Web Service Matching for RESTful Web Services Based on Parameter Semantic Network

Zhongbo Wu, Zhumei Dou, and Chaonan Song
Sci. and Tech. on Integrated Information System Lab
Institute of Software Chinese Academy of Sciences
Beijing, China
wuzhonghan59@163.com

Zhongbo Wu
University of Chinese Academy of Sciences
Beijing, China

***Abstract*—With the increasing demands of matching RESTful web services in data sharing systems, a novel service matching method based on the parameter semantic network model is proposed. Compared with traditional matching methods of web services, the proposed method aims at classifying various web services into some certain categories based on the themes, and then giving a convincing matching rate. Furthermore, the parameter semantic network model comprehensively supports service matching including traditional web services and RESTful web services, even for those RESTful web services without WADL. Simulation results demonstrate that the proposed matching method has a high precision rate and a high recall rate for the RESTful web services.**

Keywords—RESTful, Web service, Service matching, WADL, Service parameter

I. Introduction

There are many web services and their application programming interfaces (APIs) on the Internet. These APIs provide different services, usually with multiple meaning. Many web services provide similar functions, and they provide opportunities to developers. When the original API becomes unusable or cannot meet developers' demands, we should migrate the former service to the new one. Many scholars have studied the web service to make similar services discovered. They propose the methods that based on keywords [1], identifier and cluster [2].

Various scenes need web service matching technique, such as service transition: when the former product directory service cannot be used, its existing data need to be transplanted to service which has the familiar function. In principle service matching technique depends on port matching. Port matching always rely on a common port descriptive language, such as WSDL (web service descriptive language) [3]. With rapid development of RESTful web services, the usage of WSDL has become less and less. WADL (web application descriptive language) is REST style web services and alternative WSDL which describes SOAP (simple object access protocol). However, WADL is not the necessary descriptive language of the RESTful web services, so we need to propose a method independent of any descriptive language. From service parameters' point of view, we put forward a method that based on a parameter semantic network.

In this paper, a web service matching method is proposed for RESTful web services based on the parameter semantic network. The rest of this paper is organized as follows. Section II briefly introduced the existing methods for web service matching. In Section III, we propose the service matching method according to the parameter semantic network. The experiment results and discussion are included in Section IV. Section V concludes the work.

II. Related Work

Web service matching is based on similarity, and it is used for searching new web services, so the research of similarity is essential. Generally, web service matching is similar to the matching problems of other areas, such as database matching, text matching and software pattern matching of programming languages. But these matching problems are still different from web service matching, so the research achievements on these areas are not suitable for our study [4][5].

In database pattern matching, pattern matching is based on the pattern's predicted semantic, and the pattern's information integration is useful. Pattern matching methods which use domain knowledge and experience to improve success rate mostly contain language and structure analysis [6]. There are two reasons why these pattern-matching methods can't be used in web service matching. Firstly, pattern matching and web service matching have different granularity, we try to match features of two service models in web service matching, but pattern structures will be matched in pattern matching. Secondly, pattern matching usually needs higher coincidence degree, but web services are often complementary associated with each other and do not have such a high degree of coincidence.

Document matching has been studied for long time in information retrieval. Such methods depend on the word frequency. Because the description of web service don't contain enough text, such methods are not suitable for web service matching [7].

Another closely related problem is the matching of software components; it is an important subject in software

© 2014. The authors - Published by Atlantis Press

reusing [8]. The problem involves different programs, several features need to be considered, such as data type, parameter names, parameter types, parameter order and so on. Despite there are some apparent similarities between signature matching and web service matching, due to different levels of expression and data structures, the matching method available in one area maybe not available in other areas. Stroulia et al find web service via computing structure and semantic similarity of web service, they mainly used the feature-matching method.

Web service discovery is closely related to network service matching. Web service discovery is used to search the web services' repositories (universal description, discovery, and integration, UDDI) on the network. Common web search services are based on keywords, which make us to only find the services contain specified keywords when looking for service. Some recent researches on this area use semantics to get better results.

