International Journal of Web Engineering 2012, 1(1): 1-5
DOI: 10.5923/j.web.20120101.01

Web Service Matchmaking Using Web Search Engine and Machine Learning

Incheon Paik*, Eigo Fujikawa

School of Computer Science and Engineering University of Aizu, Aizu-Wakamatsu, Fukushima, Japan

Abstract Web Services discovery that locates adequate services, has been studied very actively for better quality of service retrieval. Starting from conventional keyword matching, logic-based matching and combination of the methods with information retrieval approach have been proposed to enable better discovery performance. The combining method using term-similarity can overcome the decision failure when the keyword or the logic-based methods were applied, and it was shown that the methods outperform the existing methods. And researches to aggregate matchmaking variants by machine learning has been attempted, and it also improves the discovery performance. The approaches still suffer from fixed corpus set for term similarity calculation. In this research, we attempted to calculate the similarity based on search engine to reflect the current Web context. Tokenized terms are used for the matchmaking degree. Variants for the matchmaking from ontology and term similarity are aggregated using Support Vector Machine (SVM) with non-linear kernel function. Matchmaking test on the trip domain service discovery was conducted. Experimental result based on the standard measure of precision and recall rate for the top 1-20 services of matched result on the trip domain test set are shown.

Keywords Service Discovery, Match Making, Machine Learning, Semantic Similarity

1. Introduction

Service discovery locates matched service for requesters' queries or various kinds of service compositions. Specific registries such as Universal Description Discovery and Integration (UDDI), services on Web and dedicated database, even service search/retrieval system can be considered as registries to provide service corpus. It is one of the hottest issue to retrieve the best matched services semantically to a given query. Service discovery will highly contribute to performance of service composition[11,12]. Automatic discovery of correct service instances for abstract services is essential for automatic service composition too. An important issue for successful and effective retrieval of relevant services in the future semantic Web is how well discovery agents perform semantic matching in a way that goes far beyond standard service discovery method.

Fundamental principle of matchmaking is calculating vocabulary and semantic matching degree of terms in service description which consists of operation, input/output parameters. As follow-up research for the service discovery, an approach that aggregate results gotten from the several variants using machine learning is shown by[8]. The idea will allow concise integration of results from multiple sources for solution for the matching. Applying to different situations is very effective for evaluation and improvement. The approaches[3,7] to measure similarity are based on dictionary or fixed corpus, and it results in weak for latent semantic relations. Web based measuring method provide good result for latent semantics of the terms. In this paper, we attempt to apply two changes on similarity measuring method using web based corpus and aggregation way for above discovery method.

As similarity measuring way, web based measuring method Web-PMI, Web-Dice, Web-Jaccard, and Web-Overlap proposed by[1] are used for computation. Aggregation by SVM are used for results from above information retrieval methods and similarity measuring method with WordNet.

We try to realize this approach on trip domain and show precision/recall as the quality of discovery by using services and training set for machine-learning which are specifically prepared for this experiment.

2. Related Work

2.1. Semantic Similarity between Words

Semantic similarity measures play important roles in information retrieval. Bollegala[1] shows a similarity measuring method using page-count and text-snippets that have gotten from web search engine. This method had given us the way that leverage widely collected information and knowl-

* Corresponding author:
paikic@u-aizu.ac.jp (Incheon Paik)
Published online at http://journal.sapub.org/web

edge, that is not available from an exactly defined knowledge context such as dictionary or ontology description for specific domain. In this research, Web-PMI, Web-Jaccard, Web-Dice, and Web-Overlap measuring are used for similarity computation between parameter-types which is represented with multiple keyword and appears in service or request descriptions. The term similarity method has been considered to calculate matching degree of service[7].

2.2. Aggregation of Service Constitutions Using Machine Learning

An approach that integrates various similarity measuring result between services and requests using SVM was proposed by[8]. SVM is widely used on the field of machine-learning for categorization. In SAWSDL-Mx2[8], this approach had been used for integration of ontological matching result and some information retrieval results. Our approach that uses several different text similarity measuring method already described in section 2.1. Semantic similarity, also uses this integration for only the text similarity measuring results. Separation of ontological matching and text based similarity matching, will allow to improvie the matching quality.

2.3. Service Discovery and Clustering

Services for several domains have been increased, and a large scale service network to simulate the service situation based on complex network. A composition approach has been developed using planning approach on the service network[10]. A data-driven service composition approach has been introduced using service data correlation model, which motivates investigation of service discovery[6].

A repository has been created to find proper biomedical services. The repository consists of clusters that have created based on functional similarity by information retrieval technology[13,14].

