Service Recommendation in an Evolving Ecosystem: A Link Prediction Approach

Keman Huang
Tsinghua National Laboratory for Information Science and Technology, Department of Automation
Tsinghua University
Beijing 100084, China
hkm09@mails.tsinghua.edu.cn

Yushun Fan
Tsinghua National Laboratory for Information Science and Technology, Department of Automation
Tsinghua University
Beijing 100084, China
fanyus@tsinghua.edu.cn

Wei Tan
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
wtan@us.ibm.com

Xiang Li
Department of Automation
Tsinghua University
Beijing 100084, China
cqlixiang@mails.tsinghua.edu.cn

***Abstract*— Services computing is playing a critical role in recent years in many fields and we observe a rapidly increasing number of web accessible services and their compositions nowadays. However, our earlier empirical study reveals that, overall the public available services are under-utilized, and when they are used, they are used mostly in an isolated manner. This phenomenon inspires us to further explore a methodology to help consumers understand the usage pattern of the service ecosystem, including interactions among services, and the evolution of these interactions. Based on the derived usage pattern, this methodology also introduces a service recommendation method that suggests both services and their compositions, in a time-sensitive manner. We firstly construct an evolution network model from the historical usage of the services in the ecosystem. Then a rank-aggregation-based link prediction method is proposed to predict the evolution of the ecosystem. Based on this link prediction method, we can recommend services and compositions of interest to service developers. Through an experiment on the real-world mashup-service ecosystem, i.e., ProgrammableWeb, we demonstrated that our approach can effectively recommend services and compositions with better precision than the methods we compared.**

Keywords- Rank Aggregation, Link Prediction, Network Evolution, Service Composition Recommendation

I. INTRODUCTION

Services computing is playing a critical role to enable service vendors and consumers to collaborate with each other over the Internet [1]. Providers, consumers, and services with their compositions, form a service ecosystem which evolves over time [2]. Furthermore, cloud computing technology makes it possible to share various types of services and resources on it [3]. As a result we can see a rapidly increasing number of web accessible services, their compositions, as well as services & compositions repositories [4]. For example, as of 2012/10/17, BioCatalogue[1] site collects over 2300 life science-specific services, 174 services providers and 698 members; myExperiment[2] collects over 2000 biological workflows; the Programmableweb[3] site has over 7630 services and 6568 compositions.

[1] http://www.biocatalogue.org/
[2] http://www.myexperiment.org/
[3] http://www.programmableweb.com/

In spite of these encouraging numbers, however, our earlier studies [5, 6] reveal that only a few services are ever used by the consumers and reuse rate of most services are rather low. For example, in ProgrammableWeb, only 22.2% published services are ever used in any mashup; 75.8% services which are ever used are invoked by no more than 5 compositions [5]. In BioCatalogue and myExperiment, similarly, only a few life science services are ever used in any workflow. Among the ever-used services, most of them are used individually rather than in collaboration with one other [6]. Furthermore we also observed that service ecosystems, including the consisting services/compositions, and their interactions, are continuously evolving. Therefore, *effectively discovering interesting services, their potential compositions in an evolving ecosystem and recommending them to the consumers* can help to promote these service ecosystems and make them self-sustainable. In this paper, , we propose a novel framework based on link prediction to recommend services and potential compositions so as to facilitate the re-use of the services in an evolving ecosystem.

The main contributions of this paper are as follows:

1) A novel network model that formally describes not only the interactions among services, but also the evolution of these interactions over time. The modeling of the evolution aspect is important because services and their interactions are constantly changing and a practical service recommendation approach should take this into consideration.

2) A rank aggregation based link prediction method for service composition recommendation, which can not only recommend services but also service compositions. Our method is scalable to a large-scale evolving service ecosystem. Also, we tackle the challenge of data sparseness in link prediction.

3) A comprehensive experiment on the real-world dataset, i.e., ProgrammableWeb, which contains 7630 services, 6568 compositions and 21876 correlations. Compared to many machine learning methods, such as J48, Naïve bayes, Adboost, LibSVM and Random Forests for link prediction in Weka [7]; we gain a 31.75% improvement in terms of precision for service pair recommendation. Compared to the popularity based method TopPop [8] for service recommendation, the precision of our approach is 3%~ 5% better.

978-0-7695-5025-1/13 $26.00 © 2013 IEEE
DOI 10.1109/ICWS.2013.74

Compared with existing service composition approaches, our approach moves a significant step forward in addressing the three observations below. First, we observe the fact that the historical usage of the services in the ecosystem embeds services/compositions interactions, and more importantly, services and their interactions evolve over time. This issue, however, is rarely been considered in existing service composition approaches. For example, services themselves are evolving overtime, e.g., publishing, perishing, and changing of interfaces and implementation [9]. Also there is an attachment effort in service usage; in terms of the services used in ProgrammableWeb in each month (a service is used in a month if a mashup using this service is published to ProgrammableWeb in that month), we find that about 80% of the services used in a given month are already used before that month. The second heuristics is that, un-observed service pairs (links that are not in our data set) cannot be simply considered as non-existing. Instead, some missing links may be due to the incompleteness of the data set and are likely to appear in the near future. Also the links are very sparse and no more than 0.1% service pairs will be re-used in the same composition. This extreme sparsity problem brings an extreme data imbalance issue [10] for recommendation. The third one is that recommendation ideally should not be limited to individual services [11]. Therefore our approach not only suggests "which service will be used in the future", but also "which services groups will be used".