Inspired by automated discovery and usage of web service, a number of new researches in web service matching have emerged. Wang et al [3] proposed a flexible interface matching method for package, this method is based on data types, operation structures, natural language description and identifier semantic.

Mothhari-Nezhad [9] proposed a semi-automatic interface matching method, which combines business operations by defining the service agreement for service operation constrains. Mikhaiel et al [10] and Mothhari et al argued which service matching method is more accurate and efficient between interface matching and protocol matching such as WSDL and BPEL. The method proposed by Mothhari-Nezhad et al is based on the XML schema interface, also employed the protocol level information.

So far, most of the web services matching methods are based on WSDL description. However, RESTful web services are becoming a mainstream with the development of the Internet. RESTful Web service description language is based on WADL, and RESTful web services do not have WADL files in most cases. This will make the RESTful web services matching problem become more researchable.

Representational State Transfer is a way of sending an special structure HTTP request to the server from the client, including how to regulate access to information resources, processing request servers and a software architecture style of returning response. When the client is in the “resting” state, which means the client can interact with the user or in the transformation. The client is sending requests and waiting for responses when this happens. These requests and responses are representing a transfer of resources, because they have the resources of the current and expected state.

In the REST style, each resource has a global identifier such as HTTP URI and the client need to know the identifier and the desired operation. It also need to understand the format which is usually a HTML, XML or JSON (JavaScript object notation) metadata. RESTful web services, also known as the RESTful API, specify the resource which consists of three components: web services, supportive data URI: JSON, XML, YAML, and support the operation using the HTTP method, for example, POST GET , PUT or DELETE. A RESTful API can be described by using WSDL and SOAP over the HTTP, it can be an abstract concept purely based on the top of SOAP.

III. PROPOSED METHOD

Fig.1 illustrates the overall process of how to build the parameter semantic network, which consists of three main modules: data preprocess, parameter semantic network module, and training of similarity matrix. The data preprocess begins with the collecting of parameters from API documents and end with importing the parameters into the MySQL database. The second part explains how to build the mathematical model of the parameter semantic network. The last part will train the similarity matrix and get the thresholds and the result of matching.

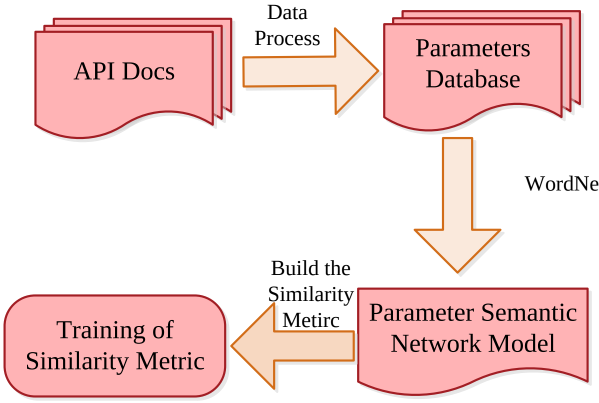
API Docs
Data Process
Parameters Database
WordNet
Build the Similarity Metirc
Parameter Semantic Network Model
Training of Similarity Metric

Figure 1. A flow diagram of building the parameter semantic network

A. Data Preprocessing

At the beginning, we collect several themes (e.g. weather, shopping, entertainment and so on) of API documentations of RESTful web service and mark the theme for each service. There are four different themes for the web services, which are map, shopping, entertainment and photo. Since different vendors may use different pattern of naming interface, some words whose original meanings are same may not be synonymous in the semantic database, resulting in data distortion. In some service, some parameters may use the abbreviated form, for which we need to build a mapping table used to expanse the abbreviated parameters. In this regard, manual corrections procedures combined with automatic correction procedures are needed. We often need to remove some parameters that have no semantic meaning (e.g. “page”,” key”) to obtain a clean data sources. According to the characteristics of service, we can divide parameters into three categories: theme parameters, request parameters and response parameters. Among these three kinds of parameters, the request parameters are necessary. The theme parameters are not prerequisite but are very important in the matching of web services. Most of the response parameter can be fetched except for some services are paid. Each parameter is stored in the form of string, separated by a space. For example, there is a string " location latitude longitude name" stands for the request parameters for the Amenimaps map service. According to the above rule, the Amenimaps web service has four request parameters: location, latitude, longitude and name. Then import the data into database. In database we create a table named restURL which have six columns. The names of six columns are company, theme, category, URL, requeststring and respondstring. The specific flow chart is show in Fig.2.