3. Service Similarity Measure and Aggregation

3.1. Service Description

We defined our dedicated format for description of service and request for our analysis, which can be extracted from standard web service description format WSDL produced by W3C. In this research, service discovery algorithm has been implemented based on above service and request description. Operation specifically declared as a part of web service and query description, had been assumed as having *OutputType* stand for returned data type from the operation, *OperationName* just the name of operation, and *Inputs* that means the list of input parameters given as arguments of the operation on the structure. Each feature is represented as *ParameterType* in this research. In real operation, *OperationName* is not shown as *ParameterType*. However, *OperationName* has same characteristics with other feature on *Operation* means that *OperationName* having both of *name* and *keyword-list*. Therefore, all of the feature *OutputType*, *OperationName*, and *Input* has been represented using *ParameterType* for simplifying the implementation.

Description detail which used in this research about *Operation* and *ParameterType* are shown as following format.

Definition 1 (Operation)

Operation:= OutputType OperationName Inputs

Where,

OutputType: ParameterType

OperationName: ParameterType

Inputs: ParameterType*

Definition 2 (ParameterType)

Parameter name: String

Keyword list: String+

We assume that the information extraction is available from standard WSDL description also, and each operation which is contained into real service can provide each keyword based information.

3.2. Operation Matching

In our analysis, match-making between operations which have been executed based on above service/request description has a structure *Figure1* on the match-making processing. *OperationComparer* which is taking two input request and service provides final comparison result by using *ParameterTypeComparer*. The module provides a similarity between *ParameterType* as sum of keyword similarity computed by using *KeywordComparer* based on each information retrieval method. In previous research, The works [5,7,8] show ontological matching on the approach, but our approach excludes the ontological matching and tries to evaluate the quality and worth of information retrieval potential on exactly required matchmaking service discovery.

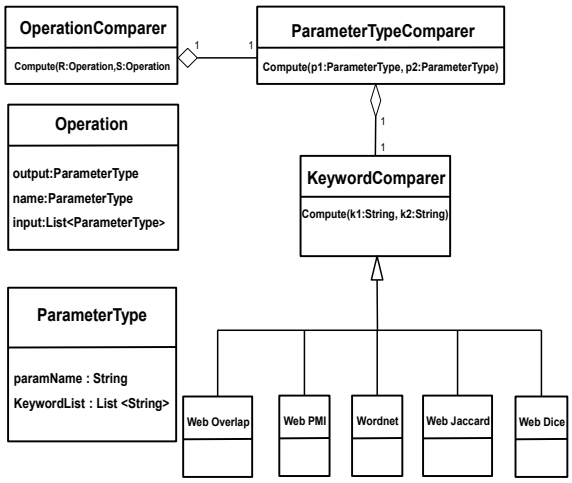

Figure 1. Comparer Structure for Operation Matchmaking

3.3. Similarity Measuring between Parameter

We attempted to apply search engine based 4 algorithms

called as Web-PMI, Web-Jaccard, Web-Dice and Web-Overlap shown by[1], for similarity measuring between keywords of each ParameterType. In general, these specific declared parameter-type used on the web service description have been compared by using also specific declared ontology definition. In the case that exactly declared only method, it's difficult to detect a pattern which means various different expression and frequently occurred in many services but both is same semantic having. This method aims to cover above differences between parameter-types by using web resource as widely purpose used resource.

The four equations use page-count result from search engine. In this research, google[15] search engine API produced by google cooperation was used for getting the page-count. Google search engine has been widely used from many researchers and developer. Thus, we chose it for evaluation of the possibility on service discovery field. This suggestion that use search engine for service discovery aims at an evaluation retrieval quality, and tests whether these web search engine based method are sufficient or not.

$$\mathrm{WebJaccard}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \le c \\ \dfrac{H(P\cap Q)}{H(P)+H(Q)-H(P\cap Q)} & \text{otherwise} \end{cases} \quad (1)$$

$$\mathrm{WebOverlap}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \le c \\ \dfrac{H(P\cap Q)}{\min(H(P),H(Q))} & \text{otherwise} \end{cases} \quad (2)$$

$$\mathrm{WebDice}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \le c \\ \dfrac{2H(P\cap Q)}{H(P)+H(Q)} & \text{otherwise} \end{cases} \quad (3)$$

$$\mathrm{WebPMI}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \le c \\ \log_2 \dfrac{H(P\cap Q)/N}{H(P)H(Q)/N^2} & \text{otherwise} \end{cases} \quad (4)$$

3.4. SVM Result Aggregation

3.4.1. Feature Vector Specification

In this paper, as shown by SAWSDL-MX result[8], an integration approach using SVM has been used for aggregation of multiple information retrieval algorithms and binary classification to categorize test data as two categories such as *similar* or *dissimilar*. Machine learning carries our classification with using multiple ontological results directly. In this experiment, we propose new training feature vector shown as means for service specific format.

