Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 1313–1322

Procedia Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Fuzzy Matching of OpenAPI Described REST Services

Cong Peng[a], Prashant Goswami[a], Guohua Bai[a]

[a]*Department of Creative Technologies, Blekinge Institute of Technology, Karlskrona, Sweden*

Abstract

The vast amount of Web services brings the problem of discovering desired services for composition and orchestration. The syntactic service matching methods based on the classical set theory have a difficulty to capture the imprecise information. Therefore, an approximate service matching approach based on fuzzy control is explored in this paper. A service description matching model to the OpenAPI specification, which is the most widely used standard for describing the defacto REST Web services, is proposed to realize the fuzzy service matching with the fuzzy inference method developed by Mamdani and Assilian. An evaluation shows that the fuzzy service matching approach performed slightly better than the weighted approach in the setting context.

Keywords: service matching; fuzzy inference; OpenAPI; REST; fuzzy set theory

1. Introduction

The Web is growing exceedingly not only for the human-oriented Web pages, but also for the machine-oriented Application Programming Interfaces (APIs). The vast amount of Web services, on the one hand, provides multiple choices and imaginative utilization. On the other hand, it also became effort consuming when aiming for discovering high quality services for composition or orchestration. The quality of a Web service, in a sense, can be seen as how well it fulfils a service consumer's requirements [1]. In other words, it is to what extent a service matches the purpose and functionalities of a service consumer.

The purpose of service matching is evaluating the matching degree based on similarity, purpose, functionalities and so on [2]. Many approaches of service matching are based on extracting information from service descriptions [2, 3, 4]. Due to the advantage of lightweight implementation and naturally distributed architecture, the REST architectural styled Web services proposed by Roy Fielding [5] have become the defacto standard.

E-mail address: cong.peng@bth.se

1877-0509
10.1016/j.procs.2018.08.081

However, unlike SOAP-based traditional Web services, the REST Web service lacks a widely adopted formal description language. Although there are description formats include Web Application Description Language (WADL)[1] for describing REST services, and most of the previous studies of REST service matching are based on WADL, they are not widely adopted in real world production.

Although there have been works done on semantic service matching [6, 7, 8], it is difficult to integrate with the current REST Web services development technology stack due to the complicated ontology building and reluctant adoption. There are also works done based on the clustering algorithms such as k-means [9] and fuzzy c-means [10]. One of the problems with these unsupervised clustering methods is their performance are highly relying on the semantic annotation or the embedded ontology of the service description, so they suffer from the same problems with semantic service matching.

The keyword-based service matching has a difficulty to capture the semantics of the imprecise information from a query or in a service description. It is because the classical set (crisp set) theory is incapable of modeling the imprecise or vague information due to its either true or false membership assignment. It leads to the low precision and recall for the exact match.

Therefore, approximate matching approaches that apply fuzzy sets theory have been introduced to service matching, such as the works done by [6, 7, 11, 12, 13]. In fuzzy sets theory, unlike crisp set, there is a membership function assigning different membership degrees to each element between 0 to 1, which makes it more appropriate to model imprecise information [14, 15]. However, the aforementioned works are based on WADL, Web Services Description Language (WSDL)[2] or other description formats that are not widely adopted.

OpenAPI specification[3] (widely known as Swagger) has become the most popular standard to describe and document RESTful Web services, thanks to its simple syntax and well built platform. However, there lacks study of fuzzy service matching based on OpenAPI. The aim of this paper is to explore REST Web services matching based on the OpenAPI specification, and to realize approximate matching by applying fuzzy logic.

