

Ranking-Based Suggestion Algorithms for Semantic Web Service Composition

Rui Wang, Sumedha Ganjoo, John A. Miller and Eileen T. Kraemer
University of Georgia, Athens, GA, 30602

{wang@cs., sganjoo@, jam@cs., eileen@cs.}@uga.edu

Abstract

The process of selecting Web services from a large number of potential services available on the Web is a challenging task for users engaged in Web service composition. This work is devoted to resolving this issue by suggesting Web services to the user. Our suggestion algorithm ranks all the available services for the user based on the semantic annotations of a service's inputs, outputs and functionality, as well as pre-conditions and effects, if available. This paper presents multiple algorithms for making suggestions during Web service composition. These algorithms extend traditional Web service discovery algorithms; in particular, they include new techniques for ranking the effectiveness of data mediation.

Keywords: Ranking-based suggestion, Path ranking, Web service composition, Data mediation, Similarity measures, SAWSDL/WSDL-S

1. Introduction

Web service composition techniques provide means for combining Web services to form workflows or processes to perform complex tasks. Many researchers [1, 2] have worked on aiding users in composing such processes. Our previous work [3, 4] focused on making it easier for less sophisticated users to compose a process using existing Web services.

Given a design canvas containing a partially completed workflow design, plugging in the next Web service and connecting it to the existing services can become quite difficult. Users need help not only in resolving the heterogeneities of messages exchanged between Web services, but also in selecting suitable services from among the large number of available candidate services on the Web.

In this paper, we present algorithms for suggesting the most suitable services to aid users composing Web services. Our basic algorithm makes the suggestion under the assumption that there is one particular service that needs to be plugged in. Upon a user's request, our suggestion algorithm makes recommendations about which services to insert into a partially completed workflow design, between the relevant workflow prefix

and suffix. An extension to this algorithm allows a chain of services to be plugged into the current process when one service does not suffice. To support the suggestion algorithms, a few supporting functions were also either developed or upgraded, including those for data mediation and ontological concept similarity.

Our suggestion algorithms calculate the ranking scores based on data mediation, service functionality and formal specification. Both functionality and data mediation algorithms require a similarity measure algorithm to compare ontological concepts specified using, for example, the Web Ontology Language (OWL) and annotated using the Semantic Annotations for WSDL (SAWSDL) [5, 6] standard. This similarity measure algorithm generates a score by semantically comparing two ontological concepts. We chose to use the W3C SAWSDL standard for semantic Web service, because it provides a simple and effective way to add semantics to existing Web Service Description Language (WSDL) files.

The rest of this paper is organized as follows. In Section 2, the data mediation algorithm is discussed. Section 3 presents the service suggestion algorithms. Similarity measures, adopted from our prior work on discovery, are briefly described in Section 4. Section 5 covers the related work on assisting user composing Web services. The conclusions and future work are given in Section 6.

2. Data mediation

As discussed by several researchers, many compositions fail due to services having differing syntax and data structures (or more generally data heterogeneities) [7]. There have been some efforts [8] to reduce these failures by utilizing data mediation. We argue in this paper that data mediation should certainly be taken into account in ranking candidate services.

We implemented a data mediation algorithm that is called the path ranking-based bi-directional data mediation algorithm. Bi-directional means that it combines our previous work on bottom-up [3] and top-down [7, 9] data mediation. Path ranking is included, because we rank every path in the input/output messages to find the best match between messages according to their semantic annotations. Another feature of our data mediation algorithm is that we consider the output

messages of not only the directly preceding service operation, but also the indirectly preceding service operations, as well as the global input.

2.1. Data path ranking

As an essential part of our data mediation algorithm, the PATH-RANK function finds the best matching output path for one path of the input message. We use a tree to represent an input/output message of a Web service's operation. For example, Figure 1 shows the tree representing the output message of the getOrtholog operation of the ortholog Web service. The root of the tree is the <message> node in the SAWSDL file, which is the node getOrthologResponse. The leaves of the tree, out1 and out2, are called bottom nodes. A data path is defined as an array of nodes from any bottom node to the root node. In Figure 1, there are two paths: {N₄, N₃, N₂, N₁} and {N₅, N₃, N₂, N₁}.

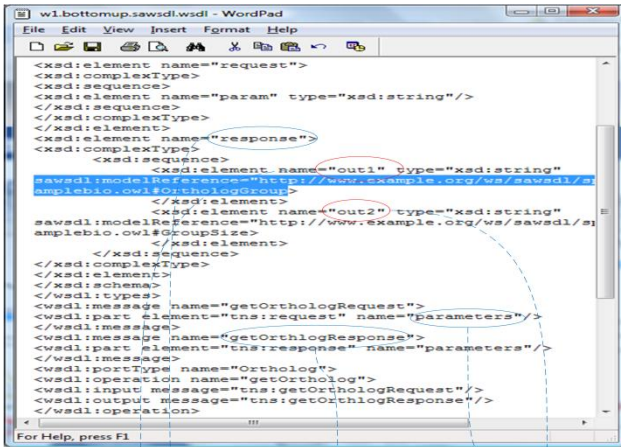


Figure 1. A tree representing the output message of a Web service

The pseudo-code for the PATH-RANK function is shown in Figure 2. It is used to rank all the paths of all outputs of all operations in current process to match a path in the input message of the candidate operation. By applying the PATH-RANK function to all the paths of the input message, the mappings from the output messages to the input message are established. Note, for a tree with n nodes, the number of full paths is less than n . Input/output message mappings between multiple Web services are based on the semantic annotations in the SAWSDL files.

