ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013

Semantic Matching and Resource Discovery Algorithms for RESTful Web Services

Dr.Yongju Lee

School of Computer Information, Kyungpook National University, Sangju, Korea

ABSTRACT: Recently, the implementation architecture of mashups has changed from the existing SOAP-based style to the REST style. This is mainly because the REST style has several advantages: lightweight, declarative and easy to access. The growing number of available RESTful web services raises a challenging search problem: how should the desired web services be located. This paper proposes semantic matching and resource discovery algorithms based on the ontology learning method for RESTful web services. These algorithms allow mashup developers to automate the discovery and composition of RESTful web services eliminating the need for programmer involvement.We describe an experimental study on a collection of 168 RESTful web services. The experimental results show that our approach achieves up to 30% improvement for recall performance, and up to 18% for precision performance compared to the keyword searching method.

K*eywords*: Semantic matching, resource discovery, RESTful web service, similarity, ontology.

I. INTRODUCTION

With the advent of Web 2.0, the use of RESTful web servicesisexpected to overtake that of the traditional SOAP-based web services. A RESTful web service is a simple web service implemented using HTTP and the principles of REST (Representational State Transfer) [1].The growing number of RESTful web services available on the web raises the challenging issue of how to locate the desired web services.However, the existing keyword searching methods are insufficient for the bad recall and the bad precision. Adding semantics to RESTful web services may help to overcome these limitations.

Although there are several researches that add semantics to SOAP-based web services [2]-[5], the addition of semantics to RESTful web services poses a greater challenge. Most RESTful web services do not require a description language such as WSDL (Web Service Description Language), as the principal objective of REST is simplicity. The lack of this description language makes it difficult to achieve the automated discovery and composition of the web services. With RESTful web services gaining more popularity on the web, the interest in RESTful semantic web services is growing.

We recently proposed an ontology learning method to build semantic ontologies automatically [6]. We extend the ontology learning method to support thesemantic discovery and automatic composition of RESTful web services (which are essential techniquesfor RESTfulsemantic web services). We first introduce the ontology learning method and propose the semantic matching and composable resource discovery algorithms for RESTful web services. A significant key issue is how to locate the desired web services. The efficient discovery can play a crucial role in conducting the composition of web services.

The remainder of this paper is organized as follows. In Section 2, we begin by introducing the ontology learning method.We present the semantic matching algorithmin Section 3 and the composable resource discovery algorithmin Section 4.We describe our experimental evaluationin Section 5. Finally, we discuss related work in Section 6 and conclude the work in Section 7.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013

II. OVERVIEW OF ONTOLOGY LEARNING METHOD

The successful employment of semantic web services is dependent on the availability of highquality ontologies. Building such ontologies is difficult and costly, thus hampering web services deployment. Our ontology learning method [6] automatically generates ontologies from WADLs (Web Application Description Languages)[7] and their underlying semantics.

A. PARAMETER CLUSTERING

We consider the syntactic information that resides in WADLs, and apply a mining algorithm to obtain their underlying semantics. The key ingredient of this technique is to cluster parameter names in the collection of web services into semantically meaningful concepts. We utilize the heuristic as the basis of our clustering, in that parameters tend to express the same concept if they frequently occur together. This allows us to cluster parameters by exploiting the conditional probability of their occurrences in the requests and responses of web service resources. Specifically, we are interested in the association rules of the form $R{:}t_1 \Longrightarrow t_2$, where t_1 and t_2 are two terms. The problem of association rules can be computed by using the well-known Apriorialgorithm.

The support is the probability that t_1 and t_2 occur in a request/response. The confidence is the probability that t_2 occurs in a request or responses, given that t_1 is known to occur in it. Ideally, parameter clustering results should have the following two features: the cohesion of a concept (the connections between parameters inside the concept) should be strong, and the correlation between concepts (the connections between parameters in different concepts) should be weak. We say that t_1 is closely associated to t_2 if the confidence of the rule $R{:}t_1 \Longrightarrow t_2$ is greater than a threshold value δ (i.e., $conf(R{:}t_1 \Longrightarrow t_2) > \delta$).To measure the overall quality of a clustering, we define score = coh/cor where coh and cor are the average of all the cohesion and correlation values, respectively. Our goal is to obtain a high score that will reflect tight connections inside the clusters but loose connections between clusters.

