Subsequence-Based Dynamic Web Service Composition

Ping Liang and Sartra Wongthanavasu

***Abstract*—Semantic web services are broadly applied in intelligent heterogeneous web. Particularly with the development of the Internet of Things (IoT), countless internal and external data are available which need web services to get bundled and processed. For many proprietary organizations, semantic service ontology or Universal Description, Discovery, and Integration (UDDI) for Web Services Description Language (WSDL) is designed to orchestrate services. Annotation has been used as rules to create semantics ontology or descriptive logic of execution schema for intelligent learning. It is not flexible enough due to the reason that the current annotation or BPEL-based process definitions are based on context-related domain knowledge or QoS requirements. A more flexible method for dynamic composition is needed to address user' arbitrary requests. This paper is to propose a context-free sequence-based method to automate the composition of semantic web services. Its basic approach is to collect bags of services which logic relation is defined by the keywords of user query. Then the common services are found and we use the subsequence algorithm to find a shortest path from the connected services of two bags to fulfill the user query. The found sequence of web services can be automatically executed by the framework of Spring+RESTFul. The dynamic web service composition is completed in an unsupervised way.**

***Index Terms*—Subsequence, dynamic composition, composition automation, semantic web service.**

I. Introduction

Semantic web service is one of the solutions, which are to annotate and offer semantics for web services to achieve the interoperability between web services. The semantics is defined in formal logic, graph theoretic methods or BPEL-based process [1], etc., to meet the requirements of the dynamic and changing queries requested by users. For supervised and unsupervised learning, the automation is either too complex or requires human intervene [2], particularly in the area of service composition [3]-[5].

So this paper presents a new approach based on subsequence to address the problem without relying on BPEL. The rest of the paper is given in 6 sections. The Section II describes the related works being done in the area and the Section III is to detail the algorithm described in the Section II. Then an implementation of eBay framework based on Spring+RESTFul is given in the Section IV to use eBay web services' WSDL to integrate RESTful to complete the semantically composed processing in the Section V. Finally, a conclusion in the Section VI is to summarize and explain the issue addressed in the paper and future works to be done.

Manuscript received July 1, 2016; revised August 12, 2016. This work was supported in part by Computer Science Department of Khon Kaen University, Thailand.

The authors are with the Computer Science Department of Khon Kaen University, Khon Kaen, 40000, Thailand (e-mail: liang_p@hotmail.com, wongsar@kku.ac.th).

II. Related Works and Proposed Method

Service orchestration has many methods for composition, including model checking, semantics ontology and entropy-based clustering, QoS-driven systems, etc., to categorize web services, validate the functionality and property of web services and then complete the composition [3], [4], [6], [7]. These methods treat the service composition as workflow or business processes. The pre-defined business logic can be virtually automated through search paths or business processes [8]. So the services are composed along with the user's selection and direction, which means human intervene, normally in advance.

This paper proposes a method for service orchestration and takes the approach of combining the bag of set theory with sub-sequence algorithm to automate the composition of services for user query, which is unsupervised learning [9]. The proposal is applicable in a well-documented domain, which means the services are well-defined by their interface--input and output. Both input and output are defined in XML schema within WSDL description

We are looking for a computable way to fulfill user query with more intelligence and hopefully less human intervene. This paper intends to tackle the issue by the algorithm as below,

1) Based on user query, a user sequence of web services is composed by the keywords as the input *kin* or output *kout* and their logic relationship formed.
2) We first use MapReduce [10] to map eBay web services to the pairs of key and value as (*k1*<*ConnectedWebService*>,*v1*:*keyword*<*String*>). The key is the connection of two web services. It means that the second service has its input matched up with or contained in the output of the first service. The value is the user keyword. So each keyword has its bag of the connected web services.
3) Then we need to reduce, we look into the (*k1*,*v1*) created in the map function. The value *v1* is a user keyword. Now we allow the keywords to be the classifier to find all the shortest paths from the input keywords *kin* to the output keyword *kout*.