These mappings will then be used to facilitate adding the new service to the process. For example, in a Business Process Execution Language (BPEL) process, these matched paths can be easily turned into XML Path Language (XPath) expressions and used in the <assign> elements of the BPEL process.

```

PATH-RANK ({P1, P2, ..., Pn}, P0)
// P0 is one path on the input message of the candidate
// operation
// {P1, P2, ..., Pn} is a set of existing paths that will be
// compared to P0
for  $i$  in {1, 2, ..., n} do
    S $i$  = COMPARE-2-PATHS (P $i$ , P0)
end
 $k$  = arg-max{S1, S2, ..., Sn}
return < S $k$ , P $k$  >
// S $k$  is the matching score between P $k$  and P0, P $k$  is the
// best matching path to P0
    
```

Figure 2. PATH-RANK function

```

COMPARE-2-PATHS (P $i$ , P0)
// P $i$  and P0 are the two paths to be compared
{A1, A2, ..., A $j$ , ..., A $m$ } = semantic annotations of all
// nodes of P0
{W1, W2, ..., W $m$ } = weights of all nodes of P0
{A'1, A'2, ..., A' $j$ , ..., A' $z$ } = semantic annotations of all
// nodes of P $i$ 
L = min {m, z}
return  $\sum_{j=1}^L$  CS(A' $j$ , A $j$ ) * W $j$ 
// CS(A' $j$ , A $j$ ) is the ontological concept similarity score
// of A $j$  and A' $j$  (see section 4)
    
```

Figure 3. COMPARE-2-PATHS function

The PATH-RANK function works by iteratively invoking the COMPARE-2-PATHS function. The COMPARE-2-PATHS function calculates the matching score between two given paths based on their semantic annotations. For example, we can compare the input path P₀ to output path P₂ (shown in Figure 4) using the COMPARE-2-PATHS function. The function starts from the bottom nodes of the two paths. It invokes the ontological Concept Similarity (CS) algorithm (see section 4) to calculate the similarity score of the two ontological concepts annotating the messages in the WSDL file. A weight is assigned to each node on the path of the input message of the candidate operation. The weights increase going down the tree, i.e., $W_{i+1} = a * W_i$, $\sum_{i=1}^m W_i = 1$, m is the height of the tree, (e.g., $a = 0.5$, $m = 2$, then $W_1 = 0.67$, $W_2 = 0.33$), but can be trained by machine learning algorithms. The overall matching score for the two paths is then calculated as the weighted sum of the concept similarity scores for each pair of nodes along the two paths.

Figure 4 shows an example of how the path ranking function finds the best matching path from an output message for path P_0 of the input message. Thus, the input message of a candidate service can be fed by the output message of the best matching operation in the current process. Suppose an operation's output message has four paths: $P_1 = \{N_7, N_3, N_1\}$, $P_2 = \{N_6, N_3, N_1\}$, $P_3 = \{N_5, N_2, N_1\}$ and $P_4 = \{N_4, N_2, N_1\}$, while the candidate operation's input message has two paths: $P_0 = \{N_0, N_8\}$ and the unlabeled path $\{N_9, N_8\}$. The highest score between path P_0 and every path of the output message is selected to be the final score. Therefore, the best matching path for path P_0 is path P_2 and the matching score is 1. The matching score can vary from 0 to 1. Since it is the matching score for one path, after calculating the scores for all paths of the input, the weighted sum of them will be the data mediation score of the candidate operation that is one component of the ranking score for the candidate operation (see Section 3 for details).

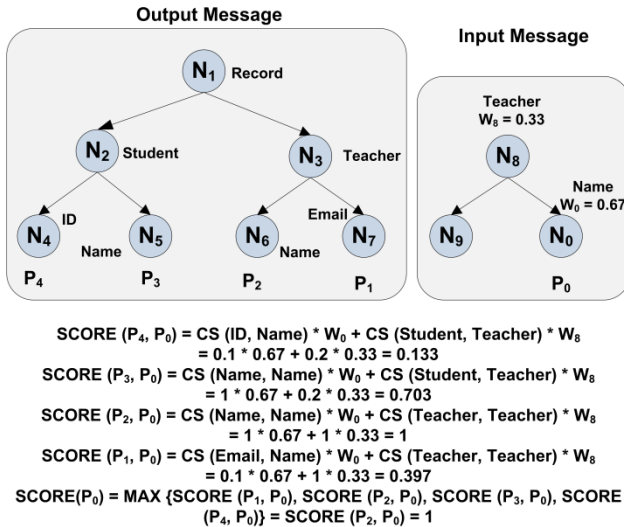


Figure 4. An example of path ranking

2.2. Bi-directional approach

This paper presents a data mediation approach that includes the features of our two previous data mediation approaches, the top-down [7, 9] and bottom-up [3] approaches. Thus, our approach is called a bi-directional approach.