Based on these three observations, we firstly model a service ecosystem using network model and formulate the service recommendation problem into the prediction of network evolution. Then we device a link prediction based method [12] to predict the evolution pattern of service interactions; we do service recommendation following the evolution pattern derived. Given the data sparsity and the limited scalability of rank aggregation method [13, 14], from a very different perspective, we introduce a rank aggregation based link prediction method. With the proposed method, three types of top-k service recommendations. i.e., individual services, services groups, and services chains, can be provided.

The rest of the paper is organized as follows. Section 2 gives the network model for service recommendation. Section 3 presents the rank aggregation based link prediction method for recommendation. Section 4 presents experimental results on ProgrammableWeb. Section 5 discusses the related work and Section 6 concludes this paper.

II. NETWORK MODEL AND RECOMMENDATION FRAMEWORK

Before we discuss the detail of these two issues, we will introduce several concepts in formalism which will be used in this paper.

A. The Network Model

Similar to our previous work [5] but in a different way, we define several networks as follows:

Composition-service network: $CSG =< M, S, E^{ms} >$ where M is the list of compositions and S is the list of services. $E^{ms} = \{(m,s) \mid m \in M, s \in S\}$ refers to the invoke relations between compositions and services. (m,s) refers that composition m invokes service s.

Service concurrence network: $SG =< S, E^{s} >$ where S is the list of services. $E^{s} = \{(u,v,w) \mid u \in S, v \in S\}$ represents the concurrence relations, (u,v) refers that service u and service v are ever invoked in the same composition and w is the weight of the edge.

The service concurrence network can be extracted from the composition-service network. Each composition will form a complete sub-graph of services in the service concurrence network. Thus we can formulate the composition recommendation into the prediction of pairs in the service concurrence network.

Furthermore, the service ecosystem is evolving over time. Thus given a series of consecutive time intervals $t_1, t_2, \cdots t_m$, we can define the snap network and the sum network for the service concurrence network as following:

The Snap Network $G_{t_i}^{snap}$ refers to the snapshot of the network which emerges during the given time interval t_i.

The Sum Network $G_{t_i}^{sum}$ refers to the network which emerges from the time interval t_1 to t_i.

The sum network is in fact the summarization of the snap network.

$$G_{t_i}^{sum} = \bigcup_{t_1 \prec t_j \prec t_i} G_{t_j}^{snap} \quad (1)$$

Here $t_1 \prec t_j \prec t_i$ refers to all the consecutive time intervals from t_1 to t_i.

For simplicity, in this paper, we refer the consecutive time interval just before t_i as $t_i - 1$. Thus $G_{t_i}^{sum} = G_{t_i - 1}^{sum} \bigcup G_{t_i}^{snap}$

Similarity, we can define the future network referring to the services and service pairs invoked in the following n time intervals.

$$G^{f}(t_i, n) = \bigcup_{t_i + 1 \prec t_j \prec t_i + n} G_{t_j}^{snap} \quad (2)$$

Obviously the future network is the sum network of the snap networks in the following n consecutive time intervals. If we set $n = 1$, the future network is in fact the next snap network.

B. Evolution Patterns

Based on the existence status of the correlation between each service pair in G_t^{sum} and $G^{f}(t,n)$, we can define the evolution of the service concurrence network as follows:

(1) *Cold Start Service Pairs*: refers to the service pairs in $G^{f}(t,n)$, which one or two endpoint services are never used before.

$$CSS(t) = \{(u,v) \mid (u,v) \in G^{f}(t,n), u \mid v \notin G_t^{sum}\} \quad (3)$$

(2) *Reused Service Pairs*: refers to the service pairs which reappear in new compositions.

$$RSP(t) = \{(u,v) \mid (u,v) \in G^f(t,n), (u,v) \in G_t^{sum}\} \quad (4)$$

(3) *Emerging Service Pairs*: refers to the services pairs which appear in $G^f(t,n)$ and the two endpoint services have ever been invoked, but never corroborated with each other in the same composition before.

$$ESP(t) = \{(u,v) \mid (u,v) \in G^f(t,n), (u,v) \notin G_t^{sum}, u,v \in G_t^{sum}\} \quad (5)$$

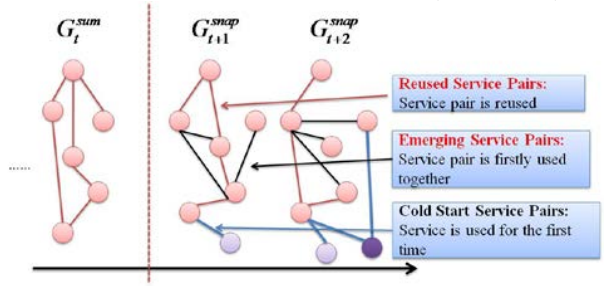

Figure 1. The evolution patterns in the service ecosystem including resused service pairs, emerging service pairs and the cold start service pairs.

