IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 10, 2019, accepted May 2, 2019, date of publication May 7, 2019, date of current version May 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2915371

Composition Context-Based Web Services Similarity Measure

FENG ZHANG[1,2], QINGTIAN ZENG[1,2], HUA DUAN[3], AND CONG LIU[1]

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]Shandong Key Laboratory of Wisdom Mine Information Technology, Shandong University of Science and Technology, Qingdao 266590, China
[3]College of Mathematics and System Science, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding authors: Feng Zhang (zhangfeng@sdust.edu.cn) and Qingtian Zeng (qtzeng@163.com)

This work was supported in part by the Humanities and Social Science Research Project of the Ministry of Education under Grant 19YJCZH240 and Grant 18YJAZH017, in part by the NSFC under Grant 61472229 and Grant 31671588, in part by the Science and Technology Development Fund of Shandong Province of China under Grant 2016ZDJS02A11 and Grant ZR2017MF027, in part by the Taishan Scholar Climbing Program of Shandong Province, in part by the SDUST Research Fund under Grant 2015TDJH102, and in part by the Scientific Research Foundation of Shandong University of Science and Technology for Recruited Talents under Grant 2014RCJJ049.

ABSTRACT Web services similarity measure is an important problem in service computing area, which is the technological foundation of service substitution, service discovery, service recommendation, and so on. Most of the existing works use a static description of services to measure the similarity between two services. However, the interaction information of Web services recorded in the historical compositions is totally neglected. In this paper, we propose a novel Web services similarity measure approach based on the notion of service composition context. Specifically, we first introduce three types of parameter correlations between service input and output parameters. These correlations can be obtained from existing services compositions. Based on parameter correlations, we propose the service composition context model. Through the composition context of a service, the composition context network is constructed using contexts of all services. Then, we propose to measure the similarity between any two services using the PersonalRank and SimRank++ algorithms by taking the obtained context network as input. By experiments, we analyze the characteristics of our proposed method and demonstrate that its accuracy is much better than the state-of-the-art approaches.

INDEX TERMS Web services, parameter correlation, composition context, services similarity.

I. INTRODUCTION

With the development of Internet, IoT (Internet of Things) [1] and cloud computing [2], Web services are becoming an ideal standard technology for data sharing. Since an invalid Web service may lead to the failure of software systems that are built on top of it, it is necessary to replace an unavailable service by another one with the same functionality to ensure smooth operation of the software system. To this end, we need to find a service which has the equivalent or similar function with the invalid one. One of the most fundamental problems is to measure the similarity between two Web services. Web services similarity measure is an important problem in service computing field, and it is the basis of other techniques, such as service replacement, service discovery, service recommendation and service composition [3].

Since a Web service has its own description document of functions and interfaces, such as WSDL (Web Services Description Language), most existing studies measure the similarity between services based on the syntactic [4], [5] and semantic description [6]. However, only relying on static description of a service cannot satisfy the requirement of services similarity measure in real-life applications. First, service descriptions are usually provided by service providers, and the actual functions of a service are not always consistent with the description. For instance, some services provide multiple input and output parameters, but the most frequently used parameters may only be part of these parameters described by the service. Moreover, descriptions of some services are automatically generated through source codes. Therefore, the descriptions cannot reflect actual functions of

The associate editor coordinating the review of this manuscript and approving it for publication was Wajahat Ali Khan.

services. In addition, for some reasons like service upgrade, services functions are not always in accordance with their descriptions. Secondly, given two Web services, even if their descriptions are similar, they cannot always be used to replace each other. For example, a Web service providing queries for phone numbers in China cannot replace the service providing the same query function in US, even though they have the same descriptions.

Most of existing methods measure services similarity only based on static descriptions of services, and totally ignore dynamic features of services. However, dynamic information of services, including their neighbors and interaction with other services in existing compositions, can exactly reflect their actual functions.

To solve the above problem, we explore a practical method to measure the similarity between Web services from the perspective of dynamic characteristics of services, which is different from existing works based on static syntactic or semantic descriptions of Web services. Specifically, we take a Web service's composition context as the basis of services similarity measure in this work. For a service (an operation of a Web service is called a service in this paper), we regard its neighbors and interactive information with other services in existing compositions as its composition context. We extract the composition context of each service from existing compositions (including a service composition or a service Mashup, hereinafter referred to as a composition), and propose a method to compute the similarity between two Web services based on service composition contexts. The main contributions are as follows. (1) We propose a service composition context model based on correlations between a service input parameter and output parameter. For a service, its composition context is comprised of its neighbor services, which have parameters correlations with it in existing compositions. (2) We construct the Web service context network by all services and their composition contexts, and propose to calculate the similarity between any two services in the composition context network. (3) Through a group of experiments, we analyze characteristics of the proposed method, and compare it with related approaches.

The remainder of this paper is organized as follows. We introduce related works in Section 2. Section 3 illustrates the Web services similarity measure framework based on service composition contexts. Section 4 presents our service composition context model, and Section 5 introduces the approach to compute services similarity. In Section 6, we report experimental results. Finally, Section 7 concludes our work and outlines the future work.

II. RELATED WORKS

In the past decade, there have been a lot of research works focusing on Web services similarity measure. These works mainly use static descriptions of services, such as operations, messages, and descriptions. In general, there are three types of works for Web services similarity measure based on static descriptions of services. First, there are many works to measure services similarity based on services' static syntactic description, such as operations, input and output messages, textual description of its function, such as [7]–[9]. In addition, some works consider the structure of input and output messages of services [10]–[12]. Second, many works, such as [13]–[15], use the ontology-based semantic technology to annotate Web services, such as OWL-S and WSMO [16]. Then, services similarity is measured using the similarity of semantic concepts. Third, some works combine above two approaches to measure services similarity. That is, both the syntactic and semantic descriptions of services are taken into account to measure the similarity of services [17]–[19].

However, there still exist some problems for Web services similarity measure only taking static service descriptions into account. First, as mentioned in the first part, two services may not be able replace each other even if their static descriptions are similar. Second, the fundamental ontology suitable for all Web services is hard to construct under the environment of Internet. As a consequence, it is hard to annotate Web services with ontology [13], [20]. In reality, most services are not annotated with ontology, which limits practical applications of approaches using semantics.

Different from above methods, some researchers believe that inherent essential behaviors of services need be taken into account to measure services similarity. Therefore, they propose to measure services similarity based on dynamic behaviors of services [21]–[24]. These works mainly use theoretical or formal tools such as process algebra and calculus to consider dynamic behaviors of services, and measure services similarity from the perspective of Web services behaviors consistency. These methods improve the accuracy of services similarity measure by using complete formal tools. However, these methods need to model Web services and their behaviors with formal tools beforehand. Furthermore, the verification complexity is very high. In practical applications with many Web services, the availability of these methods is very limited. Meanwhile, it is difficult to achieve full automation of verification [25].

There are some research works, in which core ideas are similar to our work. References [26]–[28] propose services similarity measure based on composition contexts. Through the context of a Web service in a composition, services with similar composition context can be found from existing compositions. From the view of a context model, the composition context of a service in these works focus on services that have control flow dependencies. Specifically, for a service, its composition context can be modeled as a directed graph. In the graph, a vertex represents a service that have control flow dependency with it, and an directed edge represents a control flow between two services, which is an ordered sequence of multiple atomic patterns (Sequence, AND-fork, AND-join, OR-fork, XOR-fork, OR-join). To calculate the similarity of two service contexts, both of them are regarded as the starting of the directed graph of their contexts, respectively. Then, through same neighbor services having equal path length to these two services, the distance between two

control flow with the same starting and ending services is calculated through the editing distance. Next, the similarity of atom patterns on the edges is calculated using the distance of control flows. Finally, the similarity of two service composition contexts is obtained by weighting similarity values obtained through neighbor services on different layers. In addition, Wang extends the method in [26], and takes into account not only actual input and output parameters used in a composition, but also dependency relationships between input and output parameters [29]. In addition, a BPEL (Business Process Execution Language)-based service composition process is transformed into a colored Petri net. Then, a method to extract the service composition context from a colored Petri net model of a service composition process is proposed. From the view of a service composition context model, this work still describes a composition context based on control flow dependencies, and calculates contexts similarity in the same way as [26].