The top-down approach mainly utilizes the liftingSchemaMapping and loweringSchemaMapping annotations on the root node to map two messages in a top-down manner. The liftingSchemaMapping annotation is used to map XML schema to ontology, while the loweringSchemaMapping annotation is used to map ontology to XML schema. Our bi-directional data mediation approach can handle these annotations as well. However, our approach does not force one to provide these types of annotations on the root node. If any node in

a message has one of these types of annotations, our pre-processing program will process the transformation specification (e.g., EXtensible Stylesheet Language Transformations (XSLT) file) indicated by the annotations. The mappings to the ontological concepts from the annotated node and all the nodes under the annotated node are then returned. These concepts can then be used in the path ranking function. Therefore, our approach lowers the requirements for the semantic annotations of Web services and increases the usability.

The bottom-up approach compares the modelReference annotations of the bottom nodes and then tracks the path in a bottom-up manner. Our approach retrieves the paths too; moreover, our approach takes into consideration the annotations of all the nodes on a path. This will result in a more accurate matching, since the nodes on the path will also help in resolving the heterogeneities.

2.3. Consider indirect preceding operations

Another unique feature of our algorithm is that it considers not only the directly preceding operations but also the indirectly preceding operations of the newly added operation. The pseudo-code for our DATA-MEDIATION algorithm is given in Figure 5. It first finds all the output messages of the preceding operations as well as the global input message to the overall process. Next, all the paths of these messages are compared with every path of the input message of the new operation. It invokes the PATH-RANK function iteratively to find the best matches to the paths of the input message of the candidate operation.

```

DATA-MEDIATION (CP, Ix)
// CP is the current process
// Ix is the input message of the operation to be added
// PATHS (s) is a function to retrieve all paths of
// messages inside set s
{O1, O2, ..., Om} = OUTPUT-MESSAGES (CP)
I0 = GLOBAL-INPUT-MESSAGE (CP)
{P1, P2, ..., Pn} = PATHS ({I0} ∪ {O1, O2, ..., Om})
{Px1, Px2, ..., Pxt} = PATHS ({Ix})
for Pxi in {Px1, Px2, ..., Pxt} do
    <Pi'i, Si'> = PATH-RANK ({P1, P2, ..., Pn}, Pxi)
end
return <{P1'1, P2'2, ..., Pt't}, {S1, S2, ..., St}>
// best matching paths to Ix are {P1'1, P2'2, ..., Pt't} and
// their related scores are {S1, S2, ..., St}

```

Figure 5. DATA-MEDIATION algorithm

Figure 6 shows a current process consisting of two Web service operations, which are connected as follows: $OP_1 \rightarrow OP_2$. OP_3 is the candidate operation to be added. OP_3 may take inputs from OP_1 , OP_2 and the global inputs.

Our data mediation algorithm is able to handle this case by considering outputs of all preceding operations and the global inputs. Other data mediation solutions, such as our previous top-down and bottom-up approaches are not able to deal with this situation.

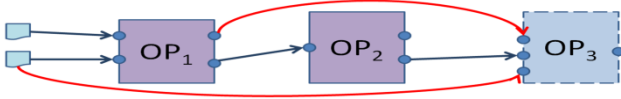


Figure 6. OP3 takes inputs from not only OP2 but also OP1 and global inputs.

3. Algorithms for suggesting services

3.1. Basic service suggestion algorithm

Our basic service suggestion algorithm ranks available Web services and suggests several top ranked services to the users. The ranking score is calculated based on the following three aspects: data mediation, service functionality and formal specification (pre-condition and effects). Users can ask the system to make suggestions as to which services to connect after, before or in the middle of the current process. They are referred to as forward, backward and bi-directional suggestions.

3.1.1 Forward suggestions

As shown in Figure 7, the forward suggestion is to recommend a Web service operation (OP_x) to be placed after the current process (CP). The forward suggestion ranking score (S) is calculated using the formula below, where W_{dm} , W_{fn} and W_{pe} are the weights for data mediation, functionality and formal specifications, respectively. Initially, $W_{dm} = W_{fn} = W_{pe} = 1/3$, but can be trained by machine learning algorithms.

$$S = W_{dm} * S_{dm} + W_{fn} * S_{fn} + W_{pe} * S_{pe}$$

where $W_{dm} + W_{fn} + W_{pe} = 1$



Figure 7. Forward suggestion

The S_{fn} score is calculated based on the functionality of the Web services. In particular, for each operation of a Web service, its functionality is specified using an ontological concept C_f . This concept is found using the modelReference provided with the `<operation>` tag in the SAWSDL file. If the user provides a desired functionality C_d for the operation to be added, it will be compared to the functionality annotation C_f for each candidate operation. The comparison uses the algorithm presented in Section 4 to calculate the functionality ranking score, i.e., $S_{fn} = CS(C_f, C_d)$. For example, $CS(\text{MultipleSequenceAlignment}, \text{SequenceAlignment})$

would score highly. If the user does not provide a desired functionality, the S_{fn} score will be set to zero.

The S_{dm} score is calculated based on the data mediation algorithm. A higher score means the input message of the candidate operation receives better matches. The pseudo-code for calculating the FORWARD-DATA-MEDIATION-SCORE is shown in Figure 8. It utilizes the DATA-MEDIATION algorithm to retrieve the scores for all the paths of the input message of the candidate operation. The weighted sum of the scores for all paths will give the S_{dm} score. We set the weight of each path to $1/(\text{number of paths})$, which indicates the percentage that each path contributes to the whole message.

In comparing the similarity of the annotations of the input and the output messages, the data mediation score is analogous to a Web service discovery score [10] that is used to rank services when discovering Web services. However, a typical discovery algorithm [10] only compares the annotations of the message node, e.g., N_1 and N_8 in Figure 4. Our data mediation algorithm traverses through the whole tree of an input/output message, which provides richer and more complete information for the input/output. This results in a more accurate matching score.

```

FORWARD-DATA-MEDIATION-SCORE (CP,  $I_x$ )
// CP is the current process
//  $I_x$  is the input message of the candidate operation to
be added after CP
 $t = |\text{PATHS}(\{I_x\})|$ 
 $\{S_1, S_2, \dots, S_t\} = \text{DATA-MEDIATION}(CP, I_x)$ 
 $W_i = 1/t$ 
return  $S_{dm} = \sum_{i=1}^t W_i * S_i$ 
//  $\{S_1, S_2, \dots, S_t\}$  are scores for all paths of  $I_x$ 
//  $W_i$  is the weight of path  $i$  of  $I_x$ 

```

Figure 8. FORWARD-DATA-MEDIATION-SCORE algorithm

The S_{pe} score is calculated based on a formal specification, which includes pre-conditions, effects, initial state and goal state. These annotations require the use of WSDL-S [6], which extends SAWSDL. In the automatic Web service composition literature, logic-based languages are used to specify pre-conditions and effects as well as describe states of a process/workflow. This allows planning algorithms to be utilized to build complete process specifications. We are doing a similar thing here, but only for a small portion of the overall design. It is well-known that the complexity trade-off is a challenge to deal with (i.e., low complexity logic leads to efficient reasoning, but limited expressivity).

For the more difficult problem of automatic composition, our prior work [11] extended the GraphPlan [12] algorithm and utilized tri-state propositional logic and description logic. In our current work which is less

dependent on using a planner, we intend to allow more expressive pre-conditions and effects by using the Rule Interchange Format (RIF) [13] and a rule engine such as Prova [14] or OpenRules [15].

To calculate the score based on a formal specification, a rule engine is used to do the logic reasoning. The formal specification score S_{pe} has two parts, the condition score S_c and the state score S_s . Their weights are W_c and W_s , respectively. Initially, $W_c = W_s = 1/2$, but can be trained by machine learning algorithms.

$$S_{pe} = W_c * S_c + W_s * S_s$$

The condition score (S_c) is decided by whether the current state (st) entails¹ the pre-condition of the candidate operation $pre(OP_x)$. The current state (st) is maintained for the current process. A candidate operation (OP_x) will be connected to the last operation (OP) of the current process.

$$S_c = \begin{cases} 1 & \text{if } st \models pre(OP_x) \\ 0 & \text{otherwise} \end{cases}$$

After a candidate operation is connected to the process, the current state (st) will change to a new state (st'). The new state (st') is determined by applying the effects of the candidate operation $effect(OP_x)$ to the current state (st).

$$\text{apply}(effect(OP_x), st) \rightarrow st'$$

The state score (S_s) is decided by how well the candidate operation (OP_x) will help toward reaching the goal state (st_g). In our future work, we will be exploring two ways to measure the distance from a given state to the goal state. The first approach uses a planner to determine the number of steps required to get to the goal state. The second, which can be thought of as a heuristic, involves computing a distance metric, i.e., $S_s = \text{Distance}(st', st_g)$.

3.1.2. Backward suggestion

A backward suggestion is used to recommend a service operation (OP_x) to be placed before the current process (CP) as shown in Figure 9. The backward suggestion ranking score (S) is similarly calculated based on three parts as the forward suggestion. However, there are some differences in calculating each part.



Figure 9. Backward suggestion

The data mediation score for a backward suggestion (Figure 10) is based on how well the input message (I) of the first operation (OP_1) of the current process (CP) is matched by the output message of the candidate operation (OP_x).

¹ http://en.wikipedia.org/wiki/Math_symbol
entail, $A \models B$ means the sentence A entails the sentence B , that is in every model in which A is true, B is also true.

The functionality score (S_{fn}) is calculated in the same way as for the forward suggestion.

```

BACKWARD-DATA-MEDIATION-SCORE(CP, OPx)
// CP is the current process
// OPx is the candidate operation that will be added
// before the current process
I = INPUT-MESSAGE( FIRST-OPERATION(CP))
t = |PATHS ({ I })|
{S1, S2, ..., St} = DATA-MEDIATION (OPx, I)
Wi = 1/t
return Sdm = Σi=1t Wi * Si
// {S1, S2, ..., St} are scores for all paths of I
// Wi is the weight of path i of I

```

Figure 10. BACKWARD-DATA-MEDIATION-SCORE algorithm

Compared to the forward suggestion score algorithm, the formal specification score (S_{pe}) still comes from two parts, the condition score (S_c) and the state score (S_s). However, the difference for the condition score (S_c) is that it is decided by whether the new state (st') after the execution of the candidate operation (OP_x) entails the pre-condition of the first operation (OP_1) in the current process. Instead of the current state, it applies the effects of the candidate operation to a state (st) that is picked among all possible states that can be applied to OP_x and as close as possible to the initial state (st_{in}). If $st = st_{in}$, then OP_x is the first operation in the new process.

$$\text{apply}(effect(OP_x), st) \rightarrow st'$$

$$S_c = \begin{cases} 1 & \text{if } st' \models pre(OP_1) \\ 0 & \text{otherwise} \end{cases}$$

The difference for the state score (S_s) is that it is the distance between initial state (st_{in}) and the state st .

$$S_s = D(st_{in}, st)$$

3.1.3. Bi-directional suggestion

Figure 11 shows that the bi-directional suggestion is to recommend a service (OP_x) to place between two operations (OP_1, OP_2) in the current process or more generally a workflow prefix and suffix. Its suggestion score calculation also includes three parts: the data mediation (S_{dm}), functionality (S_{fn}) and formal specification (S_{pe}) scores.

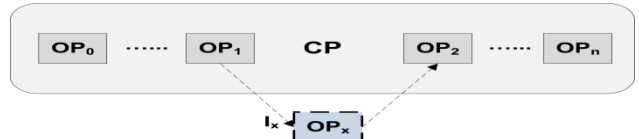


Figure 11. Bi-directional suggestion

As shown in Figure 12, the data mediation score algorithm for the bi-directional suggestions invokes both the forward and the backward data mediation score

algorithms. The final score is the average of the two scores.

The functionality score (S_{fm}) is computed the same way as the other two types of suggestions.

```

BIDIRECTION-DATA-MEDIATION-SCORE (CP, OP1,
OP2, OPx)
// CP is the current process
// OPx is the candidate operation to be added between OP1
and OP2 in CP
CP1 = the sub-process including OP1 and all operations
before OP1
CP2 = the sub-process including OP2 and all operations
after OP2
Ix = INPUT-MESSAGE(OPx)
return Sdm = (FORWARD-DATA-MEDIATION-SCORE
(CP1, Ix) + BACKWARD-DATA-MEDIATION-SCORE
(CP2, OPx)) / 2

```

Figure 12. BIDIRECTION-DATA-MEDIATION-SCORE algorithm

The formal specification score (S_{pe}) is derived from only one part, the condition score (S_c). Figure 13 shows a simplified view of a BPEL process. Focusing on the state variables, which are typically assigned from receive and invoke messages, the pre-conditions of OP_x will be satisfied if state st_1 entail pre (OP_x). Similarly, the pre-conditions of OP_2 will be satisfied if the application of the effect (OP_x) to state st_1 entails pre (OP_2).

$$S_c = \begin{cases} 1 & \text{if } st_1 \models \text{pre}(OP_x) \text{ and } st_x \models \text{pre}(OP_2) \\ 0 & \text{otherwise} \end{cases}$$

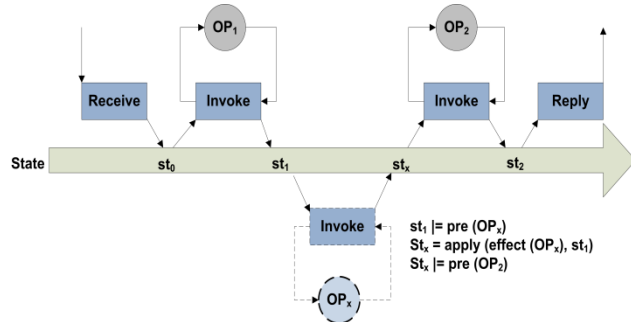


Figure 13. States in a BPEL process

4. Similarity measures

Concept Similarity (CS) [16, 17] computes the overall similarity between two concepts by comparing the concepts as well as their properties. To measure the similarity between two concepts, C_0 playing an output role and C_1 playing an input role, from the same ontology the weighted sum of $Concept_{sim}$, $Property_{sim}$ and $Syntactic_{sim}$ is used.

