Building Quick Service Query List using Wordnet for Automated Service Composition

Kaijun Ren, Jinjun Chen, Nong Xiao, Junqiang Song, Jinyang Li

1 College of Computer, National University of Defense Technology, Changsha, Hunan 410073, P.R. China
renkaijun@nudt.edu.cn; jchen@swin.edu.au
2 CS3-Centre for Complex Software Systems and Services, Swinburne University of Technology, Melbourne 3122, Australia

Abstract

Current existing semantic composition methods mainly rely on ontology reasoning to support automated service composition. However, in reality, ontologies are generally unavailable or ontology reasoning is time-consuming; thus existing semantic composition methods are becoming impractical in the general service integration field. To address this problem, in this paper, we present an innovative composition technique by combining Wordnet with ontologies together to build an extended quick service query list (EQSQL) for supporting automated service composition. In EQSQL, data structures are designed particularly to record service information and their associated semantic concepts by previously processing semantic-related computing during service publication stage. Based on EQSQL, not only a quick response can be achieved, but the semantically-similar composition quality as well even if there is a lack of concrete domain-dependent ontologies for a user query.

1. Introduction

Web service composition, by binding two or more existing services into a new one, is emerging as a prospect technology for supporting large-scale, sophisticated business process integration in a variety of e-science or e-business areas. With the increase in importance of business process automation and highly dynamic nature of the internet, automated composition methods have been the most promising methods to be applied in real world scenario. In particular, semantics, which are employed to describe service interfaces such as OWL-S described services, have been proposed as a key to increasing automation in combining multiple services by considering their control-flow and data-flow dependencies to meet user’s requirements. Recently, many different research efforts by cooperating with semantics have been proposed to aim at automating service composition [1-6]. However, despite the merits and the importance of semantic information contained by services, most existing typical semantic methods for automatic service composition still face the following problems. Firstly, these methods mainly take the direct ontology reasoning manner which is generally time-consuming to generate composition plans so that performance issues probably occur. Secondly, semantic information, such as OWL ontologies are needed to annotate service interfaces. However, in reality, such ontologies are generally unavailable. Finally, a user is also expected to specify his query with associated semantic information, which is not realistic so that many current composition methods become impractical to be applied in highly-dynamic internet service environments where a user often demands a quick query response and convenient service retrieval means just like using google search engine.

To address the aforementioned problems, in this paper, we present a novel composition technique based on our previous work[7] by combining Wordnet with ontologies together to build EQSQL for supporting automated service composition. In EQSQL, time-consuming ontology reasoning and other semantic-related computing can be processed in advance at service publication stage. Particularly, a lexical database Wordnet is employed to overcome the lack of available ontologies when publishing a service to EQSQL. As such, for a given user query, selecting and combining services from EQSQL to consider their inputs/outputs semantic compatibility can be automatically and quickly done by our proposed planning algorithm.
The remainder of this paper is structured as follows: Section 2 gives an overview of QSQL (Quick Service Query List) principles. Section 3 provides concrete methods for extending QSQL called EQSQL by using Wordnet. Section 4 presents the experiment. Section 5 gives the related work. The final section gives the conclusion and future work.

2. Overview of QSQL

In order to address the low discovery efficiency brought by the traditional semantic service discovery algorithm based on the direct reasoning manner, we have proposed a pre-reasoning based service discovery method in [8] where a preliminary quick service query list (QSQL) has been built to support single service discovery. Then, basing on this work, we have successfully presented an efficient composition algorithm in [7] to automatically construct composition plans to meet user’s requirements. In order to clearly understand our previous built QSQL, here we give a brief introduction. Generally, when service providers publish a service to a registration center such as UDDI for facilitating the invocation by other services, an interface description document such as WSDL should be available on the web. To better support service searching and matching than the keyword-based manner, such service description documents should be annotated by semantically-meaningful concepts such as OWL ontologies[9]. Currently, some semantic annotation tools [10-13] have been proposed to finish this work. As a result, based on these semantically-annotated documents, semantic discovery and composition engines by using OWL/RDF reasoners can intelligently search suitable services or combine multiple services to match user requests. However, current such methods are inefficient because ontology reasoning is time-consuming. To overcome the low efficiency problem, we have designed and built QSQL in [8] to make a large scale of ontology reasoning processed in advance when a semantically-annotated service is published. Specifically, QSQL can depend on specially-designed data structures to record the important reasoning relationships and the published service information. Such data structures primarily include two parts. One part is the domain of link, and the other part is the domain of data. The purpose of link domain is to record semantic relationships between ontology concepts. The other part is the domain of data, which is mainly used to record service information in corresponding INPUT/OUTPUT data vectors such as Exact_vector、Plugin_vector、Sib_vector、Grapar_vector、Grachd_vector according to their corresponding semantic matching degree. More details are referred to the literature [8].