2. Modeling and Matching Index Calculation

2.1. Modeling OpenAPI Description

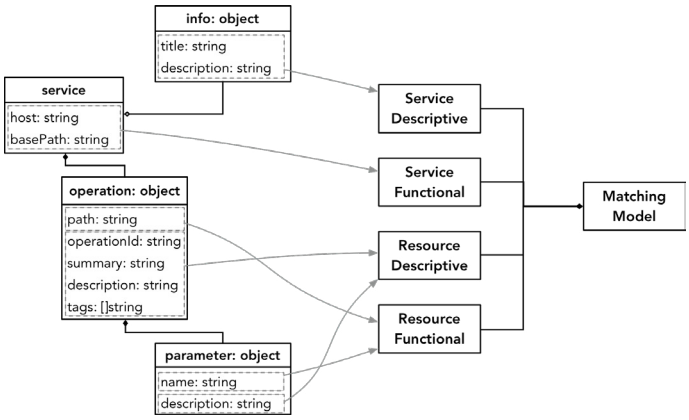


Fig. 1: Service description matching model, properties in an OpenAPI service description are categorized as service descriptive, service functional, resource descriptive and resource functional

For REST architectural style based Web services, a resource is the fundamental element that composes a service [5]. The matching of RESTful Web services is supposed to be considered on the resource-level granularity. Therefore,

[1] https://www.w3.org/Submission/wadl/
[2] https://www.w3.org/TR/wsdl
[3] https://swagger.io/specification/

as a specification for describing RESTful Web services, properties belonging to the service-level and those belonging to the resource-level in an OpenAPI description should be considered differently when doing service matching.

Furthermore, properties should be treated differently not only on the aspects of service and resource. There are properties describing what a service or a resource does, which we can categorize as descriptive properties [16]. There are also properties having no direct relation to what the service or resource is, but they are used for invocation, which we can categorize as functional properties. It is necessary to consider carefully on these differences for the modeling of service properties for matching.

Therefore, the service description model is composed of four parts, which are service-level descriptive properties, service-level functional properties, resource-level descriptive properties and resource-level functional properties. All the properties in an OpenAPI description that belong to any of the four parts will be extracted into the model. Figure 1 shows the allocation of properties in OpenAPI and the service description model for service matching. The property is represented as an object in the OpenAPI specification.

2.2. Matching Index and Score

The service matching approach aims to get a matching index of each service description from a query. The query string matching method and the calculation of matching score of each part in the service matching model intend to be very simple, since the main goal of this paper is to explore the application of fuzzy inference.

The matching index M_{mi} is calculated from the matching scores of service descriptive M_{sd}, service functional M_{sf}, resource descriptive M_{rd} and resource functional M_{rf}:

$$M_{mi} = f(M_{sd}, M_{sf}, M_{rd}, M_{rf}) \tag{1}$$

The function in the equation could be the empirical weighted calculation as:

$$M_{mi} = aM_{sd} + bM_{sf} + cM_{rd} + dM_{rf} \tag{2}$$

or the function could be a fuzzy inference method that will be explained in the next section.

The matching score of each component in the model is the maximum value of string matching ratio between the query and the tokenized text in the service description:

$$\begin{aligned} M_{sd} &= max(M_{sdi}) \\ M_{sf} &= max(M_{sfi}) \\ M_{rd} &= max(M_{rdi}) \\ M_{rf} &= max(M_{rfi}) \end{aligned} \tag{3}$$

The string matching ratio is calculated based on the Levenshtein distance [17], which is computed by the open source program fuzzywuzzy[4].

3. Matching with Fuzzy Inference

When the matching score of each part in the service description model is calculated, it will be used as the input in a fuzzy inference system to calculate a fuzzy matching index for the service description by using fuzzy logic. We discuss these details including the selected fuzzy inference method and its application to the service matching model in this section.

[4] https://github.com/seatgeek/fuzzywuzzy

3.1. Fuzzy Set Theory

Fuzzy set was firstly introduced by professor Lotfi Zadeh [14] in 1965. Fuzzy set theory extends from the classical crisp set theory, in which an element belonging to a set is binary, that is either belongs or does not belong to a set. The crisp set can be defined by a function μ_C: X → {0,1}, which is the characteristic function of the set C if and only if for all x ∈ X:

$$\mu_C(x) = \begin{cases} 1 \text{ if } x \in C, \\ 0 \text{ if } x \notin C. \end{cases}$$

In fuzzy set theory, instead of the case either or not, there is a membership function that assigns every element in the domain a membership degree within the range [0,1]. A fuzzy set A can be expressed as:

$$A = \sum_{x \in X} \mu_A(x)/x$$

where the number given by μ_A (x) over the slash is the value of the membership degree of the element x, while the number under the slash is the value of x. The slash does not mean division here, it means that one element x_i in a fuzzy set has a membership degree of μ_A (x_i) in finite sets.