$$CS = \frac{W_1 * Syntactic_{sim} + W_2 * Concept_{sim} + W_3 * Property_{sim}}{W_1 + W_2 + W_3}$$

$Syntactic_{sim}$ computes the syntactic similarity between the names and descriptions of the two concepts. If no description is attached with both the services, only name comparison is used to compute $Syntactic_{sim}$.

$$Syntactic_{sim} = \begin{cases} \frac{w_4 * NameMatch(C_1, C_0) + w_5 * DescrMatch(C_1, C_0)}{w_4 + w_5}, & Descr[C_1], Descr[C_0] \neq \emptyset \\ NameMatch(C_1, C_0) & otherwise \end{cases}$$

Here, $NameMatch$ and $DescrMatch$ compute the similarity between the names and descriptions respectively, using a string matching algorithm such as N-Gram and suffix tree.

$Concept_{sim}$ computes the similarity based on the relative position of the two concepts, A_1 and A_0 in the ontology. As shown in Figure 14, Case I of a perfect match would occur if A_1 and A_0 are annotated with same/equivalent concepts. Since, we are matching input to output; for any input concept its sub-concept is a better match than its super-concept. If A_0 is a sub-concept of A_1 (Case II), $Concept_{sim}$ decreases exponentially with the level of specialization at a decay rate of λ_1 . When A_0 is a super-concept of A_1 (Case III), $Concept_{sim}$ decreases exponentially with the level of generalization, but at a faster rate of λ_2 . Case IV, when they share a common ancestor experiences the fastest decay in the $Concept_{sim}$ measure. Figure 14 depicts the various possible relative levels of concepts in an ontology and the associated decay rates. Here the decay rates are in increasing order $\lambda_1 < \lambda_2 < \lambda_3$ and $C_1 = \text{ancestor}_x(C_2)$ iff concept C_1 is the ancestor x levels above concept C_2 in the ontology.

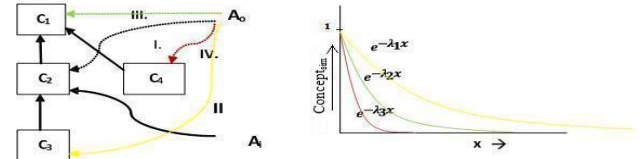


Figure 14: The relative positions of annotated concepts and their decay graphs.

$$Concept_{sim} = \begin{cases} 1, & C_1 \equiv C_0 \\ e^{-\lambda_1 x}, & C_1 = \text{ancestor}_x(C_0) \\ e^{-\lambda_2 x}, & C_0 = \text{ancestor}_x(C_1) \\ e^{-\lambda_3 x}, & \exists C_* | C_* = \text{ancestor}_{x_1}(C_0) \\ & C_* = \text{ancestor}_{x_2}(C_1) \\ & x = x_1 + x_2 \end{cases}$$

Given that concept C_1 has properties P_1 and concept C_0 has properties P_0 , $Property_{sim}$ calculates an overall similarity measure between P_1 and P_0 . The properties can be matched as one-to-one mappings, using the Hungarian algorithm [16]. Here, the property similarity $Property_{sim}$, will find a maximal mapping between property sets P_1 and

P_O (see [16]). Also $Property_{sim}$ will be penalized with a penalty for every unmatched property.

Another approach that can be adopted is allowing a one-to-many mapping from P_O to P_I to find an optimal match between the corresponding properties. Iteratively every property p_I in P_I will be matched to every property p_O in P_O to get an optimal $Property_{match}(p_I)$ for all $p_I \in P_I$. This approach is required when one p_O is allowed to match to more than one p_I , which can be the case for Web services.

In both scenarios, to compute similarity between individual properties $p_I \in P_I$ and $p_O \in P_O$, $Property_{sim}$ uses $Property_{match}$. $Property_{match}$ extended from property similarity $propSim$ as described in [16], compares the two properties using their syntactic names and descriptions, ranges, and cardinalities.

$$Property_{match} = \sqrt[3]{(Range_{sim} * Cardinality_{sim} * Syntactic_{sim})}$$

$Property_{match}$ calculates the property similarity based on the components described next: $Range_{sim}$ depends on the data type compatibility when both ranges have primitive types, on $Concept_{sim}$ when both are concepts, and is set to zero when one is a concept and the other is a primitive type. $Cardinality_{sim}$ favors the case where the number of values required by the input is the same as or lesser than that provided by the output. As $Property_{match}$ is a geometric mean of the components, exceptionally low value of any one component lowers the overall $Property_{match}$ by a large amount.