Our clustering algorithm is a refinement of a classical agglomerative hierarchical clustering. We sort the association rules in descending order first by confidence and then by support. At each step, the algorithm selects the highest ranked rule that has not been previously considered. If the terms in the rule belong to different clusters, the algorithm merges the clusters. We then compute the coh/corscore. There are two options: one is the score for the merged result in the previous step, and the other is the score for a new result reflecting the merger of two similar clusters. We select the option with a higher score. This process is repeated until eventually all result clusters satisfy the best score.

B. PATTERN ANALYSIS

Our pattern analysis technique captures the relationships between the terms contained in a parameter name, and match parameters if both the terms are similar and the relationships are equivalent. This approach is derived from the observation that people employ similar patterns when composing a parameter name out of multiple terms. In order to characterize these patterns, all 8209 parameter names from 168 RESTful web services pulled off the Internet were categorized into several buckets. As shown in Table 1, 2435(30%), 752(9%), 608(7%), 472(6%), and 368(5%) parameters were defined as the noun phrases $Noun_1$+$Noun_2$, *Adjective+Noun*, *Verb+Noun*, $Noun_1$+$Noun_2$+$Noun_3$, and $Noun_1$+*Preposition*+$Noun_2$, respectively. There were 3574(43%) parameters not covered by any of the rules. The majority of them (3548 parameters) contain only one token (e.g., city), the others (26 parameters) cannot be tokenized according to the rules defined in Table 1.

Table 1: Pattern of multiple-words.

Rule	Pattern	Occurrence	Example
1	$Noun_1$+$Noun_2$	2435(30%)	companyID
2	*Adjective+Noun*	752(9%)	highTemperature
3	*Verb+Noun*	608(7%)	updateList
4	$Noun_1$+$Noun_2$+$Noun_3$	472(6%)	telephoneAreaCode
5	$Noun_1$+*Preposition*+$Noun_2$	368(5%)	passwordOfAccount

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013

The first step in this approach is to capture the relationships between the terms in a parameter and save them in the ontology. From the above table, the rules in Table 2 are defined to transform the terms defined in the WADL file into ontological concepts and relationships.

Table 2: Relationship between terms.

Rule	Pattern	Relationship	Example
1	*$Noun_1$+$Noun_2$*	Parameter **propertyOf**$Noun_1$	companyID**propertyOf**company
2	*Adjective+Noun*	Parameter **subClassOf**Noun	highTemperature**subClassOf**Temperature
3	*Verb+Noun*	Parameter **subClassOf**Noun	updateList**subClassOf**List
4	*$Noun_1$+$Noun_2$+$Noun_3$*	Parameter **propertyOf**$Noun_1$	telephoneAreaCode**propertyOf**telephone
5	*$Noun_1$+Preposition+$Noun_2$*	Parameter **propertyOf**$Noun_2$	passwordOfAccount**propertyOf**Account

The ontology is generated from the set of parameters per the above rules. The next step is to match a query and the ontology. By the definitions, two ontological concepts are matched if and only if one of the following is true:

- One concept is a property of the other concept(e.g.,companyID**propertyOf**company)
- One concept is a subclass of the other concept(e.g., highTemperature**subClassOf**Temperature)
- Concepts are synonyms of each other (e.g., state **equivalent** province)

Based on the above conditions, our method redefines which terms are actually connected to each other. For example, Assume that cityName is to be compared against codeOfCity. The keyword search would not count these as a possible match. However, if the city term had the relationships“X**propertyOf**Y” in its pattern rule, the matching logic will return a matching score because these two parameters are closely related (perhaps using the rules “cityName**propertyOf**city” and “codeOfCity**propertyOf**city”). For details, readers may refer to our previous work [6].

III. SEMANTIC MATCHING ALGORITHM

In this section we describe how to predict the similarity of RESTful web service resources. Intuitively, we consider resources to be similar if they take similar inputs and produce similar outputs, and if the relationships between the inputs and outputs are similar. We identify a resource with a vector: resource = <description, request, response>. We determine a similarity by combining the similarities of individual vector components. Here, we describe how to compute the similarities for each of the components.