The membership function maps the input value of the element to a membership degree between 0 to 1. There are simple membership functions that using straight lines, such as triangular membership functions and trapezoidal membership functions. S-function and Gaussian are the other patterns of membership functions that are commonly used.

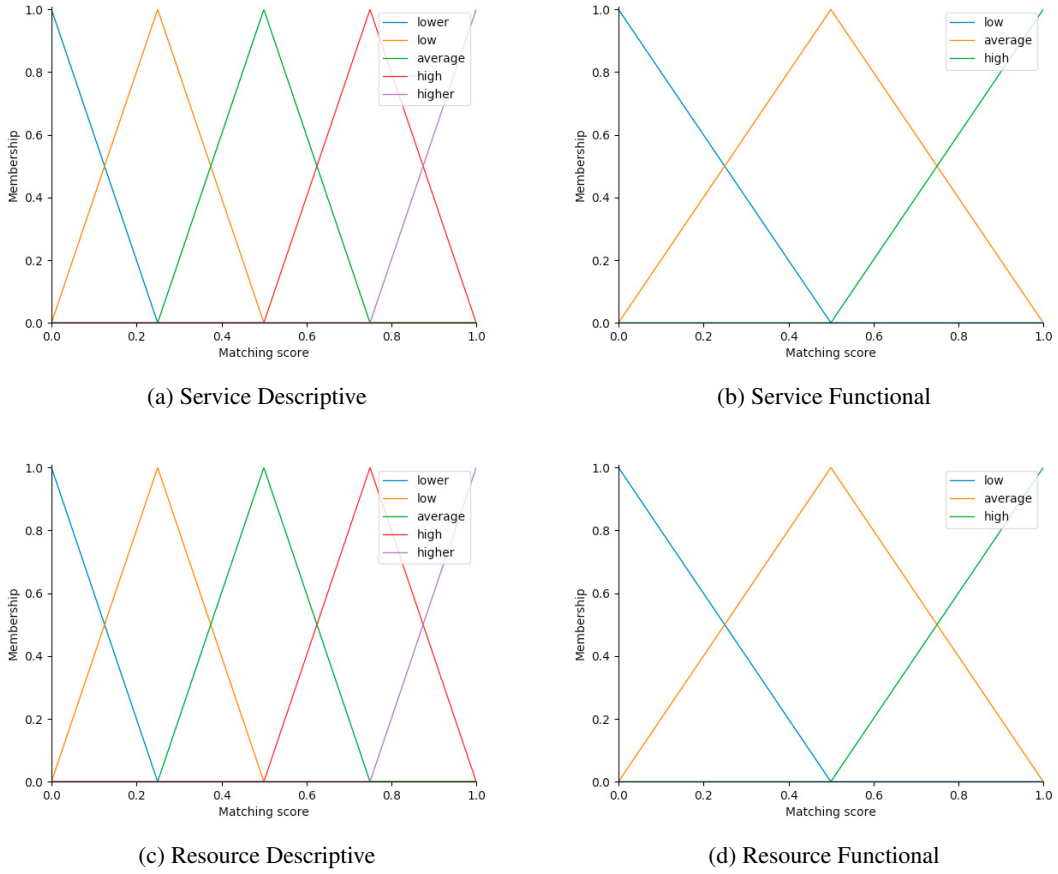
3.2. Fuzzy Control

The fuzzy inference method developed by Mamdani and Assilian [18] is one of the commonly used fuzzy methodologies. It consists of three parts:

1. The **fuzzification** process transforms variables to linguistic variables with different levels, which are expressed by lists of linguistic terms. Since the linguistic terms are interpreted as fuzzy sets then the grades of membership for the sets' elements will be determined by appropriate membership functions.
2. The **rule based processing** procedures uses linguistic IF-THEN statements to link the linguistic terms of input variables to the output state. Each IF-THEN rule is constructed as "If A, then B", in which A is called antecedent and B is called consequence. The mathematical consequences are expressed as a collection of fuzzy sets based on the rules.
3. The **defuzzification** is to dufuzzify the output fuzzy set from the rule based processing procedures. It converts the fuzzy set into a crisp value, which corresponds to the initial crisp input values. The Center Of Gravity (COG) is one of the defuzzification methods.

