op_{22}, ..., op_{2k'}\}$ is the operation set of $WSDL2$. The task at hand is to construct $k \times k'$ operation similarity matrix, $OpSimM$. Each entry in the matrix represents the similarity between operation op_{1i} from the first set and operation op_{2j} from the second one. The proposed matching algorithm operates on the sequence representations of service tree operations and consists of three steps.

A linguistic matcher is used to compute a degree of linguistic similarity for element pairs. The linguistic matcher utilizes the same steps used in level matching. It also makes use of other element properties, such as data type of elements. The output of this phase are $k \times k'$ linguistic similarity matrices, $LSimM$, where k is the number of operations in $ST1$ and k' is the number of operations in $ST2$. Eq. 1 gives the entries of a matrix, where $DataType$ is a similarity function to compute the type/data type similarity between nodes, and $combine_l$ is an aggregation function, such as average, weighted sum to combine the name and data type similarities.

$$LSimM[i,j] = combine_l(Nsim(T_i, T_j), DataType(n_i, n_j)) \quad (1)$$

We then use a *structural matcher* to compute the structural similarity between abstract part elements. This matcher is based on the node context, which is reflected by its ancestors and its descendants. The descendants of an element include both its immediate children to reflect its basic structure and the leaves of the subtrees rooted at the element to reflect the element's content. In this paper, we consider three kinds of node contexts: *child*, *leaf*, and *ancestor* context. To measure the structural similarity between two nodes, we compute the similarity of their child, ancestor, and leaf contexts, utilizing the structural properties carried by sequence representations of service trees. The output of this phase are $k \times k'$ structural similarity matrices, $SSimM$. Eq. 2 gives entries of a matrix, where $child$, $leaf$, and $ancestor$ are similarity functions to compute the child, leaf, and ancestor context similarity between nodes respectively, and $combine_s$ is an aggregation function to combine these similarities.

$$SSimM[i,j] = combine_s(child(n_i, n_j), leaf(n_i, n_j), ancestor(n_i, n_j)) \quad (2)$$

After computing both linguistic and structural similarities between elements of every operation pair, we combine them. The output of this phase are $k \times k'$ total similarity matrices, $TSimM$. Equation 3 gives the entries of a matrix, where *combine* is an aggregation function to combine these similarities.

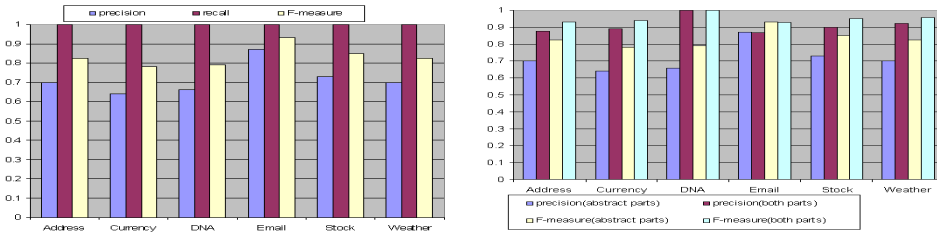
$$TSimM[i,j] = combine(LSimM[i,j], SSimM[i,j]) \quad (3)$$

Web Service Operation Similarity Matrix. We use $k \times k'$ total similarity matrices to construct the operation similarity matrix, $OpSimM$. We compute the total similarity between every operation pair by ranking element similarities in their total similarity matrix per element, selecting the best one, and averaging these selected similarities. Each computed value represents an entry in the matrix, where $OpSimM[i,j]$ is the similarity between operation op_{1i} from the first set and operation op_{2j} from the second set.

4 EXPERIMENTAL EVALUATION

Now we describe a set of experiments that validate the performance of our pro- posed algorithms. We used a collection of Web services published by XMethods[2] and QWS data set (Al-Masri and Mahmoud, 2007). We selected 43 WSDL documents from six different categories. All the experiments share the same design: each service of the collection was used as the basis for the desired service; this desired service was then matched against the complete set to identify the best target service. To evaluate the effectiveness of our proposed approach, we use precision, recall, and F-measure.

[2]http://www.xmethods.net



(*a*) Quality measures (abstract parts only). (*b*) Quality measures comparison.

Figure 3: Quality measures.

4.1 Experimental Results

The first possibility to match between Web services is to measure the similarity between their operations. In the first set of experiments, we match abstract parts of each service tree from each category against the abstract parts of all other service trees from all categories. Precision, recall, and F-measure are calculated and illustrated in Figure 3(a). As can be seen, our proposed framework has the ability to identify the desired Web service with recall (R) of 100% across all tested categories and precision (P) ranging from 64% to 87%. This reveals that our framework is almost accurate with F-measure ranging from 78% to 93%.

The second possibility to match between Web services is to exploit the similarity between concrete parts as well as the similarity between their operations. In this set of experiments, we matched the whole parts (both abstract and concrete) of each service tree against all other service trees from all categories. We computed precision, recall, and F-measure for this case, and we compared them against the results of the first possibility. Figure 3(b) reports the results. The figure shows that exploiting the whole WSDL document specifications improves the discovery quality.

5 CONCLUSIONS

In this paper, we described a new approach for Web service discovery based on schema matching techniques. The proposed approach makes use of the whole WSDL document specification and divides its elements into a concrete part and abstract parts. We devised a level matching approach for concrete parts, while we developed a sequence-based schema matching approach to compute the similarity between abstract parts. We have conducted a set of experiments to evaluate our approach. The initial results are encouraging. Further work will investigate the extension of the approach to integrate more semantic information and to exploit the full WSDL syntax in order to improve the approach performance.

REFERENCES

Al-Masri, E. and Mahmoud, Q. (2007). Qos-based discovery and ranking of web services. In *ICCCN 2007*, pages 529 – 534.

Algergawy, A., Schallehn, E., and Saake, G. (2008). A Prufer sequence-based approach for schema matching. In *BalticDB&IS2008*. Estonia.

Atkinson, C., Bostan, P., Hummel, O., and Stoll, D. (2007). A practical approach to web service discovery and retrieval. In *ICWS 2007*, pages 241–248.

Bose, A., Nayak, R., and Bruza, P. (2008). Improving web service discovery by using semantic models. In *WISE 2008*, pages 366–380. New Zealand.

Hao, Y. and Zhang, Y. (2007). Web services discovery based on schema matching. In *ACSC2007*, pages 107–113. Australia.

Ma, J., Zhang, Y., and He, J. (2008). Efficiently finding web services using a clustering semantic approach. In *CSSSIA 2008*, page 5. China.

Nayak, R. and Lee, B. (2007). Web service discovery with additional semantics and clustering. In *IEEE/WIC/ACM International Conference on Web Intelligence, WI2007*, pages 555 – 558.

Prufer, H. (1918). Neuer beweis eines satzes uber permutationen. *Archiv fur Mathematik und Physik*, 27:142–144.

Tatikonda, S., Parthasarathy, S., and Goyder, M. (2007). LCS-TRIM: Dynamic programming meets XML indexing and querying. In *VLDB'07*, pages 63–74.

Wang, Y. and Stroulia, E. (2003). Flexible interface matching for web-service discovery. In *WISE 2003*, pages 147–156. Italy.