Although above works study services similarity measure based on service composition contexts, there are still some problems to solve. First, from the view of a composition context model, only the control flow relationship between services is considered. However, for Web services, correlations between their input and output parameters, which reflect data flow relationship between services, is more import to express services dependence and interaction in existing compositions. Therefore, for a Web service, services that have parameter correlations with it should be part of the service composition context. Second, from the perspective of service composition context construction to measure the similarity, the context of a service in the above works is obtained through one composition of this service. However, a service can collaborate with different services in multiple compositions in different ways. Accordingly, contexts collected from multiple compositions should be taken as the overall composition context of a service. Third, from the perspective of context-based similarity computation, above works only choose exactly the same services from the composition context as the basis of similarity measure. However, if services in the composition contexts of two services are completely different but highly similar, the two services should also have a high similarity. For the similarity measure of service composition contexts, above works do not consider the similarity between services in the context. The categories of all above methods and their main ideas of Web services similarity measure are listed in Table 1.

TABLE 1. Existing web services similarity measure methods.

Method Category	Method Subcategory	Similarity Measure Idea
Using Web services static descriptions	Using syntactic descriptions of services	Using static syntactic descriptions, such as operations, input output messages, textual description of the function.
	Using semantic descriptions of services	Using static semantic descriptions annotated with ontology such as OWL-S and WSMO.
	Using both syntactic and semantic descriptions of services	Using both syntactic and semantic descriptions of services to measure the similarity of Web services.
Using Web services dynamic features	Using dynamic behaviors of services and formal theories	Using theoretical or formal tools, such as process algebra and calculus to measure services similarity from the view of dynamic behaviors of services.
	Using service composition contexts	Using Web services composition contexts obtained from existing services compositions.

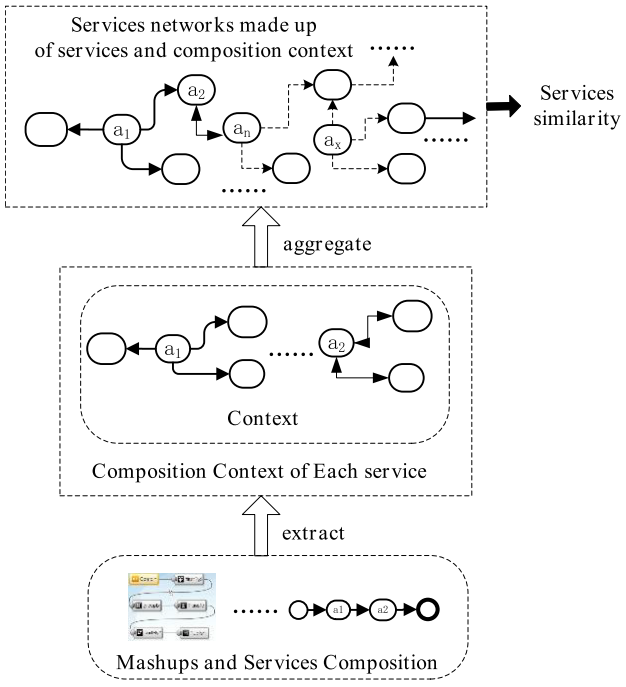

FIGURE 1. The Framework of the web services similarity measure based on the web services composition context.

III. COMPOSITION CONTEXT-BASED WEB SERVICES SIMILARITY MEASURE FRAMEWORK

In this work, we describe Web services composition context from the view of parameter correlations between services. Fig. 1 shows the Web services similarity measure framework based on service composition context proposed in this work. First, for each service, services that have parameters correlations with it are obtained through existing service compositions, and the parameter correlations between them are extracted to derive the composition context of the service. Then, we take services as vertices, and take their parameter correlations as edges. In this way, we can generate a graphical composition context for each service. Next, after obtaining the composition context of each service, all services and their contexts constitute a services composition context networks. Finally, the similarity between any two arbitrary services is calculated based on the context network.

As indicated in the framework of Fig. 1, we first present the Web service composition context based on parameters correlations. Then, we introduce the services similarity measure approach based on the composition context in detail.

IV. WEB SERVICE COMPOSITION CONTEXT

Parameter correlations typically exist between input and output parameters of multiple services in a service composition. As mentioned above, for a service, its correlations with other services can be extracted from existing compositions that contain this service [30]. Meanwhile, by extracting multiple correlations from different compositions, we can get the parameter correlations between this service and other services. Next, we give the definition of a parameter correlation between services first.

A. WEB SERVICE PARAMETER CORRELATION

Reference [31] introduces six parameter correlations, in which five types express the correlation relationship between service parameters. Generally speaking, let s and t be two services, there are two kinds of parameter correlations between them: 1) a correlation between an output parameter p_s of s and an input parameter p_t of t. The correlation expresses the one-to-one dependency between p_t and p_s. That is, when s and t are composed, the value of p_s can be assigned to p_t, and 2) the one-to-one correlation between an output parameter p_s of s and an output parameter p_t of t. In this correlation, p_s and p_t represent semantically similar attributes. When s and t are composed, composition operations to aggregate their outputs, such as *merging* and *natural joining*, can be performed based on the output of s and t [31].

Our previous work shows there are two types of parameter correlations between Web services, namely, OI (Output-Input) and OO (Output-Output) [30] correlation. They are essentially above two correlations. An OI correlation between two services expresses their composability, and an OO parameter correlation expresses the similarity relation between their output parameters. In this paper, we introduce the third parameter correlation between two service s and t, which is similar to an OO parameter correlation: II (Input-Input) parameter correlation. An II correlation expresses that an attribute value can be used as an input parameter of both s and t. An II parameter correlation essentially describes some similarity relation between p_s and p_t. In summary, there are three types of parameter correlations between two services. We denote them as OI, OO and II, respectively.

Definition 1 (Parameter Correlation, pc): Let s and t be two Web services. $\boldsymbol{pc} = (p_s, p_t, \boldsymbol{m})$ is a **parameter correlation** between s and t, where p_s and p_t are an input or output parameter of s and t, respectively, and m is a one-to-one correlation between p_s and p_t.

In definition 1, we call the service s as the ***source service***, and the service t as the ***target service*** for convenience.

Definition 2 (oopc, iipc, oipc): Let s and t be two Web services. For a *pc* between s and t: 1) let p_s and p_t be an output parameter of s and t. In an existing composition, for the output of s and t, through the values of p_s and p_t, if we can obtain their union set by the *Union* operation, or obtain their intersection set by the *Intersection* operation, or obtain their difference set by the *Difference* operation, or connect them by the *Join* operation [32], we call *pc* an **OO Parameter Correlation** (***oopc***); 2) let p_s and p_t be an input parameter of s and t, if p_s and p_t receive the same value as the input in an existing composition, we call *pc* an **II Parameter Correlation**(***iipc***); and 3) let p_s and p_t be an output parameter of s and an input parameter of t, respectively. If the value of p_s is used as the input of p_t in an existing composition, we call *pc* an **OI Parameter Correlation**(***oipc***).

In Definition 2, the semantic relationship between two parameters in a *pc* can be one of *equivalence*, *subset*, *superset*, and *overlapping* [33]. For two Web services s and t, there may be multiple *oopcs*, *iipcs* or *oipcs* between them. Note that *oopc* and *iipc* are bidirectional correlations, whereas *oipc* is a unidirectional correlation from s to t. We denote the set of *oopc*, *iipc* and *oipc* between s and t as $\mathbf{OOPC_{s<->t}}$ $\mathbf{IIPC_{s<->t}}$ and $\mathbf{OIPC_{s->t}}$, respectively. We show some examples of three correlations in Fig. 2. $\mathrm{OOPC_{searchPaper<->searchPaperByTitle}}$ contains three *oopc* s, and others contain one *pc*. In $\mathrm{IIPC_{searchBook<->searchResearcher}}$, the input parameter *name* of *searchBook* and the input parameter *researcher* of *searchResearcher* in $iipc_1$ are semantically equivalent parameters, although their names are different. In $\mathrm{IIPC_{searchPaperByTitle<->BaiduSearch}}$, *searchPaperByTitle*'s input parameter *title* and *BaiduSearch*'s input parameter *keywords* in $iipc_2$ can be assigned the same value. Furthermore, the value range of *title* is the subset of that of *keywords*. In $\mathrm{IIPC_{SearchResearcher<->getTeacherInfo}}$, the semantic relationship between the former input parameter *researcher* and the latter input parameter *Tea_name* in *iipc*4 is *overlapping*. This is because some input values can be shared by these two input parameters, whereas some input values can only be assigned to *researcher* or *Tea_name*.