3.3. Application to Service Matching Model

Here we apply fuzzy inference system to evaluate the matching index for a service description model. The four attributes in the service description model are considered as input variables, and the matching index is the output variable. All the variables are differentiated into levels and expressed by lists of terms, which is the aforementioned transformation to linguistic variables.



(a) Service Descriptive

(b) Service Functional

(c) Resource Descriptive

(d) Resource Functional

Fig. 2: Membership functions of the four input variables

Since the four parts in the model can be classified as descriptive and functional, and the matching preciousness is different between them, therefore we introduce different levels for them. The two descriptive variables are differentiated into five levels, and the two functional variables are differentiated into three levels:

$$M_{sd} = \text{“service descriptive”} = \{M_{sd0} = \text{“lower”}, M_{sd1} = \text{“low”}, M_{sd2} = \text{“medium”}, M_{sd3} = \text{“high”}, M_{sd4} = \text{“higher”}\} \tag{4}$$

$$M_{sf} = \text{“service functional”} = \{M_{sf0} = \text{“low”}, M_{sf1} = \text{“medium”}, M_{sf2} = \text{“high”}\} \tag{5}$$

$$M_{rd} = \text{“resource descriptive”} = \{M_{rd0} = \text{“lower”}, M_{rd1} = \text{“low”}, M_{rd2} = \text{“medium”}, M_{rd3} = \text{“high”}, M_{rd4} = \text{“higher”}\} \tag{6}$$

$$M_{rf} = \text{“resource functional”} = \{M_{rf0} = \text{“low”}, M_{rf1} = \text{“medium”}, M_{rf2} = \text{“high”}\} \tag{7}$$

The membership functions of the four input variables are shown in Figure 2. Figure 3 shows the membership functions of the output variable Matching index. The matching index is differentiated into five levels as well:

$$M_{mi} = \text{“matching index”} = \{M_{mi0} = \text{“very low”}, M_{mi1} = \text{“low”}, M_{mi2} = \text{“medium”}, M_{mi3} = \text{“high”}, M_{mi4} = \text{“very high”}\} \tag{8}$$

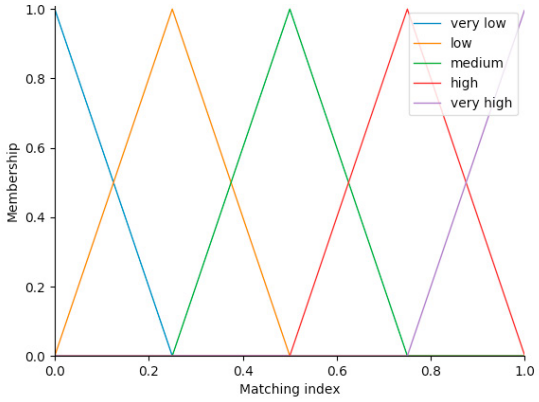


Fig. 3: Membership functions of the output variable matching index

Five fuzzy rules are defined as:

- Rule1: IF SD = “lower” AND RD = “lower” AND SF = “low” AND RF = “low”, THEN MI = “very low”
- Rule2: IF SD = “low” AND RD = “low”, THEN MI = “low”
- Rule3: IF SD = “average” OR RD = “average” OR SF = “average” OR RF = “average”, THEN MI = “medium”
- Rule4: IF SF = “high” OR RF = “high”, THEN MI = “high”
- Rule5: IF SD = “higher” OR RD = “higher”, THEN MI = “very high”

4. Evaluation

4.1. Implementation

To evaluate the fuzzy service matching, we implemented four different matchers. An equal weight matcher has equal weights for all the four parts in the service matching model, that is with coefficients (a=0.25, b=0.25, c= 0.25 and d=0.25). A default weight matcher has weight of 0.3 for the two descriptive parts, and weight of 0.2 for the two functional parts, that is with coefficients (a=0.3, b=0.2, c= 0.3 and d=0.2). A simple fuzzy matcher has simpler membership functions and rules to compare. A fuzzy diff matcher, which is the subject fuzzy matcher, has the rules and different levels of membership functions for input variables and matching index as described in the section 3.3. The fuzzy inference part is implemented with the package `scikit-fuzzy`[5] , and the text matching is implemented with the package `fuzzywuzzy`[6].