To compute the similarity of the description, we consider the tokenized resource path name and description as a bag of words, and utilize the TF/IDF method [8]:

$$\mathrm{Sim}(Q,D)=\frac{\sum_{k=1}^{t} w_{qk}\times w_{dk}}{\sqrt{\sum_{k=1}^{t}(w_{qk})^2\times\sum_{k=1}^{t}(w_{dk})^2}}$$

wherew_{dk} (or w_{qk}) represents the weight of term t_k in description D (or query Q). In order to compute the similarity of the concepts represented by the request/response, we first find all clusters that include the term of the tokenized request/response parameters. Algorithm 1 shows the process of finding cluster terms procedure. All terms in the matched clusters are the relevant terms for each tokenized parameter. We replace each tokenized parameter with its corresponding cluster, and then compute asimilarity score. The *similarity score* is defined to select the best matches for the given request/response. Consider a pair of candidate matching requests/responses: $A=(a_1,a_2,\cdots a_i,\cdots a_m)$and $B=(b_1,b_2,\cdots b_j,\cdots b_n)$, where a_i and$b_j$denote parameters. The similarity between A and B is given by the following formula:

$$\text{Similar}(A,B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\text{MaxSim}(a_i,\ b_j)}{m+n}$$

where, $\text{MaxSim}(a_I, b_j) = \max\{\text{Sim}(a_i, b_j)\}\text{for all} 1 \le j \le n, i = 1, 2, \cdots m.$

Algorithm 1: Finding cluster terms

```
1  clusterTerms = Ø
2  for each C_i in clusterSet
3      if C_i includes term then
4          for all terms in C_i
5              terms are added to clusterTerms
6          endfor
7      endif
8  endfor
9  return clusterTerms
```

We use a linear combination to combine the similarity of each component. Each type of similarity is assigned a weight based on the user's confidence. The overall similarity is calculated by using Similar_{des},Similar_{req}, and Similar_{res}:

$$\text{Similarity} = \frac{\alpha(\text{Similar}_{des}) + \beta(\text{Similar}_{req}) + \gamma(\text{Similar}_{res})}{\alpha + \beta + \gamma}$$

whereSimilar_{des},Similar_{req}, and Similar_{res} are the description, request, and response similarity, respectively. These components return a real value between 0 and 1, indicating the degree of similarity. The weights α,β, and γ are real values between 0 and 1; they indicate the degree of confidence. High weight values indicate the user's confidence. Please note that the sum of the weights does not have to add up to 1; in our experiments we use equal weights.

Our semantic matching algorithm based on the clustering technique can improve the recall performance of the search engine by introducing semantically meaningful concepts. All matches whose similarity scores exceed 0 are assigned to a candidate set. To select the best matches from the candidate set, however, an additional ontological pruning step is required. Since theparameter clusteringtechnique of Section 2 considers all terms in a cluster as an equivalent concept and ignores hierarchical relationships between the terms, matches might exist that are irrelevant to the user's intention (i.e., false positives). Thus, the *pruning process* is necessary to improve the precision.

The basic idea to improve the precision of the semantic matching algorithm is to apply the pattern rules defined in Table 2. A query is matched against all resources stored in the repository using pre-defined matching rules. Our algorithm proceeds in a Greedy fashion. If two concepts are matched to the rules, the weight is set to 1. If two concepts are not matched to the rules, then the weight is set to 0 and they are removed from the results. The core procedure for the semantic matching algorithm is shown in Algorithm 2.

Algorithm 2: Semantic matching

```
1  for each resource S in repository
2    Compute Similar_des=Sim(Q_des, S_des)
3    for each term in query request/response
4      Get clusterTerms from Algorithm 1
5      Replace term with culsterTerms
6    endfor
7    Perform pruning process
8    Compute Similar_req and Similar_res
9    Compute Similarity
10 endfor
```

IV. COMPOSABLE RESOURCE DISCOVERY ALGORITHM

Given a query and a collection of RESTful web services stored in the repository, automatically finding a resource from the repository that matches the query requirement is the resource discovery problem. For example, we are looking for a resource to search a hotel. Table 3 shows the request/response parameters of a query and a resource. In this example a service resourceS satisfy the query Q. Q requires hotelName as the response and S produces hotelName and confirmNumber. The extra response produced can be ignored. S requires countryCode and nameOfCity as the request and Q provides countryID, stateName, and cityName as the request. Aparameter can be matched with the other parameter only if there is a semantic relationship between them. Here, although countryCode and countryID are different forms, they have the same semantics since they are referred to the same concept. Also nameOfCity and cityName have the same semantics since they are properties of the same object (i.e., city). Therefore, the agent is able to infer that Q and Srequest parameters have semantically the same classes.