More detailed explanation about each feature is shown as follows.

(Element 1) All

This feature means the sum of results getting from 6 algorithms used as operation matching in the experiment. This feature will take a value between 0.0 and 6.0.

(Element 2) OutputType

This feature means the sum of OutputType comparison results. Each applied 6 algorithms is showing matching result for OutputType of the query and that of service.

(Element 3) OperationName

The value of this feature has been led using same rule with above OutputType. This feature means the sum of OperationName comparison results.

(Element 4-13) Input #N

The value of this feature has been led using same rule with above OutputType. This feature means the sum of Input #N comparison results.

(Element 14) Input Count Equivalence

If the count of inputs of the query and the service, this feature is 1. And, the other cases this feature is -1.

Table 1. Feature Vector for SVM

Element Index of Vector	Contents of the Feature
1	All(0.0-6.0)
2	OutputType results
3	OperationName results
4 – 13	Input #1 results, ... Input #10 results
14	Input Count Equivalence(1, -1)

3.4.2. Sample Feature Vector for Relevant Pair

Table 2 show sample relevant pair and feature vector getting from that. Each features are according to already specified format in Table 1, and values of that are computed and converted by using each algorithms applied in this research. In the experiment, all of the pair of request and service which are wanted to get the relevance, are converted to Table 1 format like sample and the feature vector will be classified by SVM.

Table 2. Sample of Request and Service Pair

● For Request:

Parameter Type Usage	Parameter Type Name	Keyword List
Output Type	TrainInformation	Train information
Operation Name	getTrainInformation	Get train information
Input #1	DepartureCity	Departure city
Input #2	ArrivalCity	Arrival city
Input #3	DepartureDate	Departure date

● For Service:

Parameter Type Usage	Parameter Type Name	Keyword List
Output Type	TrainList	Train list
Operation Name	getTrainList	Get train list
Input #1	DepartureStation	Departure station
Input #2	DestinationStation	Destination station
Input #3	DepartureDate	Departure date

4. Implementation and Experiment

The implementation for calculating term similarity and SVM aggregation is developed in Java. WordNet2.0 and Jena are also used for handling ontology and standard terms in Java.

4.1. Environment Setting

In this experiment, we prepared 415 services for testing of

our discovery suggestion on *Trip Domain*. These services have several different features. Keyword *get/search/find* for information taking web service assumption, *train/bus/ airplane/hotel/taxi* for service purposes, and *list/data/result/ information* for data format which is returned by operation are used for making these services by combination. Preparation of these keyword differences aims at considering appearance on the real web services in the future. And we prepare and use 400 training data as training set for SVM. By above setting, we attempted experiments for the discovery algorithm with virtually prepared 100 queries.

```
Discovery Engine execution started.
Executing training ... finished.
Loading Services ... finished.

Query: TrainInformation getTrainInformation
            (DepartureCity, ArrivalCity, DepartureDate);

Retrieved services
    No 01 => TrainInformation findTrainInformation
            (DepartureStation, DestinationStation, DepartureDate);
    No 02 => TrainList getTrainList
            (DepartureStation, DestinationStation, DepartureDate);
    No 03 => TrainList obtainTrainList
            (StartCity, DestinationCity, StartDate, SeatType);
    No 04 => TrainList getTrainList
            (StartStation, ArrivalStation, DepartureDate, SeatType);
    No 05 => TrainInfo findTrainInfo
            (StartStation, EndStation, StartDate);
```

(a) Query for Retrieving Train Information

```
Discovery Engine execution started.
Executing training ... finished.
Loading Services ... finished.

Query: BusReservationData reserveBus
            (BusID, DepartureStop, DestinationStop, SeatType);

Retrieved services
    No 01 => BusReservationResult reserveBus
            (BusID, DepartureStop, DestinationStop, SeatType);
    No 02 => BusReservationInfo reserveBus
            (BusName, DepartureStop, DestinationStop);
    No 03 => BusBookingData bookBus
            (BusID, DepartureStop, DestinationStop, SeatType);
    No 04 => BusBookingData bookBus
            (BusID, StartStop, EndStop, SeatType, StartDate);
    No 05 => BusReservationResult reserveBus
            (BusID, StartStop, EndStop, SeatRank);
```

(b) Query for Bus Reservation

Figure 2. Example of Discovery Result

4.2. Result and Evaluation

4.2.1. Example of Discovery Execution

Figure 2 gives some examples of execution in the experiment. Top 5 services of retrieval results are returned from discovery engine. *Table2* shows one request query as an example, and the scenario that executes retrieving trial for *getTrainInformation*, is contained in *Figure 2(a)*. As another case of retrieval on the *Trip Domain*, the result of one query posting result for operation *reserveBus* is shown in *Figure2 (b)*. Each service which is returned as relevant set, can be seen as similar to each operation. These are a few example for indicating the execution and environment of experiment.