4.2. Dataset

The dataset used in the evaluation is composed of Swagger description files gathered from the Web, most of which are retrieved from the API directory service APIs.guru, and a few are manually downloaded from certain Web services. When all the Swagger description files were gathered, a selection process was performed in order to select service descriptions that will be used in the evaluation.

The Swagger files that are duplicated or very similar were removed from the dataset. Those considered with low quality (as decided by 2 criteria) were also removed. One is the Swagger file itself which is poorly created. Another criterion is the number of resources contained in the description file smaller than 10. And the service description files

[5] https://pythonhosted.org/scikit-fuzzy/
[6] https://github.com/seatgeek/fuzzywuzzy

that are not in a stable version were also removed. Thereafter, the size of dataset is reduced from 585 to 301. For getting an operable size of dataset, a random selection is performed to get a dataset with the size of 30.

The final dataset then is labelled manually by looking through the content of service description files for the queries that are going to be performed in the evaluation. Four different queries were then performed, which is *program*, *cloud*, *storage* and *finance*.

4.3. Result

The service matching performance is evaluated in terms of precision and recall rates. Consider a set of relevant service descriptions D_{rel}, which can be regarded as true positives (tp) + false negatives (fn) in retrieving, and a set of service descriptions retrieved D_{ret}, which can be regarded as true positives (tp) + false positives (fp). The intersection of these two sets is the set of retrieved relevant service descriptions D_{rr}, which can be regarded as true positives (tp) in retrieving. So the two measurements are calculated as: Precision $= \frac{D_{rr}}{D_{ret}} = \frac{tp}{tp+fp}$ and Recall $= \frac{D_{rr}}{D_{rel}} = \frac{tp}{tp+fn}$. F1-measure is also calculated to provide a harmonic mean of precision and recall [19]: $F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$.

We tested the fuzzy matcher for 4 different queries with the dataset. All the measurements were calculated in median as central tendency due to the small size. The results will be presented in two parts. We firstly present the results of matching index membership function in the triangular shape. Then the results of matching index in the Gaussian membership function will be presented. For each part, we present a comparison evaluation measurements between the four matchers.

4.3.1. Comparison within Triangular Output Membership Function

We present here the results of F1-measure, precision and recall between two weighted matchers and two fuzzy matchers, where the triangular membership function is used with the fuzzy matchers. We tested the evaluation measurements at slightly different thresholds of matching index. That means if a final matching index of a service description is above the threshold, it is considered as a retrieved document, which is then counted as a true positive for calculating the measurements.

Figure 4 and Table 1 show the results at the threshold of 0.50. We can see that the four matchers have the same recall rate of 0.5. The two fuzzy matchers have higher precision rate and F1-measure than the two weighted matchers. The two fuzzy matchers have the same F1-measure of 0.66667, and the fuzzy diff matcher has the highest precision rate of 1.0.