Table 3: Example of composableresource discovery.

Resource	Request Parameters	Response Parameters
Q	countryID stateName cityName	hotelName
S	countryCode nameOfCity	hotelName confirmNumber

We describe an algorithm, which issimilar to the one in [9],to support the composition of RESTful web services using semantic descriptions. First, we can define the matching criteria as follows:

Definition 1: A resourceSmatches a queryQ when all the response parameters of Q are matched by the response parameters of S, and all the request parameters of S are matched by the request parameters of Q.

Definition 1 guarantees that the resource found satisfies the needs of the query, and the query provides all the request parameters that the resource needs to operate correctly. Our discovery algorithm is shown in Algorithm 3. This algorithm adopts strategies that rapidly prune resources that are guaranteed not to match the query, thus improving the efficiency of the system. A query is matched against all resources stored in the repository. Whenever a match between a query and resources is found, it is recorded and sorted within the matches according to highest score. A match between a query and a resource consists of matching all the response parameters of the query against the response parameters of the resource; and all the request parameters of the resource against the request parameters of the query. If one of the query's responses is not matched by any of the resource's response, the match fails. Matching between requests is computed by the same process, but with the order of the query and resource reversed. The score of a match between two requests/responses is calculated by the semantic matching algorithm.

Algorithm 3: Composableresource discovery

```
1  record = Ø
2  for each S in repository
3    if Matching(Q, S) then append S to record
4  endfor
5  sort(record)
6
7  Matching(Q, S){
8    SemanticMatch(Q.response, S.response)
9    SemanticMatch(S.request, Q.request)
10 }
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013

V. EXPERIMENTAL EVALUATION

A. EXPERIMENT ENVIRONMENT

In our experiments we compared our semantic matching and resource discovery algorithms based on the ontology learning method with the traditional keyword searching method. Our objective is to show that we can achieve high recall and high precision performance in finding similar resources, and to investigate the contribution of the two novelalgorithms (i.e., semantic matching and composable resource discovery) of our method.To run our experiments, we extracted a collection of RESTful web services from the ProgrammableWeb site [10]. The site currently has about 6,000 RESTful web services, and the total number of services is rapidly growing. We first collected the subset which associated WADL files for three domains: weather, travel, and mapping. The ProgrammableWeb site allows us to manually extract information regarding their functionality description, request, and response. This set contains 168 RESTful web services and 264 requests/responses. There are a total of 501 terms. During our experiments, while manually evaluating the clustering results, we observed that a minimum support of 3% and a minimum confidence of 80% are reasonable threshold values.

We measured the overall performance using the *recall* (*R*), *precision* (*P*), and *F-measure* (*F*). The recall is a measure of completeness, whereas the precision is a measure of exactness or fidelity. An inverse relationship exists between recall and precision, in which it is possible to increase one only at the cost of reducing the other. Usually, the recall and precision measures are not discussed in isolation. Rather, both are combined into a single measure, such as the F-measure, which is the weighted harmonic mean of the recall and precision. The recall $R = A/(A + B)$, where A stands for the number of returned relevant operations, B stands for the number of missing relevant operations, and A+B stands for the total number of relevant operations. The precision $P = A/(A + C)$, where A stands for the number of returned relevant operations, C stands for the number of returned irrelevant operations, and A+C stands for the total number of returned operations. The F-measure $F = 2PR/(P + R)$.

B. RESULTS AND ANALYSIS