3. Extending QSQL (EQSQL) By Using Wordnet

3.1. Current limitations

To simplify the complexity of composition problems, we have made three assumptions which are generally processed in the traditional composition methods in our previous methods of [7, 8] as follows. First, the parameter models of any published services in QSQL can be associated with the concrete ontology domains. Second, the semantic compatibility between one service and its succeeding one is assumed to be processed in the uniform semantic spaces. Third, a user query including inputs/outputs is also required to be mapped with concrete semantic information. About the first assumption, specifying concrete domain ontologies to parameter models is a hard task for service providers to finish. Actually, in current ontology engineering field, uniform and consensus ontologies still have a serious lack in many application domains. Additionally, creating ontology is not simple task, which usually requires many knowledge engineers to cooperate with each other. In particular, even if there is an available ontology, it is still not easy for a service provider who does not have much knowledge on ontologies to select the exact semantic information to annotate their service description documents. Back to the second assumption, there is a wide exist about the semantically-heterogeneous problems between different services which are often across multiple domains. For example, service model ws1(US$toRMB) and ws2(getStockPrice) possibly belong to two different domains such as Bank and Stock respectively. Thus, the parameter concept ‘USdollar’ of ws2 and the concept ‘USprice’ of ws1 are possibly referred by different ontology concepts from different ontology domains. Further, the semantic connecting between ws2 and ws1 can probably produce the semantic inconsistent problem. Analogously, the semantically-heterogeneous problems between different services are more like the same problems existing in the information integration and information retrieval fields. About the third assumption, it is not realistic to expect a general user to specify the exact semantic information for mapping his query. To the worst case, such semantic information is not available.

In reality, for a non-expert user, they are expecting that searching services should be as simple as possible just like using a Google search engine to do a general information retrieval where they only need to type their
needed information and then get the ranked results they want. Apparently, with the increasing number of available services on the web, the above-mentioned three assumptions are becoming some barriers for making semantic composition methods become more practical to reflect the real web behavior. Considering the comprehensive advantages brought by our QSQL composition methods in [7], we use a lexical database Wordnet to extend QSQL by eliminating the above-discussed limitations for supporting automated service composition. Thus, our methods will become more practical and reasonable to support general service retrieval. In the following sections, we will discuss the concrete strategies about how to extend QSQL.

3.2. Extending QSQL by Using Wordnet

When a service is published, it is possible that there is a lack of a concrete application ontology which is used to annotate the associated WSDL documents. In our methods, we use a general semantically-lexical database called WordNet[14] to approximately resolve this problem. Recently, WordNet has been widely used in Information Retrieval community by calculating and comparing their semantic similarity between words. WordNet can also be seen as ontology for natural language terms. It contains around 100,000 terms, organized into taxonomic hierarchies. Nouns, Verbs, Adjectives and Adverbs are grouped into synonym sets (synsets). The synsets are also organized into senses (i.e., corresponding to different meanings of the same term or concept). The synsets (or concepts) are related to other synsets higher or lower in the hierarchy by different types of relationships. The most common relationships are the Hyponym/Hypernym (i.e., Is-A relationships), and the Meronym/Holonym (i.e., Part-Of relationships). It is commonly argued that language semantics are mostly captured by nouns (and noun phrases) so that it is common to build retrieval methods based on the relationships Hyponym/Hypernym and Meronym/Holonym. Similarly, in our methods, we mainly merge the relationship Hyponym/Hypernym together with our previous definition of semantic relationships in [8](definition 1-5), and the merged relationships are listed as below:

\[
\begin{align*}
\text{has-hypernym} & \Rightarrow \text{has-supclass} \quad (1) \\
\text{has-hypernym} & \Rightarrow \text{has-subclass} \quad (2) \\
\text{has-synonym} & \Rightarrow \text{has-equalclass} \quad (3) .
\end{align*}
\]