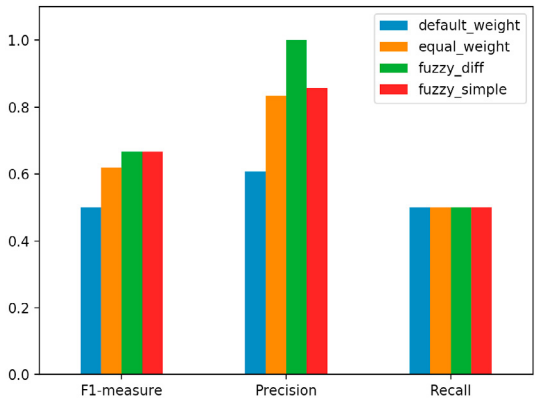


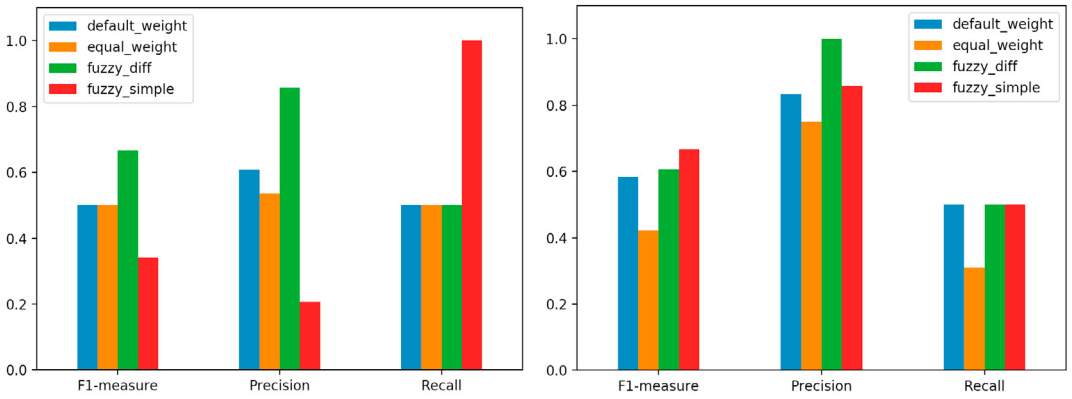
Fig. 4: Graphical comparison of measurements on the four service matchers in terms of median at threshold 0.50, where the two fuzzy matchers have higher precision rate and F1-measure than the two weighted matchers.

Figure 5a and Table 2 show the results at the threshold of 0.49. We can see that the fuzzy diff matcher has the highest F1-measure of 0.66667, and the highest precision rate of 0.85714. The fuzzy simple matcher performed worst in this case, though it has the 1.0 recall rate.

Figure 5b and Table 3 show the results at the threshold of 0.51. We can see that the fuzzy simple matcher has the highest F1-measure of 0.66667, and the fuzzy diff matcher has the highest precision rate of 1.0.

Table 1: Measurements on the four service matchers in terms of median at threshold 0.50

	F1-measure	Precision	Recall
default weight	0.5	0.60714	0.5
equal weight	0.61905	0.83333	0.5
fuzzy diff	0.66667	1.0	0.5
fuzzy simple	0.66667	0.85714	0.5



(a) Measurements at threshold 0.49 (b) Measurements at threshold 0.51

Fig. 5: Graphical comparison of measurements on the four service matchers in terms of median, where the fuzzy diff matcher has the highest F1-measure and precision rate at threshold 0.49, and the fuzzy simple matcher has the highest F1-measure at threshold 0.51

Table 2: Comparison of measurements on the four service matchers in terms of median at threshold 0.49

	F1-measure	Precision	Recall
default weight	0.5	0.60714	0.5
equal weight	0.5	0.53571	0.5
fuzzy diff	0.66667	0.85714	0.5
fuzzy simple	0.3417	0.20696	1.0

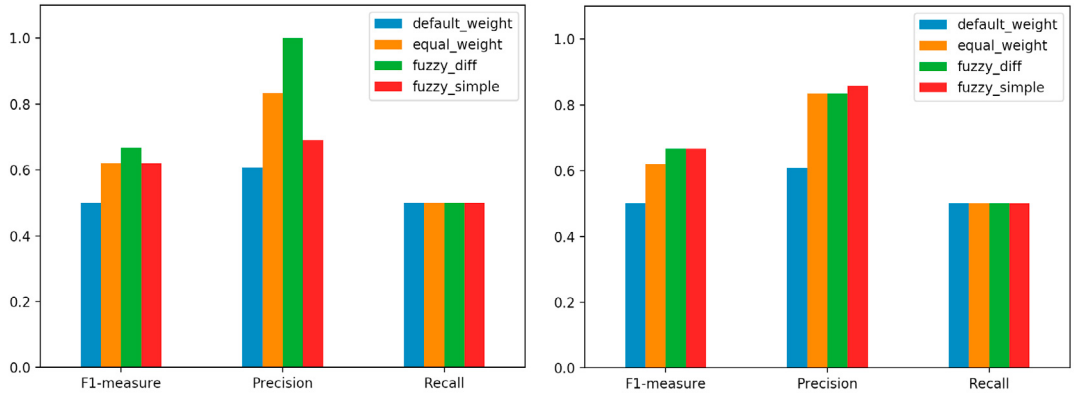
Table 3: Comparison of measurements on the four service matchers in terms of median at threshold 0.51