B. PARAMETER CORRELATIONS-BASED WEB SERVICE COMPOSITION CONTEXT

Reference [34] defines context as information that can be used to characterize the situation of an entity, which is a person, place, or object considered relevant to the interaction. Based on this definition, we can define Web services composition context from the perspective of *pc*. For a Web service, we define its composition context as services having *pc*s with it and the corresponding *pc*s among them. In this definition, given a Web service, services that have *pc*s with it characterize the situation of this service, and *pc*s among them describe their interaction in existing service compositions. Since there are three types of *pc*s, including *oopc*, *iipc*, *oipc*, we first present the definitions of Web service composition context based on *oopc*, *iipc* and *oipc*. Next, we present a complete definition of Web service composition context.

Definition 3 (Composition Context Based on OOPC): Let s be a service, its service **composition context based on**

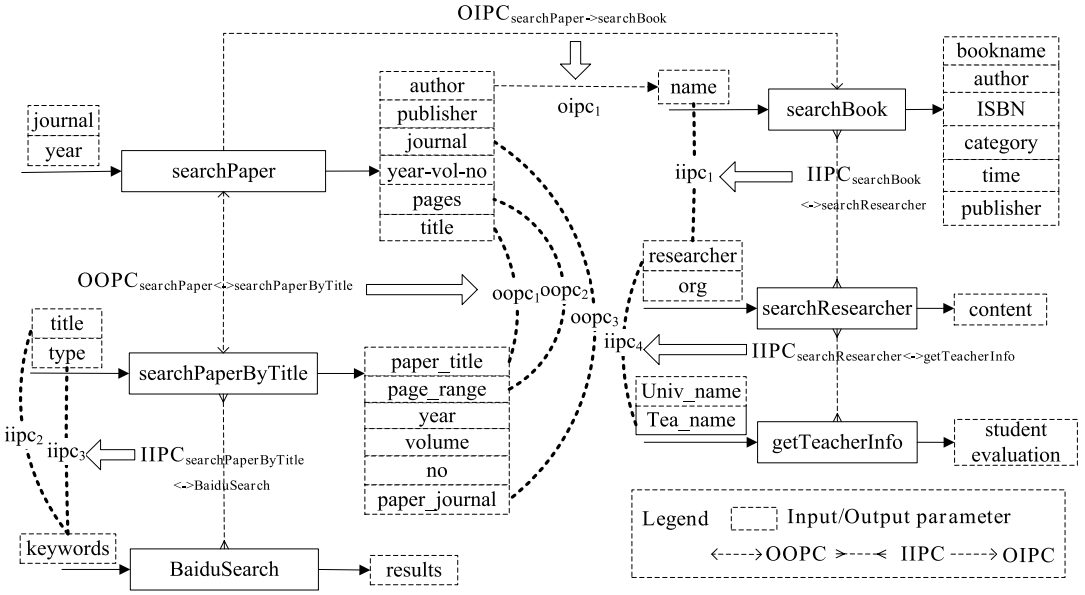

FIGURE 2. Examples of three parameter correlations of web Services.

OOPC is **CCPC-OO$_s$** = {< t_1, $OOPC_{s<->t1}$, w_{oo_1} >, < t_2, $OOPC_{s<->t2}$, w_{oo_2} > ... < t_n, $OOPC_{s<->tn}$, w_{oo_n} >} where: < t_i, $OOPC_{s<->ti}$, w_{oo_i} > ($1 \le i \le n$) is $OOPC_{s<->ti}$ between s and service t_i with the weight w_{oo_i}. $OOPC_{s<->ti}$ ={pc_1, $pc_2 \ldots pc_m$} where: pc_j($1 \le j \le m$) is an *oopc* between *s* and t_i.

The weight w_{oo_i} of $OOPC_{s->ti}$ in Definition 3 express the correlation strength of $OOPC_{s->ti}$. Let out_s and out_{ti} be the sets of output parameters of *s* and t_i. Equation (1) gives the computation method of w_{oo_i}. We can see the value range of w_{oo_i} is (0,1]. Take $OOPC_{searchPaper<->searchPaperByTitle}$ in Fig. 2 as an example, its weight is 2×3/(6+6)=0.5.

$$w_{oo_i} = \min\left(\frac{2 * |pc_1.p_s \cup \ldots pc_m.p_s|}{|out_s| + |out_{t_i}|}, 1\right) \quad (1)$$

Definition 4 (Composition Context Based on IIPC): Let *s* be a service, its **service composition context based on IIPC** is **CCPC-II$_s$** = {< t_1, $IIPC_{s<->t1}$, w_{ii_1} >, < t_2, $IIPC_{s<->t2}$, w_{ii_2} > ... < t_n, $IIPC_{s<->tn}$, w_{ii_n} >} where: < t_j, $IIPC_{s<->tj}$, w_{ii_j} > ($1 \le j \le n$) is $IIPC_{s<->tj}$ between s and service t_j with the weight w_{ii_j}. $IIPC_{s<->tj}$ = {pc_1, $pc_2 \ldots pc_m$}, pc_k($1 \le k \le m$) where: pc_k($1 \le k \le m$) is an *iipc* between *s* and t_j.

The weight w_{ii_j} of $IIPC_{s<->tj}$ in Definition 4 express the correlation strength of $IIPC_{s<->tj}$. Let in_s and in_{tj} be the sets of input parameters of *s* and t_j. Equation (2) gives the computation method of w_{ii_j}. The value range of w_{ii_j} is (0,1]. Take $IIPC_{searchPaperByTitle->BaiduSearch}$ in Fig. 2 as an example, its weight is 2×2/(2+1)=1.33>1. Therefore, w_{ii_j} =1.

$$w_{ii\_j} = \min\left(\frac{2 * |pc_1.p_s \cup \ldots pc_m.p_s|}{|in_s| + |in_{t_j}|}, 1\right) \quad (2)$$

Definition 5 (Composition Context Based on OIPC): Let *s* be a service, its **service composition context based on OIPC** is **CCPC-OI$_s$** = {< t_1, $OIPC_{s->t1}$, w_{oi_1} >, < t_2, $OIPC_{s->t2}$, w_{oi_2} >... < t_n, $OIPC_{s->tn}$, w_{oi_n} >} where: < t_j, $OIPC_{s->tj}$, w_{oi_j} > ($1 \le j \le n$) denotes $OIPC_{s->tj}$ from *s* to service t_j with the weight w_{oi_j}. $OIPC_{s->tj}$ = {pc_1, $pc_2 \ldots pc_m$}, pc_k($1 \le k \le m$) where: pc_k($1 \le k \le m$) is an *oipc* between *s* and t_j.

The weight w_{oi_j} of $OIPC_{s->tj}$ in Definition 5 express the correlation strength of $OIPC_{s->tj}$. Let out_s and in_{tj} be the sets of output parameters of *s* and input parameters of t_j. Equation (3) gives the computation method of w_{oi_j}. The value range of w_{oi_j} is (0,1]. Take $OIPC_{searchPaper->searchBook}$ in Fig. 2 as an example, its weight is $1 \times 2/(6+1) = 0.29$.

$$w_{oi\_j} = \min\left(\frac{2 * |pc_1.p_s \cup \ldots pc_m.p_s|}{|in_{t_j}| + |out_s|}, 1\right) \quad (3)$$

As mentioned above, *pc*s are derived from existing service compositions through the composition way of services [30]. Consequently, it should be noted that the accurate semantic relationship between two parameters in a *pc* cannot be obtained in this way. Accordingly, we do not consider semantic relationship between parameters in the above weight computation equations.

Based on CCPC-OO, CCPC-II and CCPC-OI, we give the complete definition of Web service composition context.

Define 6 (Composition Context): Let *s* be a service, its **service composition context** is **CC$_s$** ={CCPC-OO$_s$, CCPC-II$_s$, CCPC-OI$_s$} where: CCPC-OO$_s$, CCPC-II$_s$, and CCPC-OI$_s$ are composition contexts based on OOPC, IIPC, and OIPC, respectively.