Figure1 presents the recall, precision, and F-measure values found via each method. Our experiments compared our *semantic matching algorithm* based on the ontology learning method, which we refer to as ONTO, with WORD and CLST. WORD and CLST denote the traditional keyword searching method and the ontology learning method without the pattern analysis technique, respectively. The reason for testing the semantic matching algorithm with or without the pattern analysis technique is to assess the impact of the ontological pruning process.

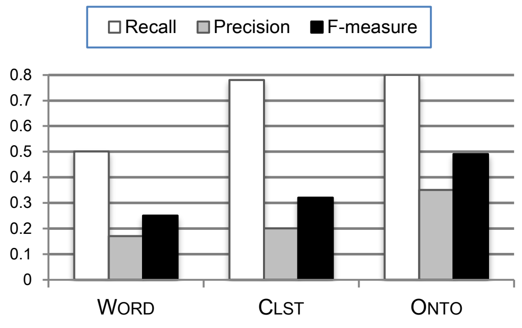


Figure1:The recall, precision, and F-measure values for different matching methods.

As shown in Figure1, our ONTO method beats all other methods. Its recall, precision, and F-measure are 80%, 35%, and 49%, respectively, much higher than those of the WORD and CLST methods. The CLST method significantly

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013

improves recall performance; this yields a recall 28% higher than that of WORD, but only improves the precision performance slightly. One major cause of this is that the clustering terms exert two-fold effects: On one hand, they provide additional evidence, which significantly assists in the number of returned relevant resources.On the other hand, they harbour noise that dilutes the high-quality evidence.

We conduct an additional experiment focusing on the performance of the pattern analysis technique. This experiment considers the efficiency of the ontological pruning. The ONTO resultsare shown inFigure1. The precision performance by the ontological pruning improves significantly; ONTOachieves a precision 15% higher than that of CLST. The ontological pruning makes it possible to search only relevant resources to ensure high performance. In conclusion, the ONTO method outperforms the WORD and CLST methods. Interestingly, CLST achieves a precision as high as WORD, and a recall as high as ONTO.

We plot the recall-precision curves (R-P curves) in Figure2. In the R-P curves, the X-axis represents the recall, and the Y-axis represents the precision. The curve closest to the upper right-hand corner of the graph indicates the best performance. The R-P curves are regarded by the IR community as the most informative graph demonstrating the efficacy of a search engine. From these curves, we can observe that ONTO and CLST methods exhibit better performance than the WORD method. In particular, our ontology learning method (i.e., ONTO) exhibits the best performance with its pattern analysis technique.

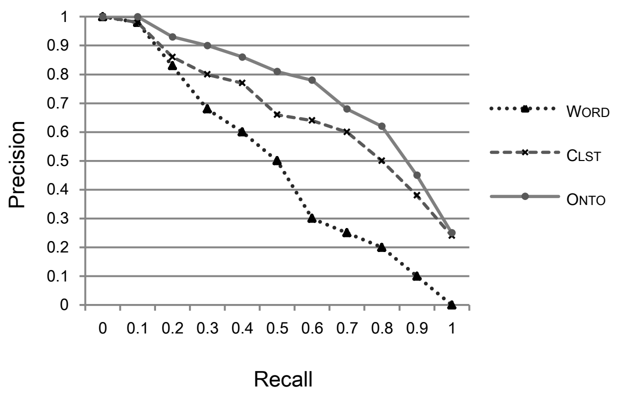


Figure2:The recall-precision curves for different matching methods.

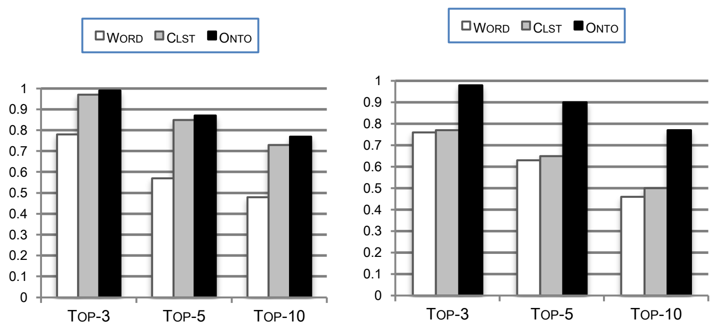
We also present the recall and precision values of top-3, top-5, and top-10 returned results found from the *composable resource discovery algorithm*. Figure3(a) illustrates the top-k recall comparison. To measure the recall, we are interested in the number of returned relevant resources that can be returned within a given set of relevant operations. We can see that both ONTO and CLST can achieve satisfying results. For top-3, top-5, and top-10 recalls, ONTO reaches 98%, 87%, and 77%, respectively, obviously higher than the other two methods. CLST is a little lower at the level of top-3, top-5, and top-10, as it measures the similarity based on the underlying semantics. WORD has very low recall, because it only matches the keywords from the user's requests, and the results are coarse.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 9, November 2013



(a) Top-k recall (b) Top-k precision

Figure3:The top-k recall and precision comparison for different resource discovery methods.

Figure3(b) presents the top-k precision comparison. As shown in this figure, ONTO beats the other two methods in each result, since it filters the irrelevant concepts. For top-3, top-5, and top-10 returned results, ONTOprecision is 98%, 90%, and 77%, respectively. ONTO can improve the precision 22%, 27%, and 31%, respectively, rather than those of WORD, 21%, 25%, and 27%, respectively, higher than those of CLST. The figure demonstrates that with the expected number of returned results increasing, the rate of performance improvement increases. Therefore, the ONTO method is superior to the WORD and CLST methods, because it can improve both recall and precision performance significantly.

VI. RELATED WORK

There are two approaches to semantic web services: *semantically enabled web services* and *semantic web technologies*. The semantically enabled web services can be either SOAP-based or RESTful web services with semantic wrappers that weave them into the semantic web. The semantic web technologies are not based on the mindset of traditional web services. They acknowledge the fact that the semantic web is for machines, and build upon it