	F1-measure	Precision	Recall
default weight	0.58333	0.83333	0.5
equal weight	0.42222	0.75	0.30952
fuzzy diff	0.60606	1.0	0.5
fuzzy simple	0.66667	0.85714	0.5

4.3.2. Comparison with Gaussian Curve Output Membership Function

We present here the results of F1-measure between the four matchers with different shapes of output membership functions. Three different shapes were tested. The first is the triangular as the one evaluated in the previous comparison. The other two shapes are Gaussian curves at different standard deviations of 0.2 and 0.3.

Figure 6a shows the results of the Gaussian membership function with standard deviation of 0.2 at the threshold of 0.50. We can see that the fuzzy diff matcher has the highest F1-measure and precision rate. Figure 6b shows the results of the Gaussian membership function with standard deviation of 0.3 at the threshold of 0.50. We can see that the two fuzzy matchers have higher F1-measure than the two weighted matchers.



(a) with standard deviation at 0.2 (b) with standard deviation at 0.3

Fig. 6: F1-measures in median at threshold of 0.50 on the four service matchers with Gaussian curve output membership function

Table 4 shows the comparison of F1-measures among the three shapes of membership functions at threshold of 0.50. We can see that the fuzzy simple matcher performs a little bit lower with Gaussian membership functions at 0.5 standard deviation, the other three matchers stay the same.

Table 4: Comparison of F1-measures in median at threshold 0.50 between different membership functions

	Triangular	Gaussian std 0.2	Gaussian std 0.3
default weight	0.5	0.5	0.5
equal weight	0.61905	0.61905	0.61905
fuzzy diff	0.66667	0.66667	0.66667
fuzzy simple	0.66667	0.61905	0.66667

4.4. Analysis and Discussion

From the results of experiments with different settings, we can see that in most cases, the two fuzzy inference-based service matchers performed slightly better than the two weight based service matchers based on the comparing of medians of measurements. What is unexpected is that, the fuzzy differ matcher with more elaborate inference rules did not outperform the fuzzy simple matcher that with simple inference rules. The fuzzy simple matcher even in one case outperformed the fuzzy diff matcher.

We compared the F1-measures of different shapes of membership functions. However, the results did not show that the Gaussian curve and Triangular shape have big impact on the matching performance. The recall rate in most cases stayed at 0.5, which is not satisfied. The text matching method used in this paper is very simple, which has a big impact on the final results of all the four matchers. A better text matching method such as implementation with a mature search engine would improve much on the matching scores of each part in service matching model. The WordNet approach was also a candidate implementation. However, we found its vocabulary does not fit the text matching of Web service description.

5. Conclusion

In this paper, we built a service matching model for REST services described in OpenAPI specification. The model considers the differences of properties in a service description and differentiates them as two different types of properties in two levels, which results in four parts: service-level descriptive properties, resource-level descriptive properties, service-level functional properties and resource-level functional properties. The four parts are treated differently for service matching. Fuzzy set theory is exploited for being applied in Web service matching. The fuzzy inference method

by Mamdani and Assilian [18] is used to calculate the fuzzy matching index based on the service matching model mentioned.

The evaluation result shows a satisfied performance. But the fuzzy matching approach does not outperform the empirical weighted matching approach significantly. More exploration on different membership functions and fuzzy inference rules may produce a better result for the fuzzy matching approach. This paper deals with applications of existing string matching algorithm and fuzzy inference method. However, it applies the methods to an unexploited problem, which is the fuzzy service matching on the widely used REST service description specification OpenAPI.