The next step is to look for the first common web service between the paths. If the common web service requires the input of two paths, the two paths join. Otherwise, the longer one is discarded. If there is a subsequence that matches up with the user sequence, then a possible service composition is achieved.

doi: 10.18178/ijfcc.2016.5.4.466

III. Implementation

A. Subsequence Generation

Using the eBay web services relationship, the algorithm is as below. Mapreduce is used because Mapreduce can process online data fast and also is able to be integrated with Spring+RESTFul framework. Based on eBay web service WSDL (http://developer.ebay.com/webservices/latest/ebaysvc.wsdl), we need to write the key and value for web service sequence generation for map and reduce functions. As given in Algorithm 1, the map function is to start with user query keywords and eBay web services. To prepare for the input of eBay web services, we modify the apireferencedocs (http://developer.ebay.com/community/codebase/apireferencedocs/default.aspx) for Trading API to generate eBay web services together. For simplicity, we limit to the fields defined as "Required". They have the annotation of "RequiredInput" in each "CallInfo" in eBaySvc.wsdl and are easy to be identified. So we can get the inputs of the web service list for the map function.

Starting from the input kin, the connection between web services is established by finding out if there are any output fields of one web service equal to or contained in another one's input fields as in Algorithm 2. If the connection exists, the pair of web services is recorded and the keyword *kin* is assigned to this specific connection. The result of the map function shall have the bags of the sequences of the connected web services. The number of the bags as labelled by the value *v1* is the same as the number of the user query keywords.

Algorithm 1: Mapping eBayWebService *W* to connected eBayWebServices *R*

```
Input:
S = UserKeywords(String[]);
W = ListofeBayWS(eBayWS,
<InputList,OutputList>);
Output: R = BagsofMappedeBayWS(Collection,
UserKeyWord)


Collection c= new Collection(<firstWS,
secondWS>,equalInOut<String>)
WHILE {(i<S.size())}
keyword = s_i;
eBayWebService[] E = new eBayWebService[];
   WHILE {(j<W.size())}
      IF {s_i is contained in w_j.InputList}
         E.add(w_j);
ENDIF
   ENDWHILE
   j=0;
   i=1;
   findBagofSequence(i,s_i,E,W,r_i);
context.write(r_i, s_i);
 ENDWHILE
```

The Algorithm 3 is the reduce function to find all the shortest paths of subsequences from the input keyword to the output. The paths must satisfy the logic relation of user queries. As described in Section 3, the shortest paths from the input *kin* to the output *kout* are generated by the map function. Then the paths are reduced either by joining or discarding depending on the logic relationship of the *kin*.

B. Service Composition

After using Mapreduce to quickly generate the shortest sequences of the connected web serivce which also can satisfy the logic relationship, we can apply the framework of Spring+RESTFul. The MapReduce integrates web service and the RESTFul is for user visualization. We use eBay Java SDK for its Trading API to implement the sequence of the web services. The linkage is realized by a chain of web service calls, same as a chain of program sub-procedures. In order to focus more on the idea of this paper, only GET among RESTFul is implemented. The issue is that the web services and their input and output are dynamic. So we keep W which is a list of eBay Web Services - eBayWS, <InputList,OutputList> and Collection of the connected web services in the form of Collection(<firstService, secondService>,equalInOut<String>). The output that associates two web services must be ensured to be passed correctly from the first service to the next one till the result is achieved. The algorithm is given in Algorithm 4.

Algorithm 2: findBagsofSequence(i,s, E, W, R): Find the bags of web services for each keyword

```
 WHILE {(i<E.size())}
   Collection c= new Collection(<firstWS,
 secondWS>,equalInOut<String>)
   eBayWebService[] EF = new
 eBayWebService[];
   WHILE {j<W.size()}
      IF {e_i.Output is contained in w_j}
         EF.add(w_j);
         c.add((e_i,w_j),e_i.Output;
      ENDIF
   ENDWHILE
   j=0;
   R.add(c, s); \\Recursion to find all the paths to generate
R;
   findBagsofSequence(i, s, ER, W, R);
 ENDWHILE
```

Algorithm 3: Reduce function

```
Input:
A = WSsequence(<sequence>[],keyword<String>);
SequenceArray<eBayWS> sequence=eBayWS()[];
kin = keyword for input
kout = keyword for output
WHILE{i<R.size()}
   WHILE{j<r_i.size()
     sequence_i=findShortestPth(r_i,kin,kout);
   ENDWHILE
   j=0;
   A.add(sequence_i, kin);
ENDWHILE
WHILE{l<A.size()}
   as1 = a_l.sequence_l;
```

```
    as2 = a_l.sequence_(l+1);
    commonService =
  findFirstCommonService(as1,as2);
    ind1 = as1.commonService.index;
    ind2 = as2.commonService.index;
    IF{as1 _(ind1-1).output != as2_(ind2-1).output
      tempSequence = getSubsequence(as2,0,ind2);
      a_l.insert(tempSequence, ind1);
    ELSE
     minL=
 finShortestPath(a_l.sequence_l,a_l.sequence_(l+1))
      a_l.delete(a_l.minL);
    ENDIF
 ENDWHILE
```