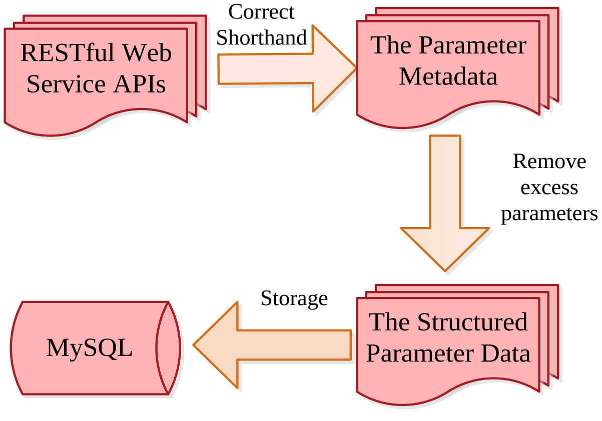

Figure 2. Data preprocessing

B. *Parameter Semantic Network Model*

In order to make the parameters connected to each other among the services, we must chose a third corpus as intermediate connector. WordNet appeared in 1985, mainly to represent a collection of words using synonyms concepts. In WordNet, word is composed of hierarchical inheritance strategy based on semantic tree. The problem of the semantic similarity of the parameters is translated into calculating the distance of parameters in the tree.

In the following, we conduct mathematical modeling for the above situation so that to have a rigorous logical basis. Each service is divided as a set. Every set has a three sub-set, which are the theme sub-set, the request sub-set and the request sub-set. Each parameter is one node of the sub-set. The parameters link to each other through semantic link library to create a semantic network of RESTful web parameter. We build the mathematical modeling as following.

Suppose that the feature vector of the web services is denoted as in (1)

$$W_i \triangleq \{T_i, R_i, A_i\}, i = 1,2,...,n \quad (1)$$

Where n is the number of the web services, W_i stands for ith web service, which has three sub-sets as following,

$$T_i = (t_{i1}, t_{i2}, ..., t_{ik}) \quad (2)$$

$$R_i = (r_{i1}, r_{i2}, ..., r_{im}) \quad (3)$$

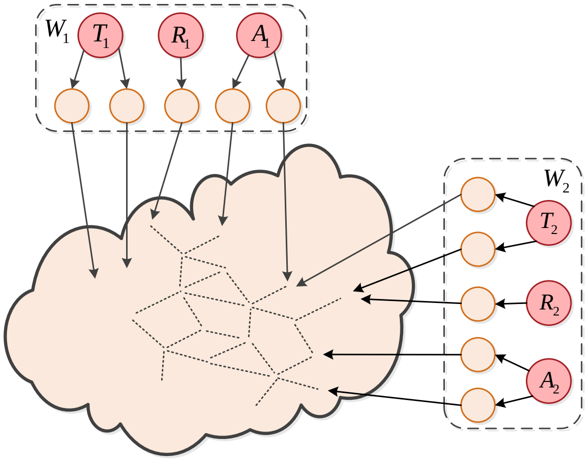

Figure 3. The Parameter Semantic Network Model

$$A_i = (a_{i1}, a_{i2}, ...a_{il}) \quad (4)$$

Where T_i is the ith set of the theme parameter, which contains k theme parameters. R_i is the ith set of the request parameter, which contains m request parameters. A_i is the ith set of response parameter, which has l respond parameters.

The parameter semantic network model is shown in the Fig.3. In the Fig.3, W_1 stands for a web service that contains the theme parameters T_1, the request parameters R_1 and the response parameters A_1. W_2 stands for a web service that contains the theme parameters T_2, the request parameters R_2 and the response parameters A_2. The clouds shaped graphic stands for the WordNet, which act as a connector between W_1 and W_2.