Obviously the sum network G_t^{sum} carries the information about the *Reused Service Paris* and the *Emerging Service Pairs* in $G^f(t,n)$. This indicates that we can employ G_t^{sum} to predict these two kinds of service pairs. However, for the *Cold Start Service Pairs*, as we cannot simply predict the existence status of them by the sum network, we will leave them as the future work and it is out of this paper's scope.

Based on the preliminary definition above, we can define the prediction task in the evolving service ecosystem as:

Given a series consecutive time intervals $t_1, t_2, \cdots t_m$ *and the snap networks* $G_{t_1}^{snap}, G_{t_2}^{snap} \cdots G_{t_m}^{snap}$ *,predict the existence of the service pairs in* $G^f(t,n)$.

Obviously, the prediction method to predict the potential network is the most important task for our framework, we will show the detail of our rank aggregation based link prediction method in the next section.

C. Service Recommendation

The output of the prediction task is a list of service pairs with the possibility to appear in the following *n* time intervals. We can get the top-k service pairs to form the predicted service concurrence snap network. Based on this network, we can offer three kinds of service recommendation for the consumers:

1) Potential Composition Recommendation

As the services in a composition will form a complete sub-graph in the service concurrence network, we can use the CPM approach [15] to extract the potential service cliques from the predicted snap service network. Each service clique can be considered as a potential composition. In fact, any algorithm that can extract the overlap community from network can be used to substitute the CPM algorithm we used in this paper.

2) Top Service Recommendation

Based on the top-k service pairs, we can get a list of the endpoint services of these service pairs which are sorted by their frequency in the top-k service pairs. Thus we can also offer the top service recommendation for the consumers.

3) Service Chain Recommendation

Given any services in the recommended services, the ServiceMap algorithm we proposed in [16] can be used to get the potential service chains for the consumers.

Note that our recommendation method in this paper is non-personalized which means that we don't do the recommendation for each developer but find out the services and their correlations which will attach all the developers' attentions. Based on the predicted network, the personalized recommend algorithm such as collaborative filtering [17] could be used but it's out of this paper's scope.

D. Recommendation Framework

TABLE I
SERVICE COMPOSITION RECOMMENDATION FRAMEWORK

Algorithm 1: rank aggregation based link prediction method for service composition recommendation

Input:

(1) consecutive time intervals $t_1, t_2, \cdots t_m$

(2) snap network series $G_{t_i}^{snap}$ $t_i = t_1, t_2, \cdots t_m$

(3) parameter: n: future n time interval network

k: top-k potential service pairs

l: the largest number of services in each compositions

Output:

(1) predicted snap network $G_p^{snap}(t,n,k)$

(2) service list $\{S\}$

(3) potential service compositions $\{C(l)\}$

01: $G_{t_i}^{sum} \leftarrow \phi, t_i = t_1, \cdots t_m$

02: For $t_i = t_2, \cdots t_m$

03: $G_{t_i}^{sum} \leftarrow G_{t_i-1}^{sum} \bigcup G_{t_i}^{snap}$

04: $G^f(t_i,n) = \bigcup_{t_i+1 < t_j < t_i+n} G_{t_j}^{snap}$

05: $\{E\}_k \leftarrow link\_prediction(G_{t_i}^{sum}, G^f(t_i,n), k)$

06: $G_p^{snap}(t_i,n,k) \leftarrow \{E\}_k$

07: $\{C(l)\} \leftarrow cpm(G_p^{snap}(t_i,n,k), l)$

08: $\{S\} \leftarrow sort(\{E\}_k)$

09: End

Table I shows the framework of our approach. For each time interval t, Line 03 firstly constructs the sum service concurrence network G_t^{sum} based on G_{t-1}^{sum} and G_t^{snap}; Line 04 builds the future snap network for test; then Line 05 employs the rank aggregation based link prediction method to predict the potential service pairs in the future so that we can get the top-k service pairs $\{E\}_k$; Line 06~07 constructs the predicted concurrence network and then extracts the potential compositions which invoke no more than l services and offer the service composition recommendation; Line 08

sort the endpoint services by their frequency in the top-k service pairs and recommend the top services for consumers.

III. RANK AGGREGATION BASED LINK PREDICTION

Link prediction [12] offers unique ways to study the inside of the network and the evolution of the network in time, thus we employ the link prediction method to predict the evolution of the ecosystem. In this section, we introduce a rank aggregation method to study the link prediction problem [14]. Firstly we give some preliminary definition and the formulation of the problem.

A. Preliminary Definition

Let $p_i^t(u,v)$ denotes a vertex pair (u,v) in the service network G_t^{sum} in time interval t. Here i is used to index different vertex pairs in this network. Obviously $i = 1,\cdots N$ where $N = \frac{1}{2}|V|(|V|-1)$, $|V|$ refers to the number of services in G_t^{sum}. For simplicity but without confusion, we can use p_i^t for short. $P^t = \{p_1^t,\cdots p_N^t\}$ is the set of all the possible service pairs in G_t^{sum}.

For each service pair p_i^t, we can calculate the values for different features such as *the current weight, the common neighbors* [12], *etc.* The detail of these features will be shown in Section IV. Based on the values we can get a ranker r_j^t for all the service pairs. $r_j^t = \{p_{j,1}^t,\cdots p_{j,N}^t\}, j = 1,\cdots fn$. Here fn is the number of features we considered for the link prediction. Based on the position in the ranking list, we can calculate the extended Borda's score [13] for each service pair:

$$B(r_j^t, p_i^t) = \begin{cases} \dfrac{k - \pi(r_j^t, p_i^t) + 1}{k} & \pi(r_j^t, p_i^t) \le k \\ \dfrac{1}{\pi(r_j^t, p_i^t)} & \pi(r_j^t, p_i^t) > k \end{cases} \quad (6)$$

Where $\pi(r_j^t, p_i^t)$ refers to the position of p_i^t in r_j^t, $1 \le \pi(r_j^t, p_i^t) \le N$. A higher position indicates a lower π and then a higher score.

The goal of the rank aggregation is to find an aggregation ranking function f to combine all the ranking lists [18] so that we can get an aggregated score for each service pair which refers to the re-used possibility in the future. In this paper, we use the linear function, i.e. $f(p_i^t) = \sum_{1 \le j \le fn} w_j B(r_j^t, p_i^t)$. Here we can rewrite the function in a matrix form as follow:

$$f^t = R^t W \quad (7)$$

Where $R_{ij}^t = B(r_j^t, p_i^t)$ and $W = [w_1, w_2, \cdots w_{fn}]^T$.

Sorting the service pairs by the f^t score in descending order, we get the aggregated ranking list of service pairs. Based on this aggregated ranking list, the top-k service pairs will be offered for the consumers based on the hypothesis that The higher the score of the service pair is, the higher the possibility is that the service pair will be used in the future.

B. Basic Rank Features

For each service pair (u,v) in the sum network, we consider the following features:

Current Edge Weight (*CEW*):

If there is no service correlation between them, the current edge weight would be 0;

Weight Common Neighbor (*WCN*):

$$WCN(u,v) = \sum_{k \in N(u) \cap N(v)} w(u,k) \times w(u,k) \quad (8)$$

which is the summarization of the product of the weight for their adjacent edges.

Adamic-Adar measure (*WAA*):

$$WAA(u,v) = \sum_{i \in N(u) \cap N(v)} \frac{w(u,i) \times w(v,i)}{\log(|N(i)|)} \quad (9)$$

Preferential Attachment measure (*WPA*):

$$WPA(u,v) = \log(\sum_{i \in N(u)} w(u,i) \times \sum_{j \in N(v)} w(v,j)) \quad (10)$$

Last Edge Occurred Time Stamps (*LE*):

$$LE_t(u,v) = \begin{cases} 0 & (u,v) \in G_t^{snap} \\ t-i & (u,v) \in G_i^{snap}, (u,v) \notin G_j^{snap}, 0 \le i < j < t \\ t+1 & (u,v) \notin G_t^{sum} \end{cases} \quad (11)$$

Average Vertex Occurred Time Stamps (*LEV*):

$$LEV_t(u,v) = LV_t(u) + LV_t(v)$$

$$LV_t(u) = \begin{cases} 0 & (u) \in G_t^{snap} \\ t-i & (u) \in G_i^{snap}, (u) \notin G_j^{snap}, 0 \le i < j < t \\ t+1 & (u) \notin G_t^{sum} \end{cases} \quad (12)$$

Thus for each service pair, a 6-dimensional vector is generated.

C. Rank Quality Evaluation

1) Quality Evaluation for Service Pairs

Here we use the existence of the service pairs as the ground truth. As we discuss above, we can not simply consider the un-observed service pairs in the next interval as the negative instance as they may appear in the near future. Thus we can define the ground truth of the service pairs as:

$$y(p_i^t, n) = \begin{cases} 1 & p_i^t \in G^f(t,n) \\ 0 & p_i^t \notin G^f(t,n) \end{cases} \quad (13)$$

Based on this, we can calculate the performance of the ranking list by precision ($P@t,k,n$), recall ($R@t,k,n$)and normalized discounted cumulative gain ($NDCG@t,k,n$).

$$P@t,k,n = \frac{\sum_{p_i^t \in top-k} y(p_i^t, n)}{k} \quad (14)$$

$$R@t,k,n = \frac{\sum_{p_i^t \in top-k} y(p_i^t, n)}{|\{(u,v) \in G^f(t,n) \,\&\, u,v \in G_t^{sum}\}|} \quad (15)$$

$$NDCG@t,k,n = \frac{1}{D}\sum_{p_i^t \in top-k} \frac{y(p_i^t,n)}{\log(1+\pi(r,p_i^t))} \quad (16)$$

Where $top-k$ refers to the service pairs which own the top-k position in the ranking list. $\sum_{p_i^t \in top-k} y(p_i^t,n)$ refers to the total number of service pairs which form at least one correlation in the future. $\{(u,v) \in G^f(t,n) \& u,v \in G_t^{sum}\}$ refers to all the service pairs in $G^f(t,n)$ which endpoint services are ever used before. In fact, they are the summarization of the reused service patterns and the emerging service patterns. D is the value for the ideal ranking so the $NDCG@t,k,n$ for the best ranking list would be 1.

2) Quality Evaluation for Services

Similarly, we get the ground truth for each service as:

$$y(u,n) = \begin{cases} 1 & u \in G^f(t,n) \\ 0 & u \notin G^f(t,n) \end{cases} \quad (17)$$

For the services in the top-k service pairs, we sort them by their frequency and get the service list $\{V\}$. For $\{V\}$ we calculate the precision ($VP@h,t,k,n$) and the recall ($VR@h,t,k,n$) as:

$$VP@h,t,k,n = \frac{1}{h}\sum_{v \in top-h} y(v,n) \quad (18)$$

Where $top-h$ refers to the top-h services in $\{V\}$;

$$VR@h,t,k,n = \frac{1}{|PV|}\sum_{v \in top-h} y(v,n) \quad (19)$$

Where $PV = \{v \in G^f(t,n) \& v \in G_t^{sum}\}$ refers to all the services which will be reused in the following n time intervals.

D. Weight Vector Computation

For a given time interval t, we calculate the performance of each basic ranker in time interval $t-1$. Then we can define the following aggregators based on different weight vectors:

1) Simple Rank Aggregator (*SR*)

$$W = [\frac{1}{fn}, \frac{1}{fn}, \cdots \frac{1}{fn}]^T \quad (20)$$

which means that we consider all the basic rankers evenly.

2) Precision-Based Supervised Aggregator (*PR*)

$$W = \frac{1}{\varpi}[P(r_1^{t-1})@t-1,k,1 \cdots P(r_{fn}^{t-1})@t-1,k,1]^T \quad (21)$$

where $P(r_{fn}^{t-1})@t-1,k,1$ means the top-k precision of the basic ranker r_{fn}^{t-1} for G_t^{snap}. ϖ is used to normalized the weight vector.

3) NDCG-Based Supervised Aggregator (*NR*):

$$W = \frac{1}{\sigma}[NDCG(r_1^{t-1})@t-1,k,1 \cdots NDCG(r_{fn}^{t-1})@t-1,k,1]^T \quad (22)$$

where $NDCG(r_{fn}^{t-1})@t-1,k,1$ means the top-k NDCG of the basic ranker r_{fn}^{t-1} for G_t^{snap}. σ is used to normalized the weight vector.

IV. EXPERIMENTS

To the best of our knowledge, ProgrammableWeb is by far the largest online repository of Web APIs (i.e, services) and their mashups (i.e., compositions). Throughout this paper, we will use "composition" and "mashup" interchangeably as they both combine atomic services to provide added value. In this section, we employ our rank aggregation based link prediction framework to study the evolution of this ecosystem and then recommend the potential service and compositions for developers..

A. Data Set Introduction

In this paper, we obtain the ProgrammableWeb data regarding services and compositions from June 2005 to July 2012. In this data set, each service contains the information such as name and the publication date; each composition contains the information such as name, creation date and the list of services in it; By removing compositions which contain no services, we get a collection of 7076 services and 6597 compositions. Table II reports the basic properties of our dataset.

TABLE II
BASIC FEATURES OF THE PROGRAMMABLEWEB DATA

Number of services (#services for short)	7076
Number of compositions (#compositions for short)	6597
Average number of services per composition	2.101
Average number of compositions per service**	12.17
Services used in at least one composition	1139

**: only consider the services which are used in at least one mashup

For each month, we get the snap network and then construct the sum network.

Figure 2 shows the evolution of the snap network and the sum network over time during our study period. Here the network density of snap network is defined as:

$$Density(t) = \frac{2|E_t^{snap}|}{|V_t^{snap}|(|V_t^{snap}|-1)} \quad (23)$$

where $|E_t^{snap}|$ is the number of service pairs and $|V_t^{snap}|$ is the number of services in the certain snap network.

The sparsity of the sum network is defined as:

$$Sparsity(t) = \frac{2|E_t^{p}|}{|V_t^{sum}|(|V_t^{sum}|-1)} \quad (24)$$

where $|V_t^{sum}|$ is the number of services in the sum network and . $|E_t^p|$ is the number of the service pairs which appear in the following snap network. These service pairs are the sum of the reused service pairs and the emerging service pairs.

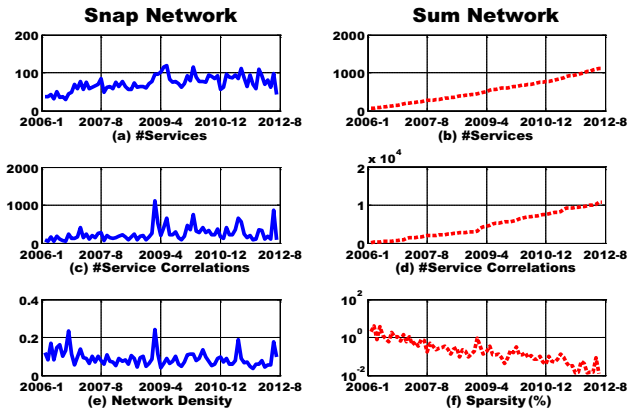

Figure 2. Scale Evolution of the Snap Network and the Sum Network. (a) Number of services in each snap network; (b) Number of services in each sum network; (c) Number of service pairs in each snap network; (d) Number of service correlations in each sum network; (e) Network density of each snap network; (f) Network sparsity of each sum network.

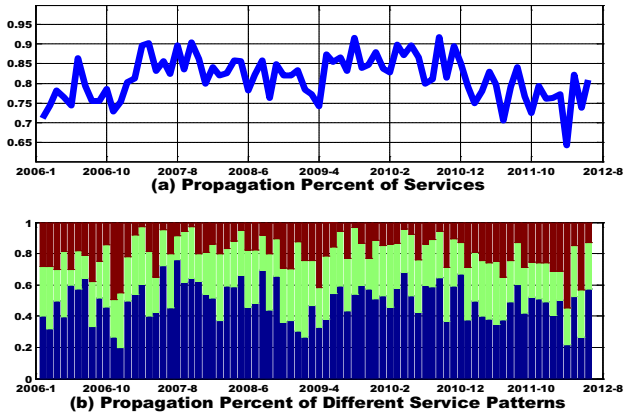

Figure 3. Propagation of the Snap Network from the Sum Network. (a) Percent of services in each snap network which are ever used before. (b) Percent of different service patterns in each snap network. From top to down is *Cold Start Service Patterns, Emerging Service Patterns and Reused Service Patterns.*

From Figure 2, we observe that the number of services in each snap network is relatively stable and it is between 50 and 100. Mostly, the number of the service pairs is no more than 500 and the network density of the snap network is about 0.1. The sparsity is declining and the value is extremely small, no more than 1% after 2007-8.

For each snap network, we get the percent of the services which have been used before, which we name as *propagated services*, and the three different service pairs we mention in Section II. From Figure 3, we can get that about 80% of services in each snap are ever used before and about 80% of service patterns are *Emerging Service Pairs* and *Reused Service Pairs*. Thus it makes sense to predict the evolution of the network based on the historical information.

B. Performance Comparison

1) Service Pair Recommendation

To study the performance of the rank aggregation based link prediction, we compare our approach with four others, in terms of precision. We consider four basic machine learning approaches, i.e., *J48, naïve bayes* (*NaBy*), *Adboost embed with libSVM* (*AdSvm*) *and Adboost embed with random forests* (*AdRf*), which all formulate the link prediction problem into the binary classification problem. We use the implementation of these four approaches in Weka [7]. For a given time interval t, we firstly train each classifier based on G_{t-1}^{sum} and G_t^{snap}; then for each service pair in G_t^{sum}, we calculate its existence possibility based on the classifier; finally we sort all the service pairs by its possibility to offer the top-k service pair recommendation.

The last three approaches are rank aggregation based methods which we presented in Section III: *Simple Rank Aggregator (SR), Precision-Based Supervised Aggregator (PR)* and *NDCG-Based Supervised Aggregator (NR).*

TABLE III
PERFORMANCE COMPARISON FOR CORRELATION RECOMMENDATION

	$P@t,k,1$				$P@t,k,10$			
	$k=5$	$k=10$	$k=200$	$k=300$	$k=5$	$k=10$	$k=200$	$k=300$
J48	0.000	0.000	0.000	0.003	0.000	0.000	0.010	0.007
NaBy	***0.000***	***0.200***	***0.200***	***0.173***	***0.600***	***0.600***	***0.630***	***0.537***
AdSvm	0.000	0.000	0.130	0.167	0.400	0.300	0.530	0.523
AdRf	0.000	0.000	0.130	0.167	0.400	0.300	0.530	0.523
SR	0.800	**0.700**	0.240	0.180	**1.000**	**1.000**	0.810	**0.580**
PR	**1.000**	**0.700**	**0.280**	**0.190**	**1.000**	**1.000**	**0.830**	0.563
NR	**1.000**	**0.700**	0.270	0.187	**1.000**	**1.000**	**0.830**	0.560

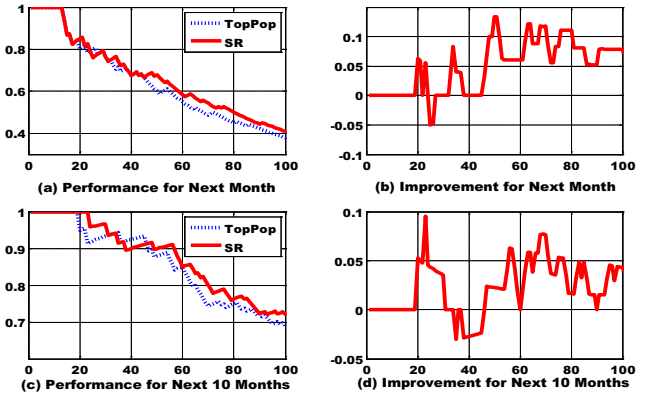

Figure 4. Precision for Service Recommendation in Jan-2008. (a) Performance comparison for next month; (b) Performance improvement of SR for next month; (c) Performance for the next 10 months case; (d) Performance improvement of SR for the next 10 months situation.

Due to the limitation of space and the consistency of the result, in this paper, we just take the time interval Jan-2008 as example. Table III reports the performance comparison based on $P@t,k,1$ and $P@t,k,10$. *NaBy* is the best in the four machine learning based methods. However all the three rank aggregation based method are much better than *NaBy*. For the top 200 service pairs, *PR* has a 40% improvement for the next month and a 31.75% improvement for the next 10 months. We should note here that the value of the

$P@t,k$,10 (k=200) for *PR* is 0.83 which means that 83% of the top 200 service pairs will appear in the future. This indicates that the performance of the rank aggregation method is significant for the service pair recommendation.

2) Service Recommendation

To study the performance of our method for service recommendation, we compare SR with TopPop [8], which recommend the top services by their popularity.

From Figure 4, we can observe that both our method and TopPop are significant for the service recommendation as the top 100 precisions for the future are both larger than 0.7 which means that 70% services we recommend will be reused in the future. Additionally, comparing with TopPop, our method gains a 5%~10% improvement for next month and 3%~ 5% improvements for the next 10 months.

V. Related Work

Our method is at the cross-road of the composition recommendation and the link prediction.

A. Composition recommendation

With a rapidly increasing number of web services on the Internet, service composition recommendation has become an important issue in industry and academia.

In recent years, some new generation recommendation tools for service compositions have been supplied. For example, the mashup composers such as Yahoo! Pipes, Google Mashup Editor, Microsoft Popfly and IBM QEDwiki are used to help the developers to find the related services and combine them into a composition [19]. These tools require the developers to select the certain services they want which make it difficult for the non-skill developers. Actually, all except Yahoo! Pipes are unavailable now.

Some try to convenience the design of the composition for mashups or service-based workflows. Some recommend services based on the *semantic similarity* [20, 21]. For example, MashupAdvisor [20] suggests the services based on the user's input and popularity of services in the service repository. However, service itself is evolving overtime and it rarely remains stable [9], which means that the service will adapt itself to fit the changing requirement. Furthermore, the RESTful services [22] have been widely used. All these make it difficult to employ the semantic-based recommendation approach in practice. Some recommend services based on the *service QoS* [23, 24]. These Qos-aware methods predict the evolution of the services' Qos and then recommend the services with the highest quality for fault-tolerant application. Some begin to consider the correlations between services and recommend services from the perspective of *complex network and social network* [11, 16, 25]. For example, ServiceMap [16] provides a GPS-like assistance to service composition based on the service correlations so that the developers can understand the used patterns of their intended services and find out the possible operation chains among them. [8] shows the efficiency of the non personalized algorithm 'TopPop' based on the service's popularity.

These existing methods take very little consideration on the correlations of services or the evolution of the service usage over time. From a very different perspective, we focus on the evolution of the service correlations over time so that we can recommend not only the services but also their compositions.

B. Link Prediction

Link prediction [12] is currently an active research area in computer science and benefits researchers and organizations in many fields such as social network [26]. It offers unique ways to study the inside of the network and its evolution mechanism. As the link prediction problem can be translated into a binary classification problem, many machine learning methods such as Decision tree, naïve bays and SVM can be used to solve it. For example, [27] proposes a supervised machine learning method based on decision tree to deal with the link prediction problem.

However, the extreme sparsity problem of the evolving service networks bring extreme data imbalance for machine learning [10].Thus from a very different perspective, we employ the rank aggregation based method to study the link prediction problem.

VI. Conclusion and Future Work

Despite the wide adoption of SOA, it is still very challenging for service consumers to understand the usage patterns in an evolving service ecosystem. Therefore, how to recommend potential services and compositions, especially in a time-aware fashion which means taking the evolution of the ecosystem into account, becomes an important issue for the promotion of the service ecosystem.

To be best of our knowledge, we are the first to 1) address the evolution aspect of services interactions and model it using a dynamic network; 2) introduce a rank aggregation based link prediction method to recommend services, compositions (services groups), service chains; and 3) recommend top-k services and service composition based on the proposed dynamic network model and link prediction method. The experiments on ProgrammableWeb shows that the rank aggregation based link prediction approach can not only recommend the services but also the service compositions efficiently:

- For service pair recommendation, compared to the machine learning methods to which we made comparison in this paper, the rank aggregation method gain a 40% improvement in terms of precision for next-month prediction, and a 31.75% improvement for next-10-months prediction. The precision for predicting the top 200 service-pairs is up to 0.83 which indicates that our method is significant for service pair recommendation.
- For service recommendation, comparing to the famous TopPop method [8], the precision of our method is 5%~10% better for next-month prediction and 3%~ 5% for

next-10-months. The precision for the top 100 service is up to 0.72.