Algorithm 4: Automatic composition of connected web services ServiceComposition(A,W)

```
 Object wsName, wsIn, wsOut;
 wsIn = keyword for input;
 WHILE{i<A.size()}
    Class<?> wsName = Class.forName(a_i.firstWS);
    IF{wsName.OutputList != empty}
      wsOut = wsName(wsIn);
    ELSE
      exit();
    ENDIF
    wsName = Class.forName(a_i.secondWS);
    IF{wsOut == a_i.secondWS.OutputList}
      wsIn = wsOut;
    ELSE
      IF{wsOut contains
 a_i.secondWS.OutputList}
        fpath =
 getSubfieldPath(a_i.secondWS.OutputList);
        wsIn = wsOut.fpath;
      ENDIF
    ENDIF
 ENDWHILE \\Note: A is the final composition of web
services
```

C. Spring + RESTFul Visualization

The sequence above is placed in Spring MVC + RESTFul framework. A RESTFul structure is applied to get user request and then display the response of the result generated by the web service sequences.

```
 @Path("generic")
 public class GenericResource {
    SequenceComposition A;
    eBayWS W;

    @GET
    @Produces("text/html")
    public String getHtml(
      @QueryParam("Keyword") String Keyword,
      @QueryParam("Output") String Output) {
        return "<html><body><form
 method=\"post\">
          Keyword:
          <input id=\"Keyword\"
 name=\"Keyword\"/>  "
          + "Output: <input id=\"Output\" \
name=\"Output\"/>
          <input type=\"submit\"/> " +
 "</form></body></html>";
    }

    @POST
    @Produces("text/html")
    public String postHtml( @FormParam("Keyword")
String Keyword ,@FormParam("Output") String Output){
      String finalResult = Keyword + "   "+ Output;

      //start hadoop mapreduce
      Job job = new Job(new Configuration());
      job.setJarByClass(WSJob.class);
      .....
      job.waitForCompletion(true);

      //eBay services composition
      Class<?> result = Class.forName(
        (ServiceComposition(A,W)));
        Field[] resultF = result.getFields();
      while(i<resultF.size()){
        finalResult += resultF_i;
      }
      return "<html><body>"+finalResult+
              ="</body></html>";
    }
 }
```

The complexity is composed by the map and reduce function plus the service composition. The MapReduce complexity includes the single-pair shortest path complexity, recursion for reducing and the first common node among sequences. The service composition is to search through eBay web service structure to match up the input and output of each web service. Based on that, the sequence of web services can be called one after another automatically with matched-up I/O. So we have the total complexity of our method as below,

$$\frac{1}{2}O(n*m+n^2\log n)+O(n*\log n) \qquad (1)$$

The single-pair shortest path complexity should be $\leqslant 1/2$ of all-pair shortest path [11] because web services are scarce and loose. This is due to the requirement of its design and also to avoid a Non-deterministic Polynomial (NP)-hard problem [7], [12];

The subsequence algorithm is not NP-hard problem as long as there is no arbitrary number of inputs [13], [14]. And we only need to compare the user query subsequence to the available paths found one by one. So, for any two subsequences, we have $O(\mathrm{N}\prod_{i=1}^{N} n_i)$, N is the number of the subsequences.

If there are m paths found, then we have

$m \times O(N \prod_{i=1}^{N} n_i)$, M is the number of the paths of the common services.

IV. Comparison with Other Methods

As for the comparison to other methods, in the overview of existing approaches in [1] it clearly indicates the advantages and disadvantages of various approaches to tackle the problem. This paper uses several open source BPEL-based tools for service composition. The comparison with other methods is given in Table I as below.