Fig. 3 shows the composition context of the two services *searchPaper* and *searchBook* in Fig. 2. As mentioned above,

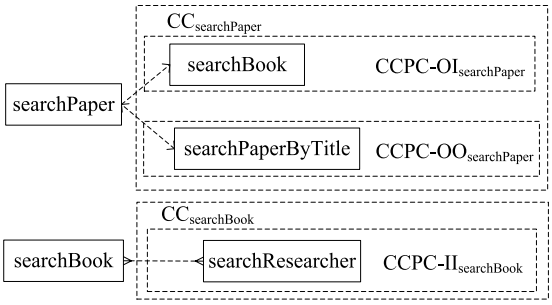

FIGURE 3. Examples of web services composition context.

a Web service may be used in different compositions, and multiple composition contexts of this service can be extracted and merged. Finally, the composition context of this service can be obtained. Therefore, the composition context of a service evolves with the accumulation of existing compositions. Due to space limitation, we do not introduce the generation and merging of service composition contexts in this paper.

V. COMPOSITION CONTEXT-BASED WEB SERVICES SIMILARITY MEASURE APPROACH

If we take a Web service and services in its composition context as vertices, and take *pc*s in the context as edges, the composition context of the service can be described as a hybrid graph, which consists of three subgraphs: (1) the weighted undirected graph based on OOPC, in which each undirected edge represents an OOPC between the service and services in its composition context, and the value on the edge is the weight of the OOPC; (2) the weighted undirected graph based on IIPC, in which each undirected edge represents an IIPC between the service and services in its composition context, and the value on the edge is the weight of the IIPC; and (3) the weighted directed graph based on OIPC, in which each directed edge represents an OIPC between the service and services in its composition context, and the value on the edge is the weight of the OIPC.

The composition context hybrid graph of all services can be merged into a **Service Composition Context Networks (SCCN)**. SCCN consists of three subnetworks: **OOPC Context Networks (OOPCCN)**, **IIPC Context Networks (IIPCCN)**, and **OIPC Context Networks (OIPCCN)**. Next, we introduce the approach to compute services similarity based on SCNN in detail.

A. BASIC IDEAS

Firstly, an edge in OOPCCN expresses some similarity relation between output parameters of the two services linked by this edge, according to Definition 2. Therefore, for a service, services with similar output parameters can be obtained directly through OOPCs of each service in OOPCCN. In addition, because *pc*s are extracted from existing compositions, the scale of OOPCCN is affected by some factors, such as the number of compositions, service usage and so on. As a consequence, in order to measure services similarity more

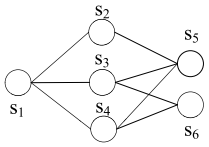

FIGURE 4. An example of OOPCCN.

comprehensively, besides services directly correlated with a service, we take services indirectly correlated in OOPCCN into account. Take OOPCCN in Fig. 4 as an example, for service s_1, the similarity between s_1 and s_2, s_3, and s_4 can be calculated through the directly correlated service s_2, s_3 and s_4. Moreover, because s_2, s_3, and s_4 are all correlated with s_5 by an OOPC, s_1 may have some similarity with the output parameters of s_5. In short, we can obtain the similarity between indirectly related services through OOPCCN.

Specifically, for two services in OOPCCN, if there are more connection paths between them, or the lengths of paths between them are shorter, or the connection paths pass through more outgoing service vertices, they should have higher similarity based on the output parameters. Briefly, we need to calculate the similarity between any two services according to the structure of OOPCCN. In graph-based link analysis algorithms, PersonalRank algorithm [35] implements personalized recommendation based on the TopicRank algorithm [36]. The core idea of PersonalRank is to obtain the relevancy between any two nodes in the graph according to the graph structure. Using this algorithm, we can obtain the relevancy between any two services in OOPCCN. Because an edge in OOPCCN expresses the similarity relation between service output parameters, we can regard the relevancy value as the similarity value between services output parameters.

Similarly, an edge of IIPCCN indicates some similarity relation of input parameters between two services. Using PersonalRank algorithm, we can get the similarity between any two services based on input parameters in IIPCCN.

Secondly, if both service s_1 and s_2 have an OIPC with s_3 in OIPCCN, it means that some output parameters of s_1 and s_2 have similarity relation with some input parameters of s_3. Accordingly, we can infer that the output parameters of s_1 and s_2 may have some similarity relation. Briefly, the similarity between s_1 and s_2 can be calculated according to the similarity between CCPC-OI_{s1} and CCPC-OI_{s2}. As mentioned above, when calculating the similarity between CCPC-OI_{s1} and CCPC-OI_{s2}, we should consider not only the same services in the two contexts, but also the similarity of services in them. Meanwhile, the services similarity in CCPC-OI_{s1} and CCPC-OI_{s2} needs to be calculated through the CCPC-OI of these services in turn. Hence, we can see it is an iterative process, and the similarity between services in the network needs to be calculated iteratively according to the structure of OIPCCN. An important method in calculating object similarity is the method based on links between objects, in which a representative algorithm is

SimRank [37]. The basic idea of the SimRank algorithm is that if the objects associated with two objects are similar, then the two objects are similar. The essence of the algorithm is to use the graph structure to calculate the similarity between any two nodes in the graph. We can see the core idea of SimRank is consistent with that of services similarity measure based on OIPCCN in this work. Therefore, we use the SimRank related algorithm to compute the services similarity in OIPCCN.

Next, we present the services similarity measure approach based on OOPCCN, IIPCCN and OIPCCN in detail.

B. OOPCCN AND IIPCCN-BASED WEB SERVICES SIMILARITY MEASURE

Let s and t be two services and $\mathbf{sim_{OO}}\ s\ t$ be the similarity between s and t in OOPCCN. Equation (4) gives the method to compute $\mathrm{sim_{OO}}(s,\ t)$ based on PersonalRank algorithm, where C_1 is the damping coefficient between [0, 1], out(t') is the set of OOPCs in CCPC-$\mathrm{OO_{t'}}$. Equation (5) gives the method to compute the similarity between service s and t in IIPCCN, which is denoted as $\mathbf{sim_{II}}\ s\ t$.

$$sim_{OO}^{k+1}(\mathrm{s,\ t}) = \begin{cases} C_1 \sum\limits_{t' \in out(t)} \dfrac{\mathrm{w_{s\to t'}} \times sim_{OO}^{k}(s, t')}{|out(t')|} & t \neq s \\ (1 - C_1) + C_1 \sum\limits_{t' \in out(t)} \dfrac{\mathrm{w_{s\to t'}} \times sim_{OO}^{k}(s, t')}{|out(t')|} & t = s \end{cases} \tag{4}$$

$$sim_{II}^{k+1}(\mathrm{s,\ t}) = \begin{cases} C_1 \sum\limits_{t' \in out(t)} \dfrac{\mathrm{w_{s\to t'}} \times sim_{II}^{k}(s, t')}{|out(t')|} & t \neq s \\ (1 - C_1) + C_1 \sum\limits_{t' \in out(t)} \dfrac{\mathrm{w_{s\to t'}} \times sim_{II}^{k}(s, t')}{|out(t')|} & t = s \end{cases} \tag{5}$$

Equations (4)-(5) are all iteration formulas. Take (4) as an example. Initially, if $t = s$, then $\mathrm{sim_{OO}}(s,\ t) = 1$; otherwise, $\mathrm{sim_{OO}}(s, t) = 0$. Next, by iteration, when the value of $\mathrm{sim_{OO}}(s,\ t)$ converges, the final $\mathrm{sim_{OO}}(s,\ t)$ is obtained. Note that $\mathrm{sim_{OO}}(s,\ t)$ calculates the similarity between s and t by the probability of moving from s to t, whereas the probability is different from that of moving from t to s. Therefore, the value of $\mathrm{sim_{OO}}(s,\ t)$ and $\mathrm{sim_{OO}}(t,\ s)$ directly obtained by (4) is not equal. However, from the view of services similarity, $\mathrm{sim_{OO}}(s,\ t)$ should be equal to $\mathrm{sim_{OO}}(t,\ s)$. Therefore, the final $\mathrm{sim_{OO}}(s,\ t)$ and $\mathrm{sim_{OO}}(t,\ s)$ are $(sim_{OO}^{n}(s, \mathrm{t}) + sim_{OO}^{m}(\mathrm{t,\ s}))/2$, where n and m represent the last iteration. Similarly, $\mathrm{sim_{II}}(s,\ t)$ and $\mathrm{sim_{II}}(t,\ s)$ are $(sim_{II}^{n}(\mathrm{s,\ t}) + sim_{II}^{m}(\mathrm{t,\ s}))/2$.

C. OIPCCN-BASED WEB SERVICES SIMILARITY MEASURE

Let s and t be two services and $\mathbf{sim_{OI}}\ (s, t)$ be the similarity between s and t in OIPCCN. Because OIPCCN is a weighted directed graph, we need to solve the following two problems to compute $\mathrm{sim_{OI}}(s,\ t)$ by SimRank algorithm.