For example, there are two records of web services in our database, as shown in Table I. Then T_1 = {street, address}, R_1 = {street, city, state, zipcode, address, urban}, A_1 = {address, delivery, line, zipcode, street, longitude, latitude}, $W_1 = \{T_1, R_1, A_1\}$. T_2 = {postcode, distance}, R_2 = {postcode, province, country}, A_2 = {distance, postcode, longitude, latitude}, $W_2 = \{T_2, R_2, A_2\}$. Both services are marked as in the category of map.

C. *The Traning of the Similarity Matrix*

Since the number of RESTful Web service published on the Internet is limited and reading an API documentation to get parameters and mark the themes requires a lot of effort, the samples of RESTful web services will be very limited. Thus, statistical learning methods, which are commonly used in the field of natural language processing, are not applicable in our case. The commonly used clustering methods are also not suitable because each service only has similarity between each other but there is no uniform basis point, so data mining method like K-Mean is not applicable. Most previous researches are focused on two RESTful service and they did not give a convincing and specific match rate. To solve this problem, we need training and cross-validation on the threshold of classification and the thresholds of theme parameters, request parameters and response parameters, so as to achieve a higher matching rate for the RESTful web services.

TABLE I. TWO RECORDS OF WEB SERVICES IN OUR DATABASE

Company	Theme	Category	URL	RequestString	ResponseString
smartstreet	street address	map	https://api.smartystreets.com/street-address	street city state zipcode address urban	address delivery line zipcode street longitude latitude
trogsoft	postcode distance	map	http://postcodes.api.trogsoft.net/Postcode/DistanceBetweenPostcodes	postcode province country	distance postcode longitude latitude

According to the mathematical model of the parameter semantic network, the mathematical computing model of the similarity matrix can be elaborated as following. The similarity value between *i*th service and *j*th service can be computed as in (5)

$$S_{ij} = \lambda_1 F\left(T_i, T_j\right) + \lambda_2 F\left(R_i, R_j\right) + \lambda_3 F\left(A_i, A_j\right) \qquad (5)$$

Where $F()$ is the calculation of the value of the similarity of the two word sets. The proportion of the theme parameter, the request parameters, the response parameter is denoted by $\lambda_1, \lambda_2, \lambda_3$ separately. Because each factor belongs to a kind of parameters, so $\lambda_1, \lambda_2, \lambda_3$ must satisfy the following equation,

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \qquad (6)$$

In (5), the expression $F()$ can be shown as the following equation in (7),

$$F\left(T_i, T_j\right) = \frac{1}{m+n}\begin{pmatrix} Max[f(t_{i1,}t_{j1}),\ldots, f(t_{i1,}t_{jn})] + \ldots \\ +Max[f(t_{im,}t_{j1}),\ldots, f(t_{im,}t_{jn})] \end{pmatrix} \qquad (7)$$

Where the *m* is the number of the word parameters of T_i and *n* is the number of the word parameters of T_j. $f\left(t_i, t_j\right)$ is the calculation of the value of the similarity between two words, which is provided by the WordNet developers, which is a convenience for the research we conduct. Firstly, we calculate the similarities of one word in one set between all the words in other set, then we select the biggest one. For every word in the first set, we do the same thing. At last, we divide the total number of the parameter words to reconcile the similarity between two word sets.

After get all the similarities between all the services, the similarity matrix can be organized as in (8)

$$\mathbf{M}_S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ & S_{22} & \cdots & S_{2n} \\ & & \ddots & \vdots \\ & & & S_{nn} \end{bmatrix} \qquad (8)$$

Assume that the number of category for the web service is four and the classification threshold is δ, if $S_{ij} \le \delta$, we consider the *i*th service and the *j*th service are similar. If $S_{ij} \le \delta$, we consider the two service in different categories. When the similarities of a web service between other web services are all bigger than δ, we choose the service as its congener which have the biggest similarity between it. Then we can get the matching result of the web services. Thus, we can adjust the value of $\lambda_1, \lambda_2, \lambda_3$ and δ, and get new result. We repeat this kind of operation until getting satisfactory results.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