TABLE I: Comparison with Existing Methods

Method & Development	Pre-condition	Dynamic Composition
Java Web Servlet	BPEL-supported process definition in XML which is a sequence of business tasks	No
Apache ODE & Java Axis 2, JBI, SMIX OSGi	BPEL-supported definition in XML, WSDL, XSD which is a sequence of business processes;	No
Orchestra & Java Web Servlet with Tomcat, JOnAS, OSGi Felix	Process definition in BPEL,XPDL which is a sequence of business processes;	No
Method in this paper	Java Web Servlet with Spring+RESTFul User query keywords	Yes

The method in this paper needs to read the static eBay WSDL to prepare the web service description for the establishment of the automatic composed sequence. It does not pre-define any business processes as other methods do. This becomes its advantage. The method in this paper only implements the GET. The other RESTFull operations require more user interaction, that is, further inputs. This may cause the problems of sequence interruption or unexpected result.

V. Conclusion

This context-free composition of services is to ignore the context-related knowledge and only follow the match-up of keywords to simplify the search. It is fully automatic and computable. However, the realization of the algorithm depends on the standardization and normalization of the annotation of service interface, like WSDL. In order to use the method in wider range, e.g. the IoT, the fuzzy methods must be introduced to clear up the ambiguity to correctly gather a bag of connected services. Future works can consider user preference and QoS in forming the sequence for dynamic service composition.

Acknowledgment

Authors thank the support from Computer Science Department, Khon Kaen University, Thailand.

References

[1] P. Bartalos and M. Bielikov, "Automatic dynamic web service composition: A survey and problem formalization," *Computing and Informatics*, vol. 30, pp. 793–827, 2011.
[2] F. Lécué, E. Silva, and L. F. Pires, "A framework for dynamic web services composition," *Emerging Web Services Technology*, vol. II, Springer, 2008, pp. 59-75.
[3] H. Q. Yu and S. R. Marganiec. (2004). Semantic web services composition via planning as model checking. [Online]. Available: http://www.cs.le.ac.uk/people/hqy1/swsc_pamc1.0.pdf
[4] S. Dietze, N. Benn, J. Domingue, A. Conconi, and F. Cattaneo, "Two-fold service matchmaking–Applying ontology mapping for semantic web service discovery, the semantic web," *Lecture Notes* in *Computer Science*, vol. 5926, pp. 246-260, 2009.
[5] P. Shvaiko and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Transaction on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 158-176, 2013.
[6] E. M. Clarke, Jr. O. Grumberg, and D. A. Peled, *Model Checking*, US: MIT Press, 1999.
[7] (2005). *Semantic Web Services Ontology (SWSO)*. [Online]. Available: http://www.w3.org/Submission/SWSF-SWSO/
[8] I. D. Pietro, F. Pagliarecci, L. Spalazzi, A. Marconi, and M. Pistore, "Semantic web service selection at the process-level: The eBay/Amazon/PayPal case study," in *Proc. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.
[9] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *J. ACM*, vol. 24, no. 4, pp. 664–675, 1977.
[10] Y. Mao, R. Morris, and Kaashoek, "Optimizing mapreduce for multicore architectures," *Tech. Rep. MIT-CSAIL-TR-2010-020*, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2010.
[11] C. Cooper, A. Frieze, K. Mehlhorn, and V. Priebe, "Average-case complexity of shortest-paths problems in the vertex-potential model," *Lecture Notes in Computer Science*, vol. 1269, pp. 15-30, 1997.
[12] F. Harary, *Graph Theory*, Reading, MA: Addison-Wesley, 1994.
[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., US: MIT Press and McGraw-Hill, 2001, pp. 350–355.
[14] D. Maier, "The complexity of some problems on subsequences and supersequences," *J. ACM (ACM Press)*, vol. 25, no. 2, pp. 322–336, 1978.

Ping Liang received the M.Sc. degree in software engineering from University of York, UK, in 2002 and is now a PhD student in computer science in Khon Kaen Univeristy, Thailand. Her major interests are in matching learning and artificial intelligence. She is also a lecturer of Southwest University for Nationalities in Chengdu, China. Her research interests include text mining, information fusion, and knowledge discovery.

Sartra Wongthanavasu received the B.Sc. degree in mathematics from Khon Kaen University, Thailand, in 1982, the M.S. degree in operations research from National Institute of Development Administration (NIDA), Thailand, in 1985 and M.S. in computer science from Illinois Institute of Technology (IIT), U.S.A., in 1996. He also received the Ph.D. degree in computer science from Asian Institute of Technology (AIT), Thailand, in 2001. Currently, he is an associate professor in the Department of Computer Science, Faculty of Science, Khon Kaen University, Thailand. His research interests include machine learning, computer vision, cellular automata, and knowledge engineering.