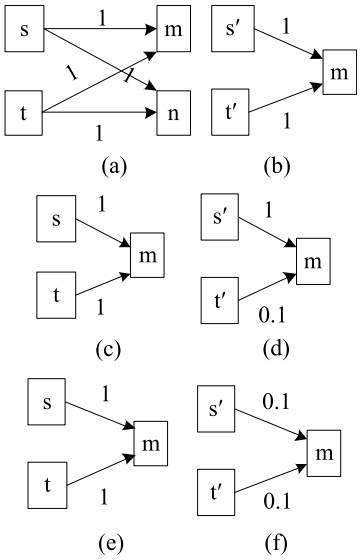

FIGURE 5. Examples of web services and OIPCs.

(1) In case that the weights of OIPCs that s and t own are all equal, the more common neighbor services they correlate, the higher their similarity is. We take (a) (b) in Fig. 5 as an example, where weights of OIPCs are the same. Services linked by s and t are m and n, whereas the service linked by s' and t' is only m, then $\mathrm{sim_{OI}}(s,\ t) > \mathrm{sim_{OI}}(s\prime,\ t\prime)$.

(2) For the same service that linked by s and t, the smaller the variance of their OIPC weights is, the higher their similarity is. Moreover, in the case that the variances of OIPC weights are identical, the similarity between services with higher weight is larger. Take (c) (d) in Fig. 5 as an example, s and t, as well as s' and t', are all linked with the service m by OIPCs. However, the variance of $\mathrm{w}_{s->m}$ and $\mathrm{w}_{t->m}$ is zero, whereas the variance of $\mathrm{w}_{s\prime->m}$ and $\mathrm{w}_{t\prime->m}$ is higher than zero, then $\mathrm{sim_{OI}}(s,\ t) > \mathrm{sim_{OI}}(s\prime,\ t\prime)$. For (e) (f) in Fig. 5, the variance of $\mathrm{w}_{s->m}$ and $\mathrm{w}_{t->m}$, as well as the variance of $\mathrm{w}_{s\prime->m}$ and $\mathrm{w}_{t\prime->m}$, are all zero, and $\mathrm{w}_{s->m} = \mathrm{w}_{t->m} = 1$, whereas $\mathrm{w}_{s\prime->m} = \mathrm{w}_{t\prime->m} = 0.1$, then $\mathrm{sim_{OI}}(s, t) > \mathrm{sim_{OI}}(s\prime,\ t\prime)$∘

SimRank ++ algorithm [38] meets above requirements. Accordingly, we propose to calculate the similarity between any two services in OIPCCN based on the SimRank ++ algorithm. Let s be a service, In(s) and Out(s) be OIPC sets of s in OIPCCN, in which each OOPC takes s as the target and source, respectively. The similarity between any two services s and t in OIPCCN can be calculated by (6).

$$sim_{OI}^{k+1}(\mathrm{s,\ t}) = \begin{cases} C_2 \cdot \mathrm{evidence}(\mathrm{s,\ t}) \cdot \sum\limits_{i \in Out(s)} \sum\limits_{j \in Out(t)} W(s, i) \\ \quad \cdot W(t, j) \cdot sim_{OI}^{k}(i, j) & s \neq t \\ 1 & s = t \end{cases} \tag{6}$$

where:

$$\text{evidence (s, t)} = \sum_{i=1}^{|In(s)\cap In(t)|} \frac{1}{2^i}$$

$$\text{W (s, i)} = e^{-variance(i)} \times \frac{w_{s\to i}}{\sum_{j\in In(s)} w_{s\to j}}$$

Equation (6) is an iterative formula, where evidence(s, t) is an increasing function of the common neighbor services between s and t. Namely, the more common neighbor services s and t have, the higher evidence(s, t) is, and the higher $\text{sim}_{\text{OI}}(s, t)$ is. C_2 is the damping coefficient in the SimRank ++ algorithm, and its value range is [0, 1]; variance(i) is the variance of OIPC weights in In(i). At the beginning of the algorithm, the initial similarity value of any two services in OIPCCN is set as follows. If s = t, then $\text{sim}_{\text{OI}}(s, t) = 1$, otherwise $\text{sim}_{\text{OI}}(s, t) = 0$. Equation (6) calculates $\text{sim}_{\text{OI}}(s, t)$ iteratively, and $\text{sim}_{\text{OI}}(s, t)$ converges after a certain number of iterations. At this time, we obtain the final similarity between any two services in OIPCCN.

D. SCCN-BASED WEB SERVICES SIMILARITY MEASURE

We can obtain the final similarity between any two services based on SCCN by synthesizing three similarity values, which are obtained based on OOPCCN, IIPCCN and OIPCCN. Notice that the similarity based on OOPCCN and IIPCCN can reflect services similarity more obviously than that based on OIPCCN. This is because the similarity between service output and input parameters can be obtained directly by OOPCCN and IIPCCN, whereas the similarity based on OIPCCN is the indirect similarity between service output parameters. Hence, we synthesize three similarity values based on different proportion of three types of *pc*s.

Specifically, let s and t be two services. After obtaining $\text{sim}_{\text{OO}}(s, t)$, $\text{sim}_{\text{II}}(s, t)$, and $\text{sim}_{\text{OI}}(s, t)$, we calculate the final services similarity by (7). In this equation, the value range of λ is [0, 1]. λ determines the proportion of services similarity value calculated by three composition contexts. As a result, λ can affect the final similarity value. In the practical application, we can determine the value of λ according to the ratio of three types of *pc*s.

$$\text{sim (a, b)} = \lambda\, sim_{OI}\,(a, b) + (1-\lambda)\,(sim_{OO}\,(a, b) + sim_{II}\,(a, b))/2 \quad (7)$$

VI. EXPERIMENTS AND EVALUATION

We perform experiments to evaluate the validity of the proposed approach for Web services similarity measure. Firstly, we analyze the effect of the proposed approach by comparing it with other existing approaches. Secondly, we analyze characteristics of our approach. Specifically, we focus on the impact of a few key parameters on the performance of the algorithm, including the scale of *pc*s, λ in (7), and the number of similar services(denoted as N) given by the proposed approach.

Note that Simrank ++ and PersonalRank are all network-based iterative algorithms with high time complexity. Therefore, there are many optimization algorithms to solve this problem [39]–[42]. Meanwhile, the similarity values can be calculated offline in the practical application. In addition, the update of similarity values can be done offline after the evolution of SCCN network structure has accumulated to a certain extent in practice. Therefore, we do not include the performance and scalability evaluation of the proposed approach in this section.

A. DATASETS

For our experiments, a certain number of Web services and their compositions are needed. According to these compositions, we obtain the above three *pc*s and construct SCCN. Finally, we can obtain the similarity between services through the proposed approach.

There are some publicly available data sets including services and their compositions. OpenAPIs and Mashups on ProgrammableWeb[1] is the most commonly used data set, which includes a certain amount of real Web services and Mashups.[2] However, we find that some services and Mashups are unavailable in the data set. Furthermore, in some available Mashups, only URLs of contained services are provided, and we cannot get the composition way of the services. This makes it difficult to obtain *pc*s according to these mashups. In addition, services and Mashups on ProgrammableWeb come from multiple parts of the world and belong to multiple domains. As a result, their *pc*s are sparse. A small number of frequently used services own the majority of correlations with other services, whereas most services have fewer correlations. Thus, the SCCN obtained by the dataset is very sparse. Briefly, the data on ProgrammableWeb cannot meet the requirements of constructing service composition context in this work. Similarly, it is also difficult to obtain required service compositions and *pc*s through other public service datasets.

Considering the above problems of these open datasets, we generate simulated service datasets for the experiments to fully understand the effect and characteristics of the proposed approach. We construct service datasets that conforms to the data characteristics of ProgrammableWeb according to existing works [43]. Specifically, we construct simulated datasets containing different number of services according to service classification, characteristics of service input output parameters and *pc*s. The simulated service datasets are produced using the method in [43], [44] as follows. (1) Given C service categories, the number of services in each category is S. (2) For each service, 2-4 input parameters and 2-7 output parameters are randomly generated. (3) To express the semantic meaning of a randomly generated input or output parameter, a predefined label is added to each input and output parameter randomly. If the input and output parameters of different services are labeled as the same label, it means they can establish an *oopc*, *iipc* or *oipc*. (4) T different labels for each service category are provided, and they are randomly assigned to

[1] http://www.programmableweb.com/
[2] http://blog.csdn.net/shimin520shimin/article/details/51209322

input and output parameters of each service in that category. Meanwhile, the parameter M that indicates the repeatability of a label is set to control the proportion of duplicate labels contained in different service categories. In this way, we can control the number of possible *pc*s between different service categories. Through above parameters, we can control the number of generated services and *pc*s by a program, and generate the corresponding scale of simulated service dataset based on experimental requirements.[3]

B. EVALUATION METHODS

The similarity values obtained by our approach provide the basis for ranking services as choosing similar services for a given service. Therefore, we verify the accuracy of the approach by similar services and their ranking for given services in the experiment.