A. SEMANTICALLY ENABLED WEB SERVICES

1) SOAP-based Semantic Web Services

SAWSDL [2] derived from WSDL-S is a lightweight solution, and is only a W3C semantic web service recommendation. It annotates WSDL components such as inputs/outputs with references to ontologies. There are more ambitious W3C submissions for semantic web services, including OWL-S [3], WSMO [4], and SWSF [5]. OWL-S is based on OWL, which describes the service in terms of profile, process, and grounding. WSMO is based on modeling web services using the 4 major elements of ontologies, web services, goals, and mediators. SWSF is one of the latest approaches to semantic web services and is built upon experience gained with OWL-S and WSMO. SOAP-based web services do not fit well with REST principles; SOAP has one endpoint and many actions on that endpoint, whereas REST provides an endpoint for each resource that will be acted upon.

2) RESTful Semantic Web Services

With RESTful web services gaining more popularity on the web, the interest in RESTful semantic web services is growing. SA-REST [11] is similar to SAWSDL, as it semantically annotates RESTful web services. As there are no WSDL files for RESTful web services, it adds the annotations to web pages that describe the services. The idea is to use Microformats such as GRDDL and RDFa to embed the annotation in HTML files. Another approach was introduced by Battle and Benson [12], wherein the Semantic Bridge for Web Services (SBWS) provides custom annotations to the WADL documents, which are similar to SAWSDL. A Shortcoming of these approaches is that these annotations are both difficult and costly.

B. SEMANTIC WEB TECHNOLOGIES

Another part of SBWS [12] involves providing a semantic REST concept. This mapped the commonly accepted HTTP methods (GET, PUT, POST, and DELETE) into extended SPARQL commands (SELECT, INSERT, MODIFY, and DELETE). The result provides the ability to retrieve, add, update, and remove RDF datasets directly from the endpoints already in use in a RESTful web service. Morbidoni et al. [13] developed a Semantic Web Pipes (SWP) for resolving RDF data and endpoints for SPARQL queries. Another approach that is based on semantic constructs is Triple Space Computing (TSC) [14]. It is based on Tuple Space Computing. The communication is shifted from being message oriented as in web services, to reading and writing RDF triples in a shared triple space. TSC has been used in both web service coordination and communication. A main drawback of these approaches is that the construction of RDF datasets is a relatively time-consuming task.

C. ONTOLOGY LEARNING METHODS

Currently, ontologies are developed manually through the collaboration of highly skilled domain experts and ontology engineers. Ontology building is therefore a time-consuming and labour-intensive task whose automation is desirable. A number of ontology learning methods have been proposed for the automatic acquisition of semantic information.Hess and Kushmerick [15] employ Naive Bayes and SVM machine learning methods to classify WSDL files in manually defined task hierarchies. Dong et al. [16] employ a clustering method that clusters parameters present in inputs and outputs of operations into parameter concepts. But these clusters are not semantic ontology and just simple a set of synonym words. Sabou et al. [17] propose an automatic extracting method that learns domain ontologies from textual documentation attached to web services. The limitation of this method is that it is confined to human-readable documentation, which is not common means of web service descriptions. Guo et al. [18] leverage relationships between words in phrases to establish ontological relationships between acquired concepts. But the authors tackle only a one-to-one mapping solution by aligning the generated ontology fragments, and taking advantage of active domain experts.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents a semantic matching algorithm and acomposasble resource discovery algorithm based on the ontology learning method to improve the recall and precision performance of RESTful web services. These algorithms can get optimal results by applying strategies that rapidly prune resources that are guaranteed not to match the query.Theproposed algorithms are crucial features in the use of RESTful semantic web services.We implemented our algorithms using the publicly available open tools (e.g.,CRFTagger, Apriori, and ClusterLib) and experimented on a collection of RESTful web services. The experimental results demonstrate that our algorithms significantly improve the recall and precision performance relative to the traditional keyword searching method. The results show that our method achieves up to 30% improvement for recall performance, and up to 18% for precision performance compared to the keyword searching method.