Generally, the XML schema is used in WSDL descriptions to provide a basic structure to the data exchange by the web service. It therefore provides a minimal containment relationship using the complexType, simpleType and element constructs. Actually, no matter how complexType is complicated and possibly nested by other complexType, complexType can always be expressed by simpleType. Therefore, the inputs/outputs message of an operation in a service can be recursively divided into independent parameter which can be associated with the corresponding simpleType. For example, The left part of Figure 1 shows description for services “StockQuote”; then the inputs/outputs message body of the operation “getStockPrice” can be divided into independent inputs/outputs parameters such as input parameters “city”, “stock”, “time”, output parameter ‘price’. In the left part of Figures 1, the red line expresses the nested hierarchy. Here we assume that the concepts from WSDL have meaningful names. Further, for each parameter name, we can rely on the WordNet to make sure its similar concepts such as synonyms, hyponyms, hypernyms. However, a same word in WordNet can possibly have multiple meanings (senses). For example, ‘city’ in WordNet has three senses including the follows: sense1 “a large and densely populated urban area; may include several independent administrative districts”; sense2 “an incorporated administrative district established by state charter”; sense3 “people living in a large densely populated municipality”. Thus, for each parameter name from WSDL, which kind of sense it belongs to, is needed to be judged in the first place. For this, we take semi-automated methods by combining the human-judging means together. First, we retrieve all senses of a word in WordNet, then retrieve test description from WSDL. According to the context, we can make sure which kind of sense a parameter name belongs to. Actually, for each parameter, selecting a concrete sense to map the corresponding parameter name can also be made by automatic text processing tool. Currently, many nature language processing methods have been proposed to approximately resolve this problem by calculating the semantic similarity between texts [15, 16]. Once a concrete sense for a parameter name is selected, we can make sure its corresponding Synonyms, Hyponyms, Hypernyms, Meronyms, Holonyms from Wordnet. For example, for the parameter name “city” of service “StockQuote”, sense “a large and densely populated urban area; may include several independent administrative districts” has been made in the first place by text processing analysis. Then Synonyms, Hyponyms, Hypernyms, Meronyms, Holonyms are respectively retrieved from Wordnet, and the right part of Figure 1 shows the corresponding hierarchy. As shown in the right part of Figure 1, the small circle which are full of white color are Hypernyms of “city”, the blue-full circles are Synonyms, the yellow-full circles are Hyponyms. The grey-full circles are Meronyms.
According to the merged semantic relationships (1), (2), (3), the service publication methods in [8] restart to insert the associated concepts from Wordnet to QSQL for recording the published service information. The basic procedures by using Wordnet for publishing a service are concluded as follows:
1) Parsing a WSDL document;
2) Separating the nested complexType structures;
3) Extracting the parameter names from simpleType;
4) Extracting text description from WSDL document;
5) For each parameter name, selecting a concrete sense by automatic text analysis or human-judgments if there is a lack of available domain ontologies such as OWL ontologies.
6) According to the concrete sense, retrieving the corresponding synonyms, Hyponyms, Hypernyms, Meronyms, Holonyms, etc.
7) According to the merged semantic relationships(1), (2), (3), using service publication algorithm in [8] to insert the related concepts to QSQL for forming the extended EQSQL.

The definition of symbols in algorithm 1

**Algorithm 1: FindAllProcessPlans**

<table>
<thead>
<tr>
<th>WSRI (O)</th>
<th>user’s query</th>
<th>WSR (O)</th>
<th>service models of EQSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>: specified inputs by user</td>
<td>t</td>
<td>: inputs of service model</td>
</tr>
<tr>
<td>o</td>
<td>: specified outputs by user</td>
<td>o</td>
<td>: outputs of service model</td>
</tr>
<tr>
<td>V1</td>
<td>: OutputExact_vector</td>
<td>V2</td>
<td>: InputExact_vector</td>
</tr>
<tr>
<td>V3</td>
<td>: OutputPlugin_vector</td>
<td>V4</td>
<td>: InputPlugin_vector</td>
</tr>
<tr>
<td>V5</td>
<td>: OutputSub_vector</td>
<td>V6</td>
<td>: InputSub_vector</td>
</tr>
<tr>
<td>V7</td>
<td>: OutputGrachd_vector</td>
<td>V8</td>
<td>: InputGrachd_vector</td>
</tr>
</tbody>
</table>

**Figure 1.** Part of WSDL description

**Table 1.** The definition of symbols in algorithm 1

**Legend:** — has – hypernym — has – hyponym

3.3. EQSQL Planning Algorithm for Automated Service Composition

We have presented an efficient planning algorithm in [7] to compose multiple services from QSQL by considering their inputs/outputs dependencies for meeting a given user query. With respect to EQSQL, the basic planning algorithm for automatic service composition is shown by Algorithm 1. As we see, the main processing procedures for a given query are almost the same as the algorithm in [7]. However, for EQSQL, the algorithm can support a general user query, and such queries are unnecessarily required to be associated with semantic information.
4. Experiment

Based on EQSQL, for a given user query, composition plans can be automatically found by algorithm 1 without much user interaction. Moreover, we have proved the high efficiency for each query in comparison to the traditional composition methods in [7]. In this section, we mainly demonstrate how well our proposed methods are for supporting automated service composition. We still use the same service collection in [7]. But differently, we republish these services for forming EQSQL by using Wordnet. In our experiment, we have produced 15 different composition queries, each with different composition length. For each query, we analyze all generated composition plans. For each single involved service in a composition plan, we respectively calculate the precision and the recall. Then, the total recall and precision rate for each query can be achieved by calculating the average value. Figure 2 shows the average precision/recall rate for 15 queries. As we see in the Figure 2, the average precision rate for the average query is about 72.9% which is less than the average recall with around 81.8%. Due to the high recall rate, we can conclude that a more general composition query can be supported by the use of Wordnet. However, compared to OWL ontologies, Wordnet has a limited semantically-expressed capability which leads to that the precision rate decrease. Hence, combining Wordnet with ontologies together to building EQSQL can gain comprehensive benefits for supporting automated service composition.

![Figure 2. Precision/Recall comparison](image)

5. Related Work

Over the last several years, different representative composition strategies have been proposed in the literatures. To the best of our knowledge, we categorize and analyze them below. [17] Proposes a goal-driven service composition approach based on a Petri nets modeling technique. With this method, goals are recursively decomposed to be sub-goals until they become executable elementary goals. In complicated dynamic environments goal decomposition may be time-consuming tasks, because of similar complex AI searching. Many research efforts regard the service composition problem as AI planning problems, where a planner is used to determine the combination of actions needed to reach the goal[18]. With this approach, an explicit goal definition has to be provided, whereas such explicit goals were usually not available. Some composition methods are based on formal models such as process algebra [19], Finite-State Machine [20], Formal Concept Lattice[21], Situation Calculus[22]. The above formal composition methods are very useful to check the existence of a composition and return a composition if one exists. However, the automatic process is hard to achieve. Semantics have been proposed as a key to increasing automation in applying Web services and managing Web processes within and across enterprises, and the World Wide Web Consortium (W3C) has recently finished an important standard--Semantic Annotations for WSDL and XML Schema (SAWSDL) for Semantic Web services(SWSs)[23]. Currently, there are many existing semantic service composition methods [1-6]. However, these existing methods still either stay in a semi-automated state, (which means composition needs a high level of user interaction) or expect that a user and service providers have much knowledge on ontologies which is often not realistic. By comparison, with EQSQL, our technique supports not only automated processing for quick service composition, but also more general queries which do not need a user to have much knowledge on semantics. Hence, our technique is more practical and more promising to be applied to address business integration problems.

6. Conclusions and Future Work

Considering ontologies are generally unavailable or ontology reasoning is much time-consuming, in this paper, we have presented an innovative composition technique by combining Wordnet with ontologies together to build EQSQL for supporting automated service composition. Based on EQSQL, not only a quick response can be achieved for each query, but the semantically-similar composition quality as well. More importantly, our technique can support a more general query where a user does not need to have much knowledge on semantics. An experiment has been conducted to demonstrate the effectiveness of our proposed technique. In conclusion, our technique is
more practical and more reasonable than other typical ones. In future, we will combine non-functional properties of services together to construct a concrete business application.

7. Acknowledgements

We are grateful for the English proofreading by D. May. This paper is partially supported by the National “973” Research Plan Foundation of China under Grant No. 2003CB317008, National Nature Science Foundation of China under Grant No. 60573135, 40505023 and 60736013, Swinburne Dean’s Collaborative Grants Scheme 2007-2008, and Swinburne Research Development Scheme 2008.

8. References