Specifically, firstly, for a service to match the similarity, we need to obtain N services most similar to it as the baseline. To this end, we need to derive the baseline similarity between any two services first. In the experiment, we calculate the baseline similarity between services by the label of service input and output parameters in the simulated dataset. If a pair of input output parameters of two services have the same label, they are considered to have the same semantic meaning, i.e., they are the same parameters. Next, the similarity between two services can be calculated according to the input and output parameters of two services owning the same labels. Finally, according to the similarity result, the set of N services (denoted as $\mathbf{S_m}$) most similar to a service is obtained. Secondly, we obtain N services most similar to a service using the proposed approach. Then, we obtain the accuracy of our approach by comparing the two groups of services and their orders.

Since the evaluation of similarity accuracy should consider both services given by the approach and their order, we take NDCG (Normalized Discounted Cumulative Gain) as the evaluation indicator of accuracy in the experiment [45]. Specifically, let s be a service and the set of N similar services obtained by our method for s be $\mathbf{S_{sim}}$. The NDCG value of results given by our approach is computed by (8), where: p_i is the ranking of s_i in S_{sim}($s_i \in S_{sim}$), rs_i is the score of s_i in S_m (s_i ranks i in S_m, then rs_i is N-i+1), $IDCG_N$ is the maximum value of the molecular. Obviously, the value range of $NDCG_N$ is [0, 1].

$$\mathrm{NDCG_N} = \frac{\sum_{i=1}^{n} \frac{2^{rs_i}-1}{log_2(1+p_i)}}{IDCG_N} \tag{8}$$

In the experiment, through our approach or the compared approach, N similar services are obtained for each service in the dataset. Then, the NDCG value of the approach is calculated by comparing the N services with the services in the baseline. Finally, the average NDCG is calculated for all services in the dataset, which is taken as the final NDCG of the approach on the dataset.

[3]https://pan.baidu.com/s/18bN1rHNgrLWrmW1Fph0M7w

C. RESULTS

The first experiment compares our approach with that proposed in [28], which is most relevant to our work. Reference [28] calculates the similarity of service composition contexts based on the control flow between services, in which the proposed composition context is different from ours. In order to compare two approaches, we first use the approach in [28] to measure the similarity of services. Specifically, for a service, services having OOPL, IIPL and OIPL with it are regarded as its context, and only the same services in the context are used to calculate the context similarity. In the experiment, given a service, two cases of the method in [28] are considered: services in the first layer and in the first three layers are chosen to compute the services similarity, respectively. For convenience, we denote the two cases and our approach as **FLC1** (services in the first layer of the context), **FLC3**(services in the first three layers of the context) and **PCC**, respectively. In addition, since the proportion of *oipl*s in a dataset has an effect on the final similarity results, the proportion of *oipl*s (the number of *oipl* s/the number of all *pc*s) of the dataset in the first experiment is set to 0.3.

Except comparing with the FLC1 and FLC3, we evaluate the influence of two parameters on the proposed method in the first experiment. The parameters include the scale of services and their *pc*s, and the number of similar services (N). We use two different simulated datasets, and parameters correlations between services are found automatically through a program. The two datasets are: (1) the dataset including 1000 services with different correlation ratios, which are computed by taking the total number of *pc*s divided by the total number of services (denoted as **pl/s**); and (2) the dataset including different number of services (100, 200, 400, 1000, 3000) and different pl/s. We perform the experiment on above two simulated datasets, and obtain the results shown in Fig. 6. Note that the results of FLC1 and FLC3 are quite close. Therefore, we show the comparison of our approach with FLC1 and FLC3 on separate figures to show the results clearly. According to the results, we can draw the following conclusions: (1) the NDCG of our method is much higher than that of both FLC1 and FLC3; (2) when $N = 1$, the effect of this method is the worst, and when $N > 3$, the effect of this method is improved and relatively stable; and (3) since *pc*s are obtained through semantic labels, we can get all the correlations in the dataset by the program. Hence, it is the case that *pc* s are fully obtained. In this case, the effect of our method is worse when pl/s is low, and the effect is improved with the increase of pl/s. After pl/s reaches a certain degree, the NDCG of our method begins to decline.

In the second experiment we analyze the number of *pc*s in SCCN on the effect of our algorithm. We construct four simulated data sets containing 200, 400, 800 and 1600 services, respectively. In each dataset, we delete some *pc*s randomly to make pl/s be 0.3, 0.5, 0.7, 0.9 and 1.1, respectively. In case pl/s is 1.1, *pc*s are not deleted, which simulates the case that all the *oopl*s, *iipl*s and *oipl*s are obtained. For other values of pl/s, different proportions of *pc*s are deleted, which simulate

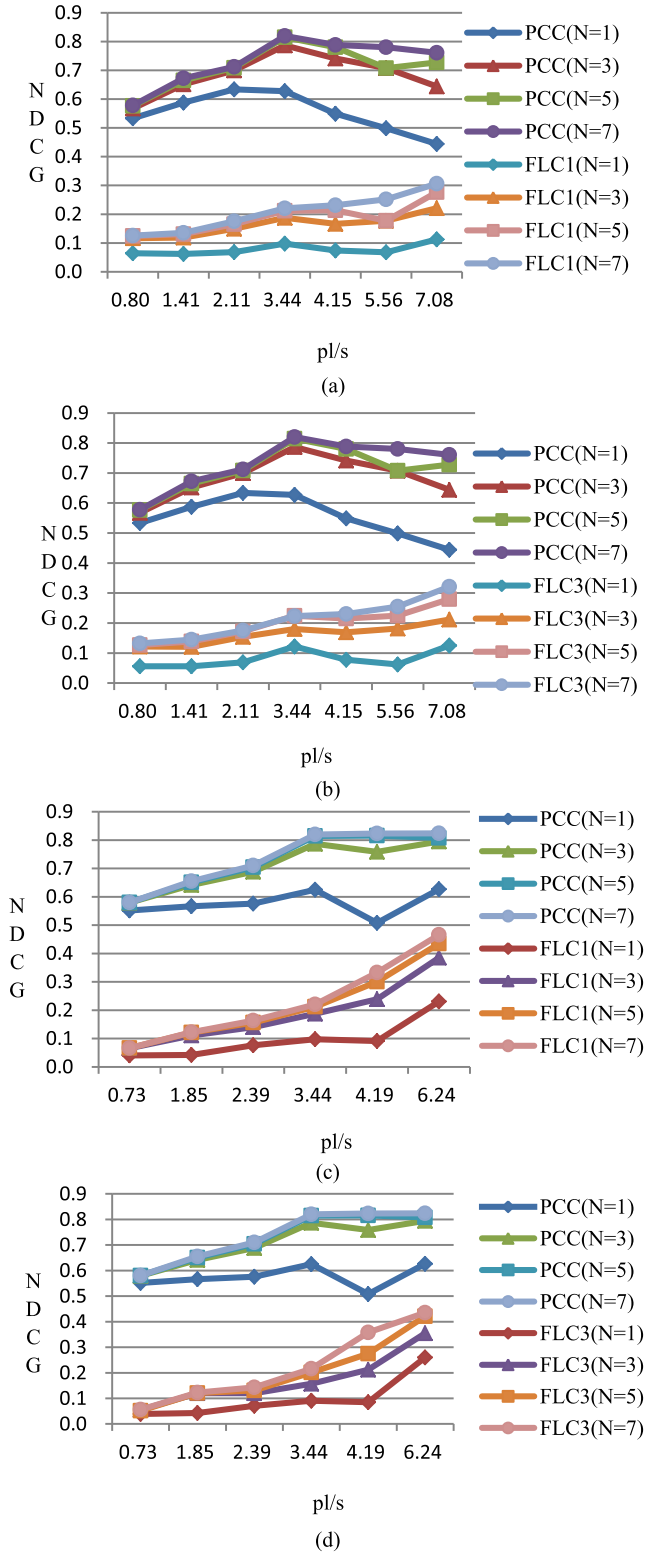

FIGURE 6. Methods comparison and the influence of N. (a) Methods comparison (PCC and FLC1) and the influence of N (the number of services is 1000). (b) Methods comparison (PCC and FLC3) and the influence of N (the number of services is 1000). (c) Methods comparison (PCC and FLC1) and the influence of N (the number of services is 100, 200, 400, 1000, 3000). (d) Methods comparison (PCC and FLC3) and the influence of N (the number of services is 100, 200, 400, 1000, 3000).

the case that we obtain incomplete *pc*s. We obtain the average NDCG values of these four datasets obtained by the proposed approach under different pl/s.