However, the text matching method in this paper is very simple. It could be improved by using search engine with mature algorithm such as Lucene. WordNet is not used in this approach due to its lacking of many terms that used in Web service description. And a larger dataset could be used to perform more tests with difference queries, which can produce a more credible evaluation result.

Acknowledgement

We gratefully acknowledge Prof. Elisabeth Rakus-Andersson for her valuable suggestions and discussions.

References

[1] Paweł Ziemba, Jarosław Jankowski, J Watrobski, and Jarosław Becker. Knowledge Management in Website Quality Evaluation Domain. In *Computational Collective Intelligence*, volume 9330, pages 75–85. Springer, Cham, 2015.

[2] Yang Xue, Chunhong Zhang, and Yang Ji. RESTful Web Service Matching Based on WADL. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC'15)*, pages 364–371, sep 2015.

[3] Ulrich Lampe, Stefan Schulte, Melanie Siebenhaar, Dieter Schuller, and Ralf Steinmetz. Adaptive matchmaking for RESTful services based on hRESTS and MicroWSMO. In *Proceedings of the 5th International Workshop on Enhanced Web Service Technologies - WEWST '10*, pages 10–17, New York, New York, USA, 2010. ACM Press.

[4] Yongju Lee. Semantic Matching and Resource DiscoveryAlgorithms for RESTful Web Services. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(9), jan 2013.

[5] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[6] Simone A. Ludwig. Fuzzy match score of semantic service match. In *IEEE International Conference on Fuzzy Systems*, pages 193–199, 2008.

[7] Li Bai and Min Liu. Fuzzy sets and similarity relations for semantic web service matching. *Computers & Mathematics with Applications*, 61(8):2281–2286, apr 2011.

[8] Zhichao Peng, Wenhua He, and Daiwu Chen. Research on fuzzy matching model for semantic Web services. In *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, pages 440–445. IEEE, nov 2008.

[9] Juan Manuel Rodriguez, Alejandro Zunino, Cristian Mateos, Felix Oscar Segura, and Emmanuel Rodriguez. Improving REST Service Discovery with Unsupervised Learning Techniques. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 97–104. IEEE, jul 2015.

[10] Giuseppe Fenza, Vincenzo Loia, and Sabrina Senatore. A hybrid approach to semantic web services matchmaking. *International Journal of Approximate Reasoning*, 48(3):808–828, aug 2008.

[11] Kuo-Ming Chao, M. Younas, Chi-Chun Lo, and Tao-Hsin Tan. Fuzzy Matchmaking for Web Services. In *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 2, pages 721–726. IEEE, 2005.

[12] Reihaneh Rabbany Khorasgani, Eleni Stroulia, and Osmar R. Zaiane. Web service matching for RESTful web services. In *2011 13th IEEE International Symposium on Web Systems Evolution (WSE)*, pages 115–124. IEEE, sep 2011.

[13] Jarosław Jankowski, Przemysław Kazienko, Jarosław Wątróbski, Anna Lewandowska, Paweł Ziemba, and Magdalena Zioło. Fuzzy multi-objective modeling of effectiveness and user experience in online advertising. *Expert Systems with Applications*, 65:315–331, dec 2016.

[14] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, jun 1965.

[15] Yang Liu, Qinglei Zhou, Elisabeth Rakus-Andersson, and Guohua Bai. A Fuzzy-Rough Sets Based Compact Rule Induction Method for Classifying Hybrid Data. In *Proceedings of the 7th international conference on Rough Sets and Knowledge Technology (RSKT'12)*, pages 63–70. Springer, Berlin, Heidelberg, 2012.

[16] Ehsan Sharifi, Reza A. Moghadam, Fernando Bobillo, and Mohamad M. Ebadzadeh. A fuzzy framework for Semantic Web Service description, matchmaking, ranking and selection. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 621–625. IEEE, jul 2011.

[17] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady, Vol. 10, p.707*, 10:707, 1966.

[18] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, jan 1975.

[19] David M. W. Powers. What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. Technical report, mar 2015.