5. Related work

Much research has been done on Web service composition, including enhanced tool support as well as automation of process composition. Most of the work done [18-22] in Web service composition does not focus on the semantic heterogeneity. Data mediation techniques can be used to resolve such issues. Some researchers [23] investigated data mediation as a ontology mapping problem. For every Web service, they create a service ontology using OWL-S and then achieve data mediation by mapping from one such service ontology to another ($WSDL_1 \rightarrow serviceOntology_1 \rightarrow domainOntology \rightarrow serviceOntology_2 \rightarrow WSDL_2$). Our data mediation approach, however, utilizes SAWSDL to establish a mapping from WSDL/XSD elements to concepts in domain ontology ($WSDL_1 \rightarrow domainOntology \rightarrow WSDL_2$).

Many studies [10, 24, 25] focus on the Web service discovery and ranking utilizing Semantic Web technologies. However, their service ranking does not aim to help service composition and when matching the input/output of a service, they only consider the message level of the input/output. Our approach goes through the whole XML schema of the message to match the input/output, which will result in a more accurate match.

The survey by Schaffner and Meyer [26] points out the importance of mixed initiative semi-automatic composition techniques and reviews the work of [27-29] on an OWL-S composer, CAT, PASSAT, respectively. They state that combining human expertise with machine assistance currently provides a more practical approach than fully automatic Web service composition. Kim et al. [28] developed a novel tool for finding errors/deficiencies in workflow designs and providing written suggestions for how to fix the problems. The OWL-S composer [27] uses backward chaining along with filtering to help the user select the next service/operation to add to their composition. Although PASSAT [29] is from another domain, the planning domain, it is relevant since it promotes interactive planning. A recent paper [30] uses Case-Based Reasoning (CBR) to make suggestions.

Of the research efforts discussed above, our work is more similar to CAT. While CAT focuses on finding deficiencies, our work is more integrated with the overall design process allowing users at any time to request suggestions from our design tool. Moreover, our algorithms consider data mediation issues in detail.

6. Conclusions and future work

As pointed out by several researchers, there is a growing need for practical and effective techniques for semi-automatic Web service composition. A critical need is to develop an approach for making suggestions to aid users composing services. In order to provide this capability, we have developed ranking schemes and suggestion algorithms that will enable a user at various times during the design process to request recommendations from our assistive tool, thereby allowing them to design workflows more quickly and more reliably.

Our algorithms can make forward, backward and bi-directional suggestions. The ranking score for suggestions comes from data mediation, functionality and formal specification. To support this, we developed a new data mediation algorithm that extends our previous work on top-down and bottom-up data mediation. Our data mediation algorithm ranks all of the paths within the input/output messages based on their semantic annotations given in SAWSDL. It has many useful features: It lowers the semantic annotation requirements in that it can function without pre-conditions and effects, without lifting- and lowering-*SchemaMappings* and even with missing *modelReferences*. It further considers the global inputs and the outputs from all preceding operations, which allows it to handle more general cases.

Future work includes finishing the formal specification part of our prototype system, developing a cascaded service suggestion algorithm for use when a single suggestion does not suffice, and conducting an evaluation that compares our various algorithms with each other,

exploring the tradeoff that better suggestions require greater effort in semantic annotation. Finally, we plan to implement some of these algorithms in our parent project, the Galaxy (<http://galaxy.psu.edu/>) bioinformatics integration and workflow system.