We measured the performance of our methodology in terms of precision and recall. In order to compute the precision and recall, we first count the services that been matched correctly (True-Positive: TP). Then we count the services that matched wrongly (False-Positive: FP). Finally, we count the services that should be put in the category but missed (False-Negative: FN). Therefore, we compute the precision rate and recall rate as in (9) and (10), respectively

$$Precision = \frac{TP}{TP+FP} \qquad (9)$$

$$Recall = \frac{TP}{TP+FN} \qquad (10)$$

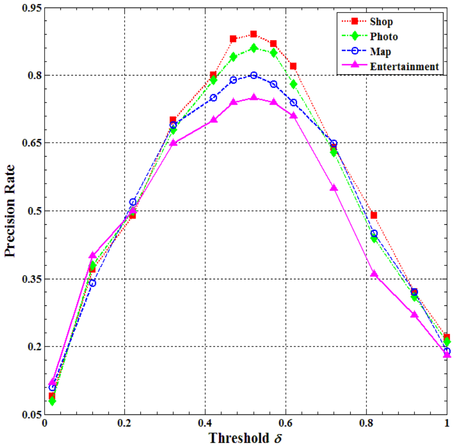

Figure 4. Precision rate

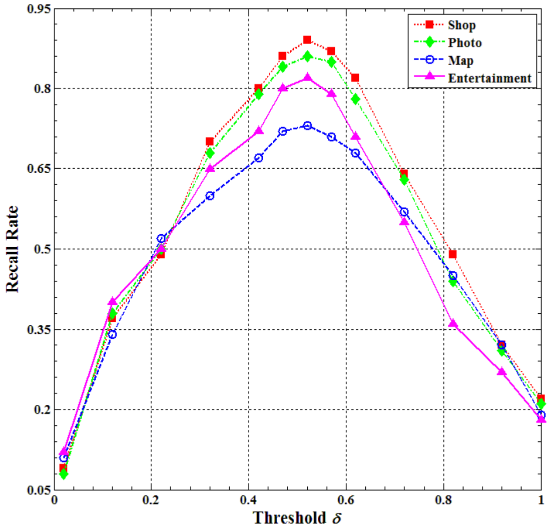

Figure 5. Recall rate

In our experiments, set λ_1 =0.5, λ_2 =0.3, λ_3 =0.2, we train the similarity matrix with the δ between 0 and 1. With the change of δ , the trend of the precision rate and the recall rate is separately shown as in Fig.4 and Fig.5.

In the Fig.4, when the value of δ is 0.02, the precision rate in four theme (Shop, Photo, Map, Entertainment) is about 0.08, which is too low to make the successful matching. Because when the value of δ is very low, one service can be matched to more than one service, which will reduce the precision rate apparently. Then when the value of δ grows, the precision rate grows too and reach the maximum when the value δ is about 0.54. When the value of δ is 1, the precision rate in four theme (Shop, Photo, Map, Entertainment) is about 0.2. Because when the value of δ is very high, one service can be matched to no service, which will reduce the precision rate apparently.

In the Fig.5, when the value of δ is 0.02, the recall rate in four theme (Shop, Photo, Map, Entertainment) is about 0.1, which is too low to make the successful matching. Because when the value of δ is very low, one service can be matched to more than one service, which will reduce the recall rate apparently. Then when the value of δ grows, the precision rate grows too and reach the maximum when the value δ is about 0.54. When the value of δ is 1, the recall rate in four theme (Shop, Photo, Map, Entertainment) is about 0.24. Because when the value of δ is very high, one service can be matched to no service, which will reduce the recall rate apparently.

After we trained the similarity matrix for hundred times, we found when λ_1 =0.5, λ_2 =0.3, λ_3 =0.2, δ =0.54, the precision rate and the recall rate can have the best results. The precision rate and recall rate of each domain are showed as in Table II.