Experimental results are shown in Fig. 7. We can see that the NDCG value increases with the rising of pl/s in SCCN when *pc*s in the dataset are not fully obtained. As a consequence, under the condition that *pc*s are not fully discovered, the more correlations are obtained, the better the performance of the proposed approach is. In the case that *pc*s are found completely, the effect of the proposed approach is the best.

In the third experiment we evaluate the influence of λ in (7). We perform the experiment on different simulated datasets to obtain the NDCG values of the most similar three ($N = 3$) services given by the algorithm under conditions that $\lambda = 0, 0.1, 0.3, 0.5, 0.7, 0.9$, and 1. Since the number of *pc*s in three composition contexts affects the similarity, we construct four copies for each dataset to evaluate the influence of λ under different *oipl* ratios. By deleting a certain number of *iipl*s and *oopl*s, the proportion of *oipl*s can be set at 0.3, 0.5, 0.7 and 0.9 respectively. We observe the influence of λ on the similarity accuracy under different *oipl* ratios (denoted as ***poi***).

We take for example the simulated dataset with 1000 services ($T = 1000$, $M = 0.01$, $C = 40$, $S = 25$, average *oipl*s $= 447$, average *oopl*s $= 690$, average *iipl*s $= 300$). Experimental results are shown in Fig. 8. We can see that the higher *poi* is, which corresponds to the case that less *iipl*s and *oopl*s are obtained, the lower the overall NDCG is. Moreover, we can see the following results. (1) In case of lower *oipl*s ratio, e.g. the ratio is less than 0.5, the final NDCG increases with the decrease of λ. The result is consistent with the previous theoretical analysis that *iipl* s and *oopl* s indicate the similarity relation in service input and output parameters. Especially, *iipl*s and *oopl*s in simulated datasets of the third experiment express the equivalence relationship between parameters, which is also the basis of baseline services similarity. Therefore, the similarity accuracy based on IIPCCN and OOPCCN is better than the accuracy based on OIPCCN. In conclusion, when the proportion of *oipl*s is low, the services similarity calculated based on OIPCCN can be neglected. (2) When the proportion of *oipl*s is higher than 0.7, if we neglect the services similarity based on OIPCCN, the similarity accuracy of the algorithm is the worst. It shows that the services similarity based on OIPCCN can improve the final effect in this case. Under this condition, we need to combine three composition contexts to calculate the similarity of services.

According to the third experiment, we can conclude that the services similarity based on IIPCCN and OOPCCN is more accurate than that based on OIPCCN. In practical application, for the case of the low *oipl*s ratio (less than 0.5), λ can be set 0, and the algorithm only depends on IIPCCN and OOPCCN to calculate services similarity. For the case of the higher *oipl*s ratio (higher than 0.5), λ can be set the value

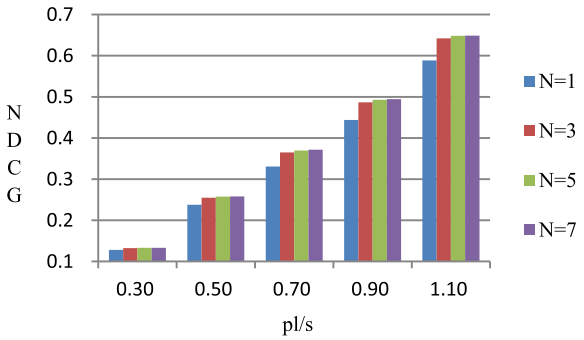

FIGURE 7. NDCGs under different cases.

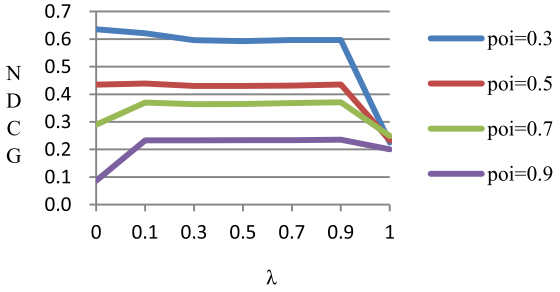

FIGURE 8. Influence of λ.

between 0.1 and 0.9 to ensure better accuracy on the services similarity.

We can draw the following conclusions based on the above three experiments. (1) Compared with the method that only considers the same services of the composition context, our approach has a better accuracy on the services similarity measure. (2) For a given set of services, the number of obtained *pc*s has an effect on the proposed method. The more *pc*s we obtain, the better accuracy of the proposed approach. (3) In the proposed method, *iipl*s and *oopl*s are better than *oipl*s in determining the services similarity. If the proportion of *oipl*s is low, services similarity can be calculated only using OOPCCN and IIPCCN. On the contrary, if the proportion of *oipl*s is high, it needs to combine OOPCCN, IIPCCN and OIPCCN to calculate the services similarity.

VII. CONCLUSION

In this paper, we propose a practical approach to measure Web services similarity from the perspective of dynamic features of services, which is different from traditional methods based on static descriptions of Web services. Dynamic interaction information of services, which contains neighbor services and correlations between their input and output parameters in existing compositions, are taken into account to measure services similarity. On the basis of this idea, we first propose a Web service composition context model based on parameter correlations. According to the context model, we build the composition context of each service from existing service compositions, and construct a global context network of all services. Using the context networks, we measure the similarity of any two services by the PersonalRank and SimRank ++ algorithms.

The proposed approach makes use of dynamic service composition contexts. However, if some services published on the Web have not been composed or invoked, we cannot get their composition records. Under such circumstances, we cannot use the proposed method to measure the similarity between these services and other services, whereas methods using syntactic and semantic descriptions of services can still work. This problem can be regarded as the "cold start" problem of our approach. In order to solve this problem and measure Web services similarity in a more comprehensive and accurate manner, we should combine our approach with traditional methods to study Web services similarity measure combining static and dynamic characteristics of services. This is one of our works in the future. In addition, the proposed service parameter correlation is the one-to-one correlation between service parameters. In order to deal with the heterogeneity of service parameters in the future, we should deal with more complex correlations, such as one-to-many or many-to-many correlations between service parameters.

REFERENCES

[1] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in Internet of vehicles: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2339–2352, Oct. 2015.

[2] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Gener. Comput. Syst.*, vol. 79, no. 3, pp. 849–861, Feb. 2017.

[3] A. Lemos, F. Daniel, and B. Benatallah, "Web service composition: A survey of techniques and tools," *ACM Comput. Surv.*, vol. 48, no. 3, Feb. 2015, Art. no. 33.

[4] M. Bravo and M. Alvarado, "Similarity measures for substituting Web services," *Int. J. Web Services Res.*, vol. 7, no. 3, pp. 1–29, Jul. 2010.

[5] L. He, L. Liu, and C. Wu, "A modified operation similarity measure method based on WSDL description," *Chin. J. Comput.*, vol. 31, no. 8, pp. 1331–1339, Aug. 2008.

[6] M. Liu, W. Shen, Q. Hao, and J. Yan, "An weighted ontology-based semantic similarity algorithm for Web service," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12480–12490, Dec. 2009.

[7] F. Liu, Y. Shi, J. Yu, T. Wang, and J. Wu, "Measuring similarity of Web services based on WSDL," in *Proc. ICWS*, Miami, FL, USA, 2010, pp. 155–162.

[8] Z. Zhou, M. Sellami, W. Gaaloul, M. Barhamgi, and B. Defude, "Data providing services clustering and management for facilitating service discovery and replacement," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 1131–1146, Oct. 2013.

[9] D. Athanasopoulos and A. V. Zarras, "Multi-objective service similarity metrics for more effective service engineering methods," in *Proc. SOCA*, Rome, Italy, 2016, pp. 208–212.