References

- [1] J. Hendler, D. Wu, E. Sirin, D. Nau, and B. Parsia, "Automatic Web Services Composition Using SHOP2," in *Proceedings of The Second International Semantic Web Conference (ISWC)*, 2003.
- [2] S. Su, Q. Liang, L. N. Chakarapani, R. N. Chikkamagalur, and H. Lam, "A Web Service composition framework: Discovery, description and invocation," in *the Sixth International Conference on Electronic Commerce (ICECR-6)*, Dallas, Texas, 2003.
- [3] Z. Wang, J. A. Miller, J. C. Kissinger, R. Wang, D. Brewer, and C. Aurrecochea, "WS-BioZard: A Wizard for Composing Bioinformatics Web Services," in *SWF'08, in conjunction with SCC'08*, Honolulu, Hawaii, pp. 437-444, Jul. 2008.
- [4] R. Wang, D. Brewer, S. Shastri, S. Swayampakula, J. A. Miller, E. T. Kraemer, and J. C. Kissinger, "Adapting the Galaxy Bioinformatics Tool to Support Semantic Web Service Composition," in *IEEE 2009 Third International Workshop on Scientific Workflows (SWF 2009) In conjunction with 7th IEEE International Conference on Web Services (ICWS 2009)*, Los Angeles, CA, USA, July 10, 2009.
- [5] "SAWSDL," <http://www.w3.org/2002/ws/sawSDL/spec/>.
- [6] R. Akkiraju, J. Farrell, J. A. Miller, M. Nagarajan, A. Sheth, and K. Verma, "Web Service Semantics - WSDL-S," in *W3C Workshop on Frameworks for Semantics in Web Service (W3CW'05)*, Innsbruck, Austria, pp. 1-5, June 2005.
- [7] M. Nagarajan, K. Verma, A. P. Sheth, and J. A. Miller, "Ontology Driven Data Mediation in Web Services," *International Journal of Web Services Research (IJWSR)*, USA, vol. 4, pp. 104-126, Dec 2007.
- [8] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar, "A Context Model for Semantic Mediation in Web Services Composition," *Conceptual Modeling - ER 2006*, vol. 4215/2006, pp. 12-25, October 28, 2006.
- [9] M. Nagarajan, K. Verma, A. P. Sheth, J. Miller, and J. Lathem, "Semantic Interoperability of Web Services – Challenges and Experiences," in *IEEE Intl Conf on Web Services (ICWS 2006)*, Salt Lake City, Utah, USA, July 9-13, 2007.
- [10] K. Verma, A. Sheth, S. Oundhakar, K. Sivashanmugam, and J. Miller, "Allowing the use of Multiple Ontologies for Discovery of Web Services in Federated Registry Environment," Department of Computer Science, University of Georgia, Athens, Georgia. Technical Report #UGA-CS-LSDIS-TR-07-011, February 2007.
- [11] Z. Wu, A. Ranabahu, K. Gomadam, A. P. Sheth, and J. A. Miller, "Automatic Composition of Semantic Web Services using Process Mediation," in *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS'07)*, Funchal, Portugal pp. 453-461, Jun. 2007.
- [12] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," *Pearson Education*, 2003.
- [13] "RIF," www.w3.org/TR/REC-rdf-syntax/.
- [14] "Prova," <http://www.prova.ws/>.
- [15] "OpenRules," <http://openrules.com>.
- [16] S. Emani, "A Comparative Evaluation of Semantic Web Service Discovery: Algorithms and Engines." vol. Computer Science Athens: University of Georgia, 2009.
- [17] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Information Technology and Management Journal*, vol. 6, pp. 17-39, Feb 2005.
- [18] F. Lecue, O. Boissier, A. Delteil, and A. Leger, "Web Service Composition as a Composition of Valid and Robust Semantic Links," *International Journal of Cooperative Information Systems (IJCIS)*, vol. 18, pp. 1-62, March 2009.
- [19] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi, "Automated Synthesis of Composite BPEL4WS Web Services," in *IEEE Intl Conference on Web Services (ICWS'05)*, 2005.
- [20] D. Roman, U. Keller, H. Lausen, J. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussle, and D. Fensel, "Web Service Modeling Ontology," *Applied Ontology*, vol. 1, 2005.
- [21] Z. Duan, A. Bernstein, P. Lewis, and S. Lu, "A Model for Abstract Process Specification, Verification and Composition," in *The 2nd Intl conf on Service oriented computing*, 2004.
- [22] J. Rao, D. Dimitrov, P. Hofmann, and N. Sadeh, "A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP's Guided Procedures Framework," in *The IEEE Intl Conf on Web Services (ICWS'06)*, 2006.
- [23] S. Izza, L. Vincent, and P. Burlat, "Exploiting semantic web services in achieving flexible application integration in the microelectronics field," *Computers in industry*, vol. 59, no. 7, pp. 722-740, 2008.
- [24] F. Emekci, O. D. Sahin, D. Agrawal, and A. E. Abbadi, "A peer-to-peer framework for web service discovery with ranking," in *IEEE International Conference on Web Services (ICWS'04)*, Washington, DC, USA, 2004, p. 192.
- [25] D. Skoutas, A. Simitsis, and T. Sellis, "A Ranking Mechanism for Semantic Web Service Discovery," in *IEEE Congress on Services*, 2007, pp. 41-48.
- [26] J. Schaffner and H. Meyer, "Mixed Initiative Use Cases For Semi-Automated Service Composition: A Survey," in *International Workshop on Service Oriented Software Engineering (IW-SOSE'06)*, located at ICSE 2006., Shanghai, China, 27-28 May, 2006.
- [27] E. Sirin, B. Parsia, and J. Hendler, "Filtering and Selecting Semantic Web Services with Interactive Composition Techniques," *IEEE Intelligent Systems*, vol. 19, pp. 42-49, 2004.
- [28] J. Kim, M. Spraragen, and Y. Gil, "An intelligent assistant for interactive workflow composition," in *IUI'04: Proceedings of the 9th international conference on Intelligent user interface*, New York, NY, USA, 2004, pp. 125-131.
- [29] K. L. M. e. al, "PASSAT: A User-centric Planning Framework," in *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, AAAI, Houston, TX, USA, 2002.
- [30] E. Chinthaka, J. Ekanayake, D. Leake, and B. Plale, "CBR Based Workflow Composition Assistant," in *Proceedings of the 2009 Congress on Services* pp. 352-355, July 06-10, 2009