TABLE II. PERFORMANCE COMPARISON

Domain	TP	FP	FN	Precision Rate	Recall Rate
Entertainment	8	2	3	0.80	0.73
Map	9	3	2	0.75	0.82
Photos	6	1	1	0.86	0.86
Shopping	8	1	1	0.89	0.89

We can explain the thresholds of the different parameters as following: theme parameter is the parameter that must contain in the URL, so it is of great decisive for the service in term of which category it belongs. The request parameter is more important than the respond parameter as it has a more direct link to the users.

V. CONCLUSION

In this paper, we propose a parameter semantic network model. Our approach proposes a solution to the more general problem, web service matching, which has many prospects in service discovery, composition and integration as well as many foreseeable applications for client. After studying a lot research work on the similar problem, such as schema matching, document matching, signature matching, we found their technologies and methods are not suitable for our case for many reasons as explained. The Parameter Semantic Network Model is a perfect solution for RESTful Web Service Matching problem, but it can be used for other semantic matching cases. We can also classify many services at a convincing matching rate, which have not been done at the same time in the previous research works.

The first place I need to improve is the acquisition of the parameters. We are now mainly using the manual methods to collect the service parameters, which is very cumbersome and lack of fault tolerance. This situation is mainly due to the lack of the uniform format of the RESTful Web API documents. With the rapid development of RESTful Web services, uniform standard will emerge, and then we will be able to use automated parameter acquisition.

The second point is that we need a more extensive evaluation for our method. Even we take a lot effort to collect the service data and collect more than 50 RESTful Web service and their parameters, the current data set is still small. In the future, as more enterprise publish their API documents in the network and have the uniform format, we will have hundred and thousand samples as the project goes on.

Last but not least, WordNet has only semantic similarity calculation method of the English words, powerless to deal with other language. This can be one of our future research directions. We can integrate our work with other techniques that have been implemented such as matching by IDs and mapping by structure (e.g. parameter data types). Future more, we can apply our methodology to the web service discovery and recommend application.

ACKNOWLEDGMENT

This work is supported by National High Technology Research and Development Program of China (863 Program, No. 2012AA011206).

REFERENCES

[1] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the Thirtieth Int. Conf. on Very Large Data Bases*, Toronto, Canada, 2004, pp. 372–383.

[2] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *Proceedings of the IEEE/WIC/ACM Int. Conf. on Web Intelligence*, Washington DC, USA, 2007, pp. 555–558.

[3] Y. Wang and E. Stroulia, "Flexible interface matching for web-service discovery," in *Proceedings of the Fourth Int. Conf. on Web Information Systems Engineering,* Washington DC, USA, 2003, pp. 147–150.

[4] R. R. Khorasgani, E. Stroulia, and O. R. Zaiane, "Web service matching for RESTful web services," In *IEEE Int. Symp. on Web Systems Evolution (WSE),* Williamsburg, USA, 2011, pp. 115-124.

[5] S. R. Ponnekanti and A. Fox, "Application-service interoperation without standardized service interfaces," In *Proceedings of the IEEE Int. Conf. on Pervasive Computing and Communications*, Texas, USA, 2003, pp. 30-37.

[6] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *J. on Data Semantics IV*. Dordrecht, Netherlands: Springer Netherlands, 2005, ch. 5. pp. 146–171.

[7] G. Salton and C. Buckley, "Global text matching for information retrieval," *Science*, vol. 253, no. 5023, pp. 1012–1015, Aug. 1991.

[8] C. Pahl, "An ontology for software component matching," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 2, pp. 169–178, Mar. 2007.

[9] H. R. Motahari Nezhad, G. Y. Xu, and B. Benatallah,"Protocol-aware matching of web service interfaces for adapter development," in *Proceedings of the 19th Int. Conf. on World Wide Web*, New York, USA, 2010, pp. 731–740.

[10] R. Mikhaiel and E. Stroulia, "Examining usage protocols for service discovery," *Service-Oriented Computing – ICSOC*. Dordrecht, Netherlands: Springer Netherlands, 2006, pp. 496–502.