[10] J. Zhang, J. Li, S. Wang, and J. Bian, "A neural network based schema matching method for Web service matching," in *Proc. SCC*, Anchorage, AK, USA, 2014, pp. 448–455.

[11] B. Kim, H. Namkoong, D. Lee, and S. J. Hyun, "A clustering based schema matching scheme for improving matching correctness of Web service interfaces," in *Proc. SCC*, Washington, DC, USA, 2011, pp. 488–495.

[12] D. Athanasopoulos, "The aspect of data translation in service similarity," in *Proc. ICWS*, Honolulu, HI, USA, 2017, pp. 188–195.

[13] L. Ngan and R. Kanagasabai, "Semantic Web service discovery: State-of-the-art and research challenges," *Pers. Ubiquitous Comput.*, vol. 17, no. 8, pp. 1741–1752, Dec. 2013.

[14] Y. Ganjisaffar, H. Abolhassani, M. Neshati, and M. Jamali, "A similarity measure for OWL-S annotated Web services," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Hong Kong, Dec. 2006, pp. 621–624.

[15] R. A. H. M. Rupasingha, I. Paik, and B. T. G. S. Kumara, "Improving Web service clustering through a novel ontology generation method by domain specificity," in *Proc. ICWS*, Honolulu, HI, USA, 2017, pp. 744–751.
[16] R. Lara, D. Roman, A. Polleres, and D. Fensel, "A conceptual comparison of WSMO and OWL-S," in *Proc. ECOWS*, Erfurt, Germany, Sep. 2004, pp. 254–269.
[17] A. Abid, N. Messai, M. Rouached, T. Devogele, and M. Abid, "A semantic similarity measure for conceptual Web services classification," in *Proc. WETICE*, Larnaca, Cyprus, 2015, pp. 128–133.
[18] W. Lu, Y. Cai, X. Che, and Y. Lu, "Joint semantic similarity assessment with raw corpus and structured ontology for semantic-oriented service discovery," *Pers. Ubiquitous Comput.*, vol. 20, no. 3, pp. 311–323, Jun. 2016.
[19] P. Plebani and B. Pernici, "URBE: Web service retrieval based on similarity evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009.
[20] N. Zhang, J. Wang, Y. Ma, K. He, Z. Li, and X. Liu, "Web service discovery based on goal-oriented query expansion," *J. Syst. Softw.*, vol. 142, no. 8, pp. 73–91, Aug. 2018.
[21] Y. Du, J. Gai, and M. Zhou, "A Web service substitution method based on service cluster nets," *Enterprise Inf. Syst.*, vol. 11, no. 10, pp. 1535–1551, Apr. 2016.
[22] L. Kuang, Y. Xia, S. Deng, and J. Wu, "Analyzing behavioral substitution of Web services based on Pi-calculus," in *Proc. ICWS*, Miami, FL, USA, 2010, pp. 441–448.
[23] S. Bourouz and N. Zeghib, "Verifying Web services substitutability using open colored nets reduction techniques," in *Proc. ICMSAO*, Hammamet, Tunisia, 2013, pp. 1–5.
[24] H. Ren and J. Liu, "Service substitutability analysis based on behavior automata," *Innov. Syst. Softw. Eng.*, vol. 8, no. 4, pp. 301–308, Dec. 2012.
[25] X.-Q. Wang, C.-Q. Huang, X. Luo, R. Nie, Y. Tang, and X.-Y. Mei, "Determining substitutability of cloud services supported by semantically extended type theory," *J. Commun.*, vol. 37, no. 2, pp. 20–30, Feb. 2016.
[26] N. N. Chan, W. Gaaloul, and S. Tata, "Composition context matching for Web service recommendation," in *Proc. SCC*, Washington, DC, USA, 2011, pp. 624–631.
[27] N. N. Chan and W. Gaaloul, "Querying services based on composition context," in *Proc. WETICE*, Parma, Italy, 2014, pp. 44–49.
[28] N. N. Chan, N. Nonsung, and W. Gaaloul, "Service querying to support process variant development," *J. Syst. Softw.*, vol. 122, pp. 538–552, Dec. 2016.
[29] H. Wang and S. Li, "Service substitution method based on composition context," *J. Commun.*, vol. 35, no. 9, pp. 57–66, Sep. 2014.
[30] F. Zhang, Y. Wen, and Y. Wei, "Data correlation of data services oriented service hyperlink and modeling research," *J. Chin. Comput. Syst.*, vol. 38, no. 2, pp. 328–333, Feb. 2017.
[31] G. Wang, Y. Han, Z. Zhang, and S. Zhang, "A dataflow-pattern-based recommendation framework for data service mashup," *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 889–902, Nov./Dec. 2015.
[32] G. Wang, S. Yang, and Y. Han, "Mashroom: End-user mashup programming using nested tables," in *Proc. WWW*, Madrid, Spain, 2009, pp. 861–870
[33] C. Liu, J. Wang, and Y. Han, "Mashroom+: An interactive data mashup approach with uncertainty handling," *J. Grid Comput.*, vol. 12, no. 2, pp. 221–244, Jun. 2014.
[34] A. K. Dey, "Understanding and using context," *Pers. Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, 2001.
[35] L. Xiang, "Using user behavior data," in *Recommendation in Action*. Beijing, China: Posts and Telecommunications Press, 2013, pp. 73–77.
[36] T. H. Haveliwala, "Topic-sensitive PageRank," in *Proc. WWW*, Honolulu, HI, USA, 2002, pp. 517–526.
[37] G. Jeh and J. J. Widom, "SimRank: A measure of structural-context similarity," in *Proc. SIGKDD*, Edmonton, AB, Canada, 2002, pp. 538–543.
[38] I. Antonellis, H. G. Molina, and C. Chang, "Simrank++: Query rewriting through link analysis of the click graph," in *Proc. WWW*, Beijing, China, 2008, pp. 1177–1178.
[39] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov, "Accuracy estimate and optimization techniques for SimRank computation," *VLDB J.-Int. J. Very Large Data Bases*, vol. 19, no. 1, pp. 45–66, Jan. 2010.
[40] Y. Cai *et al.*, "Efficient algorithm for computing link-based similarity in real world networks," in *Proc. ICDM*, Washington, DC, USA, 2009, pp. 734–739.
[41] C. Wang, Y. Zhang, Y. Bao, C. Zhao, G. Yu, and L. Gao, "Asyn-SimRank: An asynchronous large-scale simrank algorithm," *J. Comput. Res. Develop.*, vol. 52, no. 7, pp. 1567–1579, Jul., 2015.
[42] H. Liu, J. He, D. Zhu, C. X. Ling, and X. Du, "Measuring similarity based on link information: A comparative study," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2823–2840, Dec. 2013.
[43] M. B. Blake and M. E. Nowlan, "Knowledge discovery in services (KDS): Aggregating software services to discover enterprise mashups," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 889–901, Jun. 2011.
[44] G. Huang, Y. Ma, X. Liu, Y. Luo, X. Lu, and M. Blake, "Model-based automated navigation and composition of complex service mashups," *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 494–506, May/Jun. 2015.
[45] L. Yao, Q. Z. Sheng, A. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based Web service recommendation," *IEEE Trans. Services Comput.*, vol. 8, no. 3, pp. 453–466, May 2015.

FENG ZHANG received the B.S. and M.S. degrees in computer science and technology from the College of Computer Science and Technology, Shandong University, Jinan, China, in 2002 and 2005, respectively, and the Ph.D. degree in computer software and theory from the College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao, China, in 2014, where he is currently a Lecturer with the College of Computer Science and Engineering. His research interests include service computing, web data integration, and business process management.

QINGTIAN ZENG received the Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He was a Visiting Professor with the City University of Hong Kong, Hong Kong, in 2008. He is currently a Professor with the Shandong University of Science and Technology, Qingdao, China. His current research interests include Petri nets, process mining, and knowledge management.

HUA DUAN received the B.S. and M.S. degrees in applied mathematics from the Shandong University of Science and Technology, Tai'an, China, in 1999 and 2002, respectively, and the Ph.D. degree in applied mathematics from Shanghai Jiao Tong University, in 2008. She is currently an Associate Professor with the Shandong University of Science and Technology. Her research interests include process mining and machine learning.

CONG LIU received the M.S. degree in computer software and theory from the Shandong University of Science and Technology, Qingdao, China, in 2015. His research interests are in the areas of business process mining, Petri nets, and software process mining.