We are currently developinga system for automatically composingRESTful web services. More specifically, we are working in the exciting area of semanticmashups. Our research will focus on specifying RESTful web services, finding them, and integrating them to create mashups. Currentmashups require a human who has the domain knowledge and can guide the overall composition process. The incorporation of our semantic matching and resource discovery algorithms would result in further automation of the mashups. As components of future work, various optimization techniques can be applied to the algorithms.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (No. 2010-0008303).

REFERENCES

[1] Fielding R.,"Architectural styles and the design of network-based software architectures", PhD thesis, University of California, 2000.
[2] Kopecky J., Vitvar T., Bournez C., and Farrell J.,"SAWSDL: Semantic annotations for WSDL and XML schema", IEEE Internet Computing,Vol.11, No.6, pp.60-67,2007.
[3] OWL Services Coalition),"OWL-S: Semantic markup for web services", OWL-S White Paper, 2004.
[4] Vitvar T., Zaremba M., Moran M., Zaremba M., and Fensel D.,"SESA: Emerging technology for service-centric environment", IEEE Software,Vol.24,No.6, pp.56-67,2007.
[5] http://www.w3.org/Submission/SWSF/.
[6] Lee Y.J. and Kim C.S.,"Building semantic ontologies for RESTfulweb services", Proceedings of the 6th International Conference on Next Generation Web Services Practices, pp.37-40,2010.
[7] Hadley M.,"Web application description language (WADL)", http://www.w3c.org/Submission/wadl/.
[8] Salton G. and Buckley C.,"Term weighting approaches in automatic text retrieval", Information Processing and Management,Vol.24, No.4, pp.513-523,1988.
[9] Paolucci M., Kawamura T., Payne T., et al., "Semantic matching of web services capabilities",Proceedings of the 1stInternational Semantic Web Conference,pp.333-347, 2002.
[10] http://www.programmableweb.com/.
[11] Sheth A., Gomadam K., and Lathem J.,"SA-REST: Semantically interoperable and easier-to-use services and mashups", IEEE Internet Computing,Vol.11, No.6, pp.91-94,2007.
[12] Battle R. and Benson E.,"Bridging the semantic web and Web 2.0 with representational state transfer (REST)", Journal of Web Semantics,No.6,pp.61-69, 2008.
[13] Morbidoni C., Phuoc D., Polleres A., Samwald M., and Tummarello G.,"Previewing semantic web pipes", Proceedings of the 5th European Semantic Web Conference on the Semantic web: Research and Applications, pp.843-848,2008.
[14] Riemer J., Martin-Recuerda F., Ding Y., et al.,"Triple space computing: Adding semantics to space-based computing", The Semantic Web – ASWC 2006, Lecture Notes in Computer Science, 4185, pp.300-306,2006.
[15] Hess A. and Kushmerick N.,"Learning to attach metadata to web services",Proceedings of the 2nd International Semantic Web Conference, pp.258-273,2003.
[16] Dong X., Halevy A., Madhavan J., Nemes E., and Zhang J.,"Similarity search for web services", Proceedings of the 30th VLDB conference, pp.372-383,2004.
[17] Sabou M., Wroe C., Goble C., and Stuckenschmidt H.,"Learning domain ontologies for semantic web service descriptions", Journal of Web Semantics,Vol.3,No.4, pp.340-465,2005.
[18] Guo H., Ivan A., Akkiraju R., and Goodwin R.,"Learning ontologies to improve the quality of automatic web service matching",Proceedings of IEEE International Conference on Web Services, pp.118-125,2007.

BIOGRAPHY

Dr. Yongju Lee
Hereceived the Ph.D. degree in Information and Communication Engineering, KAIST (Korea Advanced Institute of Science and Technology) in 1997, Korea. He is now a Professor in School of Computer Information, Kyungpook National University, Sangju, Korea. He has published more than 60 papers in domestic and international conferences and journals. His current research interests include web services, cloud computing, semantic web technology, and mobile applications. Dr. Lee may be reached at yongju@knu.ac.kr.