Table 3. Sample of Feature Vector

Elements Index	Feature Contents	Value
1	All	2.26
2	Output Type	27.57
3	Operation Name	21.08
4	Input #1	22.82
5	Input #2	22.82
6	Input #3	22.82
7-13	Input #7-13	10000.0
14	Input Count Equvalence	1

We use our characteristic format for description of service and request on this experiment, and standard web service description forma

4.2.2. Evaluation

Recall/Precision and F-measure getting from above services and queries is shown as *Table 2*. The result are gotten as the average score of operation matching with queries that explained in the environment setting. Each turning point on the figure means Top-1, Top-3, Top-5, Top-10 and Top-20.

This evaluation of correctness is judged by human evaluation like common search engine results evaluation. Therefore, the 88 percent of the retrieved data can be useful for user's query.

Table 4. Recall / Precision / F-measure of the System

Returned Count N	Recall	Precision	F-measure
Top 1	1.00	0.98	0.98
Top 3	0.67	0.98	0.79
Top 5	0.60	0.96	0.73
Top 10	0.40	0.92	0.55
Top 20	0.35	0.88	0.50

5. Conclusions and Future Work

In this paper, Web search engine based term similarity measure and results integration by SVM are attempted to discover matched service on the trip domain. The similarity calculation measure using search engine provide latent terms hided in the Web document so that it was helpful to give more flexibility to find close terms for terms in a query. Average precision in Top 20 result data obtained is 94.4% for 100 test queries as an experiment result, and it is very high score compare to other literatures' result [7][8] on a specific domain of the trip.

In the future, we will carry out the experiment under multiple domains considering other term similarity calculation measures such as Formal Concept Analysis or Association Rule Mining with Ontology Learning that are recent tries to improve similarity measurement.

REFERENCES

[1] D. Bollegala, Y. Matsuo, and M. Ishizuka. ``Measuring Se-mantic Similarity between words using web search engines," in Proc. 16th International World Wide Web Conference (WWW2007), May 2007.

[2] C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03).

[4] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification, 2007.

[5] C. Kiefer and A. Bernstein. The creation and evaluation of isparql strategies for matchmaking. In Proceedings of the 5th European Semantic Web Conference (ESWC), Lecture Notes in Computer Science, volume 5021, pages 463–477. Springer-Verlag Berlin Heidelberg, 2008.

[6] Gu, et al., "Service Data Correlation Modeling and Its Application in Data-Driven Service Composition" IEEE Transactions on Services Computing, vol. 3, no. 4, pp. 279–291, 2010.

[7] M. Klusch, B. Fries, and K. Sycara. ``Automated semantic web service discovery with owls-mx," in Proc. 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Hakodate, Japan. ACM Press, 2006.

[8] M. Klushch, P. Kapahnke, and I. Zinnikus. ``SAWSDL-MX2: A Machine-learning approach for integrating semantic web service matchmaking variants," in Proc. IEEE International Conference on Web Services, 2009.

[9] Y. Lee and C. Kim, A Learning Ontology Method for RESTful Semantic Web Services, Proceedings of the IEEE International Conference on Web Services (ICWS 2011), Washington D.C., U.S.A, Jul. 2011.

[10] S. Oh, D. Lee, and S.R.T. Kumara, "Effective Web Service Composition in Diverse and Large-scale Service Networks" IEEE Transactions on Services Computing, vol. 1, no. 1, pp. 15–32, 2008.

[11] K. Sycara, M. Paolucci, A. Anolekar, and N. Srinivasan. Automated discovery, interaction and composition of se-mantic web services. Web Semantics, 1(1), Elsevier, 2003.

[12] K. Sycara, M. Paolucci, and J. Soundry N. Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. IEEE Internet Computing, 8(3):6673, May 2004.

[13] W. Tan, et al., "ServiceMap: Providing Map and GPS Assistance to Service Composition in Bioinformatics", Proceedings of the IEEE International Conference on Web Services (ICWS 2011), Washington D.C., U.S.A, Jul. 2011.

[14] J. Zhang, et al., "Toward Semantics Empowered Biomedical Web Services", Proceedings of the IEEE International Conference on Web Services (ICWS 2011), Washington D.C., U.S.A, Jul. 2011.

[15] Google Web Search API, https://developers.google.com/web-search/docs/ , 2010