Note that we do not take the functionality of the services into account in this paper. In the future, we will further study the evolution of the functionality in the ecosystem to construct a more comprehensive and accurate framework for services and compositions recommendation.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. 61033005 and No. 61174169) and the National High Technology Research and Development Program of China (863Program, No. 2012AA040915)

REFERENCES

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing Research Roadmap," World Scientific Publishing Co. Pty. Ltd, vol. 5, pp. 223--255, 2006.

[2] A. P. Barros and M. Dumas, "The Rise of Web Service Ecosystems," IT Professional, vol. 8, pp. 31-37, 2006-01-01 2006.

[3] J. Wang, J. Zhang, P. C. K. Hung, Z. Li, J. Liu, and K. He, "Leveraging fragmental semantic data to enhance services discovery," in IEEE International Conference on High Performance Computing and Communications, 2011.

[4] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in Proceedings of the 17th international conference on World Wide Web, 2008, pp. 795--804.

[5] K. Huang, Y. Fan and W. Tan, "An Empirical Study of ProgrammableWeb: A Network Analysis on a Service-Mashup System," in 19th International Conference on Web Services (ICWS 2012) Honolulu, Hawaii, USA, 2012.

[6] W. Tan, J. Zhang and I. Foster, "Network Analysis of Scientific Workflows: A Gateway to Reuse," Computer, vol. 43, pp. 54-61, 2010.

[7] I. H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques: Morgan Kaufmann, 2005.

[8] P. Cremonesi, M. Picozzi and M. Matera, "A comparison of recommender systems for mashup composition," in Third International Workshop on Recommendation Systems for Software Engineering (RSSE), Zurich, Switzerland, 2012, pp. 54--58.

[9] V. Andrikopoulos, S. Benbernou and M. P. Papazoglou, "On the Evolution of Services," IEEE Transactions on Software Engineering, vol. 38, pp. 609-628, 2012-01-01 2012.

[10] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 42, pp. 463--484, 2012.

[11] Z. Maamar, L. K. Wives, Y. Badr, S. Elnaffar, K. Boukadi, and N. Faci, "LinkedWS: A Novel Web Services Discovery Model Based on the Metaphor of Social Networks," Simulation Modelling Practice and Theory, Elsevire Science Publisher, vol. 19, pp. 121-132, 2011.

[12] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American society for information science and technology, vol. 58, pp. 1019--1031, 2007.

[13] J. A. Aslam and M. Montague, "Models for metasearch," in Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, pp. 276--284.

[14] F. Wei, W. Li and S. Liu, "iRANK: A rank-learn-combine framework for unsupervised ensemble ranking," Journal of the American Society for Information Science and Technology, vol. 61, pp. 1232--1243, 2010.

[15] G. Palla, I. Der E Nyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," Nature, vol. 435, pp. 814--818, 2005.

[16] W. Tan, J. Zhang, R. Madduri, I. Foster, D. De Roure, and C. Goble, "ServiceMap: Providing Map and GPS Assistance to Service Composition in Bioinformatics," in 2011 IEEE International Conference on Services Computing (SCC), 2011, pp. 632-639.

[17] S. Chen, F. Wang, Y. Song, and C. Zhang, "Semi-supervised ranking aggregation," Information Processing & Management, vol. 47, pp. 415--425, 2011.

[18] M. Weiss and G. R. Gangadharan, "Modeling the mashup ecosystem: structure and growth," R&D Management, vol. 40, pp. 40-49, 2010.

[19] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "Mashup advisor: A recommendation tool for mashup development," in IEEE International Conference on Web Services, 2008, pp. 337--344.

[20] D. Le-Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, and C. Morbidoni, "Rapid prototyping of semantic mash-ups through semantic web pipes," in Proceedings of the 18th international conference on world wide web, 2009, pp. 581--590.

[21] D. Benslimane, S. Dustdar and A. Sheth, "Services mashups: The new generation of web applications," IEEE Internet Computing, vol. 12, pp. 13--15, 2008.

[22] R. T. M. Lyu, I. K. C. King and Z. Zheng, "Component Ranking for Fault-Tolerant Cloud Applications," IEEE Transactions on Service Computing, vol. 4, pp. 1--12, 2011.

[23] H. Liu, Z. Zheng, W. Zhang, and K. Ren, "A global graph-based approach for transaction and QoS-aware service composition," KSII Transactions on Internet and Information Systems (TIIS), vol. 5, pp. 1252--1273, 2011.

[24] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse," in 2011 IEEE International Conference on Services Computing (SCC), 2011, pp. 48-55.

[25] J. Leskovec, D. Huttenlocher and J. Kleinberg, "Predicting positive and negative links in online social networks," in Proceedings of 19th International Conference on World Wide Web, 2010, pp. 641--650.

[26] V. Leroy, B. B. Cambazoglu and F. Bonchi, "Cold start link prediction," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 393--402.

[27] R. Raymond and H. Kashima, "Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs," Machine Learning and Knowledge Discovery in Databases, pp. 131--147, 2010.

[28] P. Sarkar, D. Chakrabarti and M. Jordan, "Nonparametric Link Prediction in Dynamic Networks," arXiv preprint arXiv:1206.6394, 20