Recommendation in an Evolving Service Ecosystem based on Network Prediction

Keman Huang, Student Member, IEEE, Yushun Fan*, Wei Tan, Senior Member, IEEE

Abstract—Service computing plays a critical role in business automation and we can observe a rapid increase of web services and their compositions nowadays. Web services, their compositions, providers, consumers and other entities such as context information, collectively form an evolving service ecosystem. Many service recommendation methods have been proposed to facilitate the use of services. However, existing approaches are mostly based on all-time statistics of usage patterns, and overlook the temporal aspect, i.e., the evolution of the ecosystem. As a result, recommendation may consist of obsolete services and also does not reflect the latest trend in the ecosystem. In order to overcome this limitation, we propose an innovative three-phase network prediction approach (NPA) for evolution-aware recommendation. Firstly we introduce a network series model to formalize the evolution of the service ecosystem and then develop a network analysis method to study the usage pattern with a special focus on its temporal evolution. Afterward a novel service network prediction method based on rank aggregation is proposed to predict the evolution of the network. Finally, using the network prediction model we present how to recommend potential compositions, top services and service chains, respectively. Experiments on the real-world ProgrammableWeb data set show that our method achieves a superior performance in service recommendation, compared with those that are agnostic to the evolution of a service ecosystem.

Note to Practitioners—Understanding the usage pattern and the evolution mechanism can help better recommend services and their compositions to developers. Our hypotheses are: 1) Historical information carries the usage patterns and evolution mechanism of the ecosystem; 2) services are not isolated but collaborative with each other, therefore we should not only concern about individual services but also their correlations. Based on these hypotheses, our network prediction based approach constructs the network series model based on historical information, and transform the evolution prediction into a network prediction problem. Furthermore we predict the future behavior of the network based on link prediction. Based on the predicted network, we recommend potential compositions, services and service chains that are evolution-aware and better reflect the up-to-date trend in the ecosystem.

Index Terms—Network Series Model, Network Analysis, Link Prediction, Rank Aggregation, Network based Recommendation, Evolving Service Ecosystem

1. INTRODUCTION

WITH the wide adoption of Service-Oriented Architecture (SOA), various resources such as data [1], web page [2] and scientific process [3] are published as services and we can observe a rapid increasing number of services and their compositions [4, 5]. Through the composition of services, services providers and consumers collaborate with each other creating added values over the Internet [6]. Therefore, service providers, consumers, their compositions and the composition developers, collectively form a service ecosystem [7]. ProgrammableWeb1, myExperiment2 and Biocatalogue3 are a few examples of such ecosystems. A service ecosystem is constantly evolving over time, as new services emerging, existing services perishing, and services dynamic collaborating with each other. How to identify high-quality services and their compositions and recommend them to composition developers remains a challenge to facilitate the re-use of services and promote the growth of the ecosystem.

Existing service recommendation approaches are mainly based on semantic similarity of the services interface [8, 9], Quality of Services (QoS) combined with the Collaborative Filtering (CF) [10, 11] and social network embedded in the service ecosystem [12-15]. These approaches are based on static information, and have three pitfalls. First, in real-life service ecosystems, services will merge, upgrade, retire and become obsolete [16]. This evolution nature is not considered by most existing approaches. As a consequence, their recommendations do not reflect the latest trend in the ecosystem and may be obsolete. Second, for most of the real-life web services, the QoS information is missing. The direct fetching of these QoS information, especially over time, is usually unrealistic. Third, many developers start a composition task with a very vague goal in their mind [17], and a lot of recommendation methods which need a complete and clear input are not applicable in this context.

Based on our experiences working on several service ecosystems [18-22], we believe that studying the evolution pattern of a service ecosystem can shed some light to address the three pitfalls listed above. First, the historical usage information in the ecosystem reflects the published, upgraded and perishing of services and their popularity over time. It enables us to filter out obsolete services/services-pair and obsolete services/services-pair and methods are not applicable in this context.

Keman Huang is with the Automation Department, Tsinghua University, Beijing, 100084 China. (E-mail: keman@tsinghua.edu.cn). Corresponding author.

Yushun Fan is with the Automation Department, Tsinghua University, Beijing, 100084 China. (Corresponding, E-mail: fanyus@tsinghua.edu.cn). Corresponding author.

Wei Tan is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA (E-mail: wtan@us.ibm.com).

1 www.programmableweb.com
2 www.myexperiment.org
3 www.biocatalogue.org
recommend services being actively used currently or in the near past. Second, historical usage information can be seen as an indirect reflection of services’ QoS and consumers’ preferences, when QoS information is not readily available directly. Third, the evolution of the ecosystem can predict the compositions likely to appear in the future, and can be complementary to other approaches requiring a more precise input. With these intuitions, we investigate the evolution of a services ecosystem and propose a novel network prediction based approach to recommend the valuable services and their compositions. We try to address the following issues:

*What is the usage pattern of the services and their compositions in the ecosystem and how are they evolving over time?*

*How to predict the evolution of the service ecosystem, and leverage such a prediction for recommendation?*

As a service ecosystem can be considered as the collection of services and their compositions and network analysis is a powerful tool to understand the nature of large scale networks [23], we construct two networks: *Composition-Service Network (CSN)* which is a bipartite graph describing invocation relations between compositions and services; *Service-Service Network (SSN)* which is a homogeneous network describing the concurrency relation between the two services. Then the *network series model (NSM)* which consists of the *snap CSN series*, the *snap SSN series* and the *temporal collapsed sum SSN series*, is presented to model the evolution of the service ecosystem. Based on the *NSM*, we formulate the evolution prediction problem into a *temporal network link prediction task* [24]. Then we present a comprehensive analysis about the static usage patterns and their evolution over time in the service ecosystem.

Applying the insights of the service ecosystem from the network analysis, we study the fundamental features including the temporal and topological features, and then quantify their influence to the ecosystem evolution. In order to leverage different fundamental features and their evolution patterns, we use the rank aggregation method to combine the utility of each factor to solve the temporal network link prediction problem and then form the predicted SSN. Finally three network based methods are presented to fulfill different requirements: detecting the compositions in the predicted SSN to offer potential composition recommendation; ranking services based on network degree to offer top-\(k\) service recommendation; searching service chains in the predicted SSN for given services to offer instance recommendation for service composition. Experiment on real-life data set demonstrates that, this evolution-aware approach enables us to filter out obsolete services/services-pair and recommend services being actively used currently or in the near past.

The main contribution in this paper is a novel network prediction based approach to incorporate the evolution mechanism of the service ecosystem for service recommendation:

1) We present a network series model to formally define the evolution of a service ecosystem and transform its prediction into a temporal network prediction problem. Then we construct a comprehensive study on its static and evolution patterns to gain insight about the service ecosystem.

2) We quantify the fundamental features including the temporal and topological features for ecosystem evolution and introduce a rank aggregation method to solve the prediction problem. Based on the predicted network, we suggest three recommendation methods to fulfill different users’ requirements.

3) Experiments on the largest real-world service ecosystem ProgrammableWeb, demonstrate that our approach can effectively predict the evolution of the service ecosystem and offer helpful recommendations.

The rest of this paper is organized as follows. Section 2 presents the network prediction based approach for recommendation, including the motivation, the network series model to describe the evolution of a service ecosystem, and the formal definition of the problem; Section 3 presents a comprehensive network analysis including static and evolutionary, on the aforementioned network series model. A rank aggregation method is presented in Section 4 to predict the future behavior of the service network. Section 5 shows the three network based recommendations. Section 6 surveys the related works and Section 7 concludes the paper and discusses the future work.

### III. THE OVERALL NETWORK PREDICTION METHODOLOGY

#### A. Motivation

Fig. 1 illustrates key players in a service ecosystem and their interactions. *Service providers* publish *services* into the ecosystem; services are classified into different *categories* based on their functionality. Composition *developers* choose one or more services to combine them into a *composition* (aka, mashup, workflows).

![Fig. 1. The conceptual model for a service ecosystem consisting of services, service providers, compositions and composition developers.](image)

In a service ecosystem, typically there are many services with similar functions, and therefore developers face the problem of service selection. Taking the service ecosystem ProgrammableWeb as an example, there are more than 300 services offering the “Map” related functionality. A straightforward and evolution-agnostic method is to recommend the services with top popularity based on the occurrence in the whole data set. However, as the ecosystem is
continuously evolving, we observed that over time some services upgrade their function and API; some services become more popular while others become obsolete; even for services that used to be popular historically, they may become perished. Our empirical study shows that 18% services in the whole PW services set, such as “Google Buzz”, “Yahoo Maps”, and “AOL Video” are no longer available as of September 2013 [25]. “Technorati” with 13th popularity historically is unavailable now, and no compositions invoke it since February 2010. Similarly, “Digg” ranked 14th in all time popularity but is never invoked since February, 2012. This observation inspires us to consider the evolution pattern to filter out the obsolete services such as “Technorati” and “Digg”, for a more up-to-date recommendation.

Also, services will collaborate with each other over time and some will form the composition patterns. For example, services “Twilio” and “Twilio SMS”, services “Amazon S3” and “Amazon EC2”, are usually composed together, respectively.

Furthermore, most of the developers only have a fuzzy, but not a clear goal about the composition to be made. For example, a developer may want to make a map-related composition but having decided little about the detail functionality. The evolution pattern can help the developers to understand the usage patterns in the ecosystem and help them to select the right services at a particular moment to construct valuable compositions.

Motivating by the dynamics in a service ecosystem, incorporating its evolution mechanism is necessary to filter out obsolete services, recommend up-to-date composition patterns and guide developers to form valuable compositions. With this insight, we propose a novel network prediction based approach to take the evolution mechanism into account for service recommendation. In the following sections, we firstly introduce the network model for ecosystem evolution and three network based recommendation methods.

B. Network Model for Ecosystem Evolution

To describe the network models more clearly, we summarize the network notations in Table I.

1) Service Network Model

For each composition, we build a link between it and the services invoked within it. Thus we can get a composition-service network as a bipartite graph. For each service pair occurred in a composition, we build a link between them and derive a concurrence service-service network. Fig. 2 shows an example of these two networks. Here we formally define them as follows:

**Definition 1 (Composition-Service Network, CSN)** The composition-service network is a bipartite graph $CSN = (C, S, E'' \subseteq C \times S)$. $E''$ is where $C$ refers to the compositions, $S$ refers to the services and $E'' = \{(c, s) \mid c \in C, s \in S\}$ refers to the invoking relations between compositions and services.

**Definition 2 (Service-Service Network, SSN)** The service-service network is a homogeneous network $SSN = (S, E')$. $E' = \{(u, v) \mid u \in S\}$ refers to the invoked frequency of service $u$. $E' = \{(u, v, w) \mid u, v \in S\} \text{, } w \in S$ represents the service concurrence relations and $(u, v, w)$ means that service $u$ and service $v$ are invoked together in $w$ compositions.

2) Service Network Series Model

Given a series of consecutive time intervals $t_1, t_2, \ldots, t_m$, we get the compositions published in each interval $t_i$ as well as the services invoked in these compositions, to construct a Snap $CSN_G^{\text{map}}(t_i)$: then for each snap CSN, we generate a Snap $SSN_G^{\text{map}}(t_i)$. Thus we can consider the service ecosystem as the snap SSN series and its evolution can be defined as the evolution of the snap SSN along the time. The prediction task for the ecosystem evolution can be transformed into a temporal network prediction task as:

**Definition 3 (Temporal Network Prediction for Service Ecosystem Evolution)** Given a series of consecutive time intervals $t_1, t_2, \ldots, t_m$ and the snap SSN series $G^{\text{map}}_\text{SSN}$, predict the services and their concurrence relations in the snap SSN in the future.

In order to solve the network prediction task, we calculate

![Composition-Service Network and Service-Service Network](image-url)
$M_{\text{sum}}^{\infty}(t_i, l, \theta)$ as the sum of the Corresponding adjacency matrices for all the snap SSNs in the past network series (PNS):

$$M_{\text{sum}}^{\infty}(t_i, l, \theta, u, v) = \sum_{p=\max(0,t_i-l)}^{t_i-1}(1-\theta)^{t_i-p}M_{\text{sum}}^{\infty}(t_p, u, v) \quad (1)$$

Here if we set the decay parameter $\theta = 0$, all the relative snap SSNs are considered as the same.

Similarly, we calculate the corresponding adjacency matrix for $G_{\text{sum}}^{\infty}(t_i, g)$ as:

$$G_{\text{sum}}^{\infty}(t_i, l, \theta, u, v) = \sum_{p=\max(0,t_i-l)}^{t_i-1}(1-\theta)^{t_i-p}G_{\text{sum}}^{\infty}(t_p, u, v) \quad (2)$$

Fig 3 shows the illustration of the relations among these networks.

![Fig. 3. The illustration for snap SSN, sum SSN and ground truth SSN.](image)

3) **Network Evolution Patterns**

As shown in Fig 4 and Table II, given a time interval $t_i$ and the ground truth SSN $G_{\text{sum}}^{\infty}(t_i, g)$, we can identify the following evolution patterns based on $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

![Fig. 4. The illustration for evolution patterns in a service ecosystem.](image)

**TABLE II**

<table>
<thead>
<tr>
<th>DENOTATIONS OF EVOLUTION PATTERNS IN SERVICE ECOSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Service Concurrence Relations</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

a) **Evolution Patterns of Services**

**Re-used Services** (RUS): refer to the services in $G_{\text{sum}}^{\infty}(t_i, g)$ which are also in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

$$RUS(t_i, l) = \{u \mid u \in G_{\text{sum}}^{\infty}(t_i, g) \& u \in G_{\text{sum}}^{\infty}(t_i, l, \theta)\} \quad (3)$$

**Cold-start Services** (CSS): refer to the services in $G_{\text{sum}}^{\infty}(t_i, g)$ but not appear in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

$$CSS(t_i, l) = \{u \mid u \in G_{\text{sum}}^{\infty}(t_i, g) \& u \not\in G_{\text{sum}}^{\infty}(t_i, l, \theta)\} \quad (4)$$

b) **Evolution Patterns of Service Concurrence Relations**

**Re-used Service Concurrence Relations** (RUSC): refer to the links in $G_{\text{sum}}^{\infty}(t_i, g)$ which are also in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

$$RUSC(t_i, l) = \{(u, v) \mid (u, v) \in G_{\text{sum}}^{\infty}(t_i, g) \& (u, v) \in G_{\text{sum}}^{\infty}(t_i, l, \theta)\} \quad (5)$$

**Emerging Service Concurrence Relations** (ESC): refer to the links in $G_{\text{sum}}^{\infty}(t_i, g)$ which are not in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$ but the endpoints both appear in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

$$ESC(t_i, l) = \{(u, v) \mid (u, v) \in G_{\text{sum}}^{\infty}(t_i, g) \& (u, v) \not\in G_{\text{sum}}^{\infty}(t_i, l, \theta)\} \quad (6)$$

**Cold-start Service Concurrence Relations** (CSSC): refer to the links in $G_{\text{sum}}^{\infty}(t_i, g)$ which are not in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$ and at least one of the two endpoints is not in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$.

$$CSSC(t_i, l) = \{(u, v) \mid (u, v) \in G_{\text{sum}}^{\infty}(t_i, g) \& (u, v) \not\in G_{\text{sum}}^{\infty}(t_i, l, \theta)\} \quad (7)$$

For CSS and CSSC, they both contain the services which are never used before so that no prediction algorithms which are solely based on the sum SSN series can give any informative prediction. In this paper, we will focus RUS including RUSC and ESC in this paper. Thus the prediction task for service ecosystem evolution can be further rewritten as follows from the temporal link prediction perspective:

**Definition 4 (Temporal Link Prediction Task for the Service Ecosystem Evolution)** Given the snap SSN series $G_{\text{sum}}^{\infty}$ from $t_1$ to $t_m$, at time interval $t_i$, the task is to find a predictive function $f : S \times S \rightarrow R, S \in G_{\text{sum}}^{\infty}(t_i, l, \theta)$ to infer in what possibility that each service pair $(u, v)$ in $G_{\text{sum}}^{\infty}(t_i, l, \theta)$ will build a concurrence relation in $G_{\text{sum}}^{\infty}(t_i, g)$.

**C. Service Recommendation based on Network**

The output of the prediction task is a list of service pairs each with its possibility to appear in $G_{\text{sum}}^{\infty}(t_i, g)$. Thus we can get the top-$k$ service pairs and then construct the predicted SSN. Based on this network, we can offer three service recommendations for different developers, as shown in Fig 5.

![Fig. 5. Three recommendation methods based on the predicted SSN.](image)
1) Potential Compositions Recommendation

The services in a composition will construct a complete sub-graph in the SSN. Conversely we can fetch the complete sub-graphs from the predicted SSN and each one can be considered as a potential composition. As the potential composition will gain a high possibility to appear in the future, we can recommend them to the developers who have no idea about their compositions, showing them the possible ways to use services together.

2) Top Services Recommendation

In the predicted SSN, we can sort the services by their degrees. Obviously, the services with higher degrees gain higher possibility to be used by the developers in the future. Thus the service ecosystem manager should pay more attention to their performances which can increase the satisfaction of the developers.

3) Service Chains Recommendation for Compositions

The predicted SSN is an undirected graph with a possibility to each edge. Thus for the developers who select some services in the predicted SSN for their compositions, given any two services, the ServiceMap algorithm we proposed in [19] can be used to find the services chain with the highest possibility to inspire the developers to form a composition.

D. Network Prediction Based Approach

Based on the discussion above, we propose our network prediction approach (NPA) to capture the evolution mechanism of the service ecosystem for the recommendation. As shown in Fig 6, the NPA approach consists of three components:

![Network Prediction Approach](image)

Fig. 6. The network prediction approach (NPA) for service and composition recommendation consisting of three components: network analysis, network prediction, and network based recommendation.

1) Network Analysis Component

This component uses the collected historical information to construct the network series for the service ecosystem (P1) including the snap CNS series, the snap SSN series and the time-aware collapsed sum SSN series; then static network analysis methods (P2) are introduced to study the usage pattern in each single network; and the evolution pattern analysis component (P3) is employed to study the dynamic pattern between the adjacent networks.

2) Network Prediction Components

Based on the observations in the network analysis component, this component firstly studies the features that reflect the evolution of the snap SSN series (P4). Then the link prediction method based on rank aggregation (P5) can be used to solve the link prediction task. Finally we get the top-k service pairs to construct the predicted snap SSN for consumers (P6).

3) Network based Recommendation Components

This component offers the three kinds of recommendations based on the predicted snap SSN generated by the prediction components: the potential compositions recommendation (P7), top services recommendation (P8) and service chains recommendation for compositions (P9).

IV. NETWORK ANALYSIS

A. Data Collection

We aim to study the evolution mechanism of the service ecosystem and capture the evolution information for the service and composition recommendation. Thus we collect the data regarding services and compositions from June 2005 to August 2012 in ProgrammableWeb, one of the biggest service ecosystems nowadays. Each service contains its name and the publication date as well as its availability status in August 2012; each composition contains its name, creation date and the list of services invoked within it; each developer contains the name, publishing compositions and the register time. By removing compositions which contain no services and the developers who publish no compositions, we get a collection of 7077 services, 6726 compositions. Table III reports the basic properties of our dataset.

<table>
<thead>
<tr>
<th>Basic Features of the ProgrammableWeb Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of services (#services for short)</td>
</tr>
<tr>
<td>Number of compositions (#compositions for short)</td>
</tr>
<tr>
<td>Average number of services per composition</td>
</tr>
<tr>
<td>Average number of compositions per service</td>
</tr>
<tr>
<td>Services used in at least one composition</td>
</tr>
<tr>
<td>Number of developers with at least one composition</td>
</tr>
</tbody>
</table>

: only consider the services which are invoked in at least one composition

B. Network Construction (P1)

In this paper, we set the time interval as one month and get the compositions published in each month as well as the services involving in these compositions. Then we construct the snap CSN series and the snap SSN series. Thus \( t_1 = \text{June-2005} \) and \( t_2 = \text{August-2012} \).

![Network Size of Snap CSN Series](image)

![Network Size of Snap SSN Series](image)

Fig. 7. Network size of the snap CSN series and the snap SSN series over time.

Fig 7 shows the network size of the snap CSN series and the snap SSN series over time. For each snap CSN, we only consider the services which are invoked in at least one composition. We can observe that for each month, there are about 50–150 compositions published into the ecosystem and about 60–120 services will be invoked in the compositions.

Each composition is published by one developer, so we can
calculate the number of compositions that each developer has published and the number of services that each developer has ever used. Fig 8 shows the result plotted on the log-log axes. From the figure, we can see that both distributions meet the power-law distribution with long tail. Furthermore, though there are more than 50,000 developers in the community, only 2384 developers (4.8%) ever publish compositions in the ecosystem. 97.32% of the developers publish no more than 5 compositions. Thus, we can conclude that most of the developers are the cold start consumers who are not familiar to the ecosystem.

C. Static Network Analysis (P2)

In this section, we engage in some high-level investigation of the static usage patterns in each snap network. In particular, we study the following questions:

For the snap CSN series: How are services used in the compositions in each time interval?

For the snap SSN series: How do services coordinate with each other in each time interval?

1) Composition Patterns in the Snap CSN series

First of all, we construct the sum SSN $G_{ssn}^{sum}(t_{i}, \infty, 0)$, we sort them by their used frequency and then get the QF distribution $f_{i}(i) = j$ which means that there are $j$ services each of which participates in $i$ compositions. Furthermore, we calculate the cumulative distribution (CF) of the edge weight which is formally defined as: $F_{i}(i) = \sum_{k=0}^{i} f_{i}(k) = \text{Total number of services which invoke in more than or equal to } i \text{ compositions.}$

From Fig 10 (a) we can see the CF distribution meets the power law distribution which indicates that some popular services are used frequently while most of the services are not so popular. For example, the most popular service “Google Maps” has been invoked in 2394 compositions while the second most popular one “Twitter” is used in 742 compositions. In fact, for all the services which are invoked by at least one composition, only 23.67% (270) services are reused more than 5 times.

Similarly, for all the edges which refer the service concurrence relations in $G_{ssn}^{sum}(t_{i}, \infty, 0)$, we sort them by their weight and calculate the QF distribution, and then we get the CF distribution of the edge weight. Fig 10 (b) shows that it also meets the power law distribution, which indicates that a few popular services concurrence relations will gain higher possibility to be reused. For example, “Google Map” with “Twitter”, “Twitter” with “Facebook”, “Google Map” with “Flickr”, “Flickr” with “YouTube” and “YouTube” with
“Last.fm”, “Twilio” with “Twilio SMS”, “Amazon S3” with “Amazon EC2” are some popular service pairs [18].

Furthermore, for each time interval we get the sum SSN $G_{ssn}^{uw}(t,\infty,0)$. Similarly we get the CF distribution of services and the CF distribution of edge weights for each network. We find that all of them meet the power-law distribution. Then we build a fitted curve for each CF distribution in the log-log axes and get its value of the exponent (VE). Thus we can get the evolution of VE over time, which is shown in Fig 10 (c) and Fig 10 (d). As it can be seen, the VE curve for service is decreasing which indicates that the power law effect is declining. However, the VE curve for the edge weight is increasing which indicates that the power law effect is strengthening.

**D. Evolution Pattern Analysis (P3)**

As we aim to figure out the possibility that each service pair in $G_{ssn}^{uw}(t,\infty,\theta)$ will construct an edge in $G_{ssn}^{uw}(t,\delta,g)$. Thus given the $G_{ssn}^{uw}(t,\infty,\theta)$, we can calculate the percentage of the different evolution patterns in $G_{ssn}^{uw}(t,\delta,g)$. In this paper we name them as the propagate rate (PR).

$$PR(t,\infty,\delta,g,RUS) = \frac{|RUS(t,\delta)|}{|G_{ssn}^{uw}(t,\delta,g)|}$$  \hspace{1cm} (8)

$$PR(t,\infty,\delta,g,RUSC) = \frac{|RUSC(t,\delta)|}{|G_{ssn}^{uw}(t,\delta,g)|}$$  \hspace{1cm} (9)

$$PR(t,\infty,\delta,g,ESC) = \frac{|ESC(t,\delta)|}{|G_{ssn}^{uw}(t,\delta,g)|}$$  \hspace{1cm} (10)

$$PR(t,\infty,\delta,g,CSSC) = \frac{|CSSC(t,\delta)|}{|G_{ssn}^{uw}(t,\delta,g)|}$$  \hspace{1cm} (11)

Here $|G_{ssn}^{uw}(t,\delta,g)|$ refers to the number of services in the ground truth network $G_{ssn}^{uw}(t,\delta,g)$ and $|G_{ssn}^{uw}(t,\delta,g)|$ is the number of edges. Note that $\theta$ does not affect the propagate rate. Here we set $\delta = \infty$ and $g = 1$. As seen in Fig. 11, about 80% of the services in each $G_{ssn}^{uw}(t,\delta)$ are RUS, and 80% of the edges are either RUSC or ESC. In fact, the average percentage of the CSSC in our study period is only 19.65%. Thus we can conclude that the sum SSN carries the evolution information which can be used to predict the evolution of the snap SSN.

Similarly, we get the propagate rates for different $\delta$ and then we define the relative propagate rates (RPR) as:

$$RPR(t,\infty,\delta,g,*) = \frac{PR(t,\infty,\delta,g,*)}{PR(t,\infty,\delta,g,\#)}$$ \hspace{1cm} (12)

Here $\# = \{RUS, RUSC\}$. Also

$$RPR(t,\infty,\delta,g,RUSC+ESC) = \frac{PR(t,\infty,\delta,g,RUSC)+PR(t,\infty,\delta,g,ESC)}{PR(t,\infty,\delta,g,RUS)+PR(t,\infty,\delta,g,ESC)}$$  \hspace{1cm} (13)

Then the Marginal RPR (MRPR) for different $\delta$ is defined as:

$$MRPR(t,\infty,\delta,\#) = RPR(t,\infty,\delta-1,\#) - RPR(t,\infty,\delta,\#)$$  \hspace{1cm} (14)

Here $\# = \{RUS, RUSC, RUSC + ESC\}$, $RPR(t,\infty,\delta,0,\#) = 0$ and $l = 1,\cdots i$ Note that we consider $RUSC + ESC$ instead of $ESC$ because some ESC for a smaller $\delta$ will turn out to be RUSC for a larger one.

Finally we can get the average RPR (ARPR) and the average MRPR (AMRPR) during our study period, which are shown in Fig 12. We can see that the ARPR to $l$ is a monotonically increasing function which value is between 0 and 1. This indicates that the more snap SSN we taking into account, the more relative historical information can be used for prediction. However, the AMRPR to $l$ is a monotonically decreasing function. This indicates that the marginal revenue for considering the older snap SSN is decreasing. The older snap SSN is less likely to be relevant for determining the future SSN. In fact, when we set $l = 10$, the ARPR for RUS and RUSC+ESC are both more than 90%, also all the AMRPR are less than 0.05.

**E. Summary for Network Analysis**

From the observations shown above, we can draw the conclusions for the usage pattern and evolution phenomenon in the service ecosystem as follows:

- a) Most of the compositions contain no more than 5 services. And though the average number of services invoked within compositions is increasing, the value is still no more than 4. Thus when we extract the potential compositions from the predicted SSN, we should pay more attention to the ones with no more than 4 services.

- b) The power law phenomenon is common and there should be a preferential attachment for the service selection as well as the service pair reused. The power law effect for the service pair reused is strengthening while for the service selection, it is weakening. All these indicate that the topology of the SSN will carry hidden knowledge and we should take it into account for the prediction task.

- c) Though the sum SSN contains information about 80% of the RUS, as well as the RUSC and ESC for the following snap
SSN, there is a significant attenuation effect: the older usage history is less likely to be relevant for determining the future re-use of services than the recent ones. This indicates that though we can use the historical information for network prediction, we should consider the temporal pattern over time.

V. NETWORK PREDICTION

A. Evolution Features Study (P4)

Based on these observations shown above, we can conclude that the usage pattern of the nodes and the edges, including the static and dynamic properties, can help for the evolution prediction. In this section, we design the features relevant to the evolution, and then quantify their influence to the evolution of the snap SSN. Firstly we introduce some notations which help to understand the features. Let $N_i(u)$ be the set of neighbors of the service $u$ in $G_{sum}^{\text{ssn}}(t_i, l, \theta)$; $w_i(u, v)$ is the edge weight between service $u$ and service $v$; paths$(u, v, n)$ represents the paths between $u$ and $v$ with length $n$.

1) Features Definition

a) Current Sum Edge Weight (SEW)

This factor simply uses $w_i(u, v)$ as the score of the service pair $(u, v)$. If these two services are never used together in the same composition before, the score will be zero. This factor captures the basic preferential attachment for the service pair selection.

b) Common Neighbors (CN)

This factor calculates the score based on the number of common neighbors of the service pair. Here we use the weighted version which is defined as:

$$CN_i(u, v) = \sum_{k \in N_i(u) \cap N_i(v)} w_i(u, k) \times w_i(v, k)$$

(15)

The CN score has been widely used in social science regarding the basic social psychological theory: structural balance theory [26]. For the SSN, the structural balance property supposes that if two services have many common neighbors, they will construct a concurrence relation in future.

c) Adamic/Adar (AA)

This factor is extended from the CN factor by assigning a higher weight to the common neighbor who has a smaller degree [27]. Here we use the weighted version:

$$AA_i(u, v) = \sum_{k \in N_i(u) \cap N_i(v)} \frac{w_i(u, k) \times w_i(v, k)}{\log(|N_i(k)|)}$$

(16)

d) Reciprocal Last Edge Occurred Time Stamps (RLE)

The hypothesis for RLE is that the consumer will pay more attention to the more recent links. If the service pair has not been used together for a long time, it is rarely to be used together in future. Thus we can define the score as:

$$LE(u, v, t_i) = \begin{cases} 0 & (u, v) \in G_{\text{sum}}^{\text{ssn}}(t_i) \\ i-j & (u, v) \in G_{\text{sum}}^{\text{ssn}}(t_i), (u, v) \not\in G_{\text{sum}}^{\text{ssn}}(t_i), i-j \leq k \leq i \\ i+1 & (u, v) \not\in G_{\text{sum}}^{\text{ssn}}(t_i), i-j \leq k \leq i \end{cases}$$

(17)

$$RLE(u, v, t_i) = \frac{1}{LE(u, v, t_i) + 1}$$

(18)

e) Vertex Preferential Attachment (VPA)

This factor is motivated by the preferential attachment phenomena for the service reused we observe before.

$$VPA(u, v) = \log(\sum_{i \leq k \leq n} w_i(u, k)) + \log(\sum_{j \leq k \leq n} w_i(v, j))$$

(19)

The basic intuition is that: the more popular services are more likely to be reused so links between them are more likely to appear in future.

f) Reciprocal Last Vertex Occurred Time Stamps (RLV)

Similarly, the longer the service makes its last connection, the less possibility that it will be reused in future. Thus we can define the score for each service pair $(u, v)$ as:

$$RLV(u, v, t_i) = \begin{cases} 0 & (u, v) \in G_{\text{sum}}^{\text{ssn}}(t_i) \\ i-j & (u, v) \in G_{\text{sum}}^{\text{ssn}}(t_i), (u, v) \not\in G_{\text{sum}}^{\text{ssn}}(t_i), i-j \leq k \leq i \\ i+1 & (u, v) \not\in G_{\text{sum}}^{\text{ssn}}(t_i), i-j \leq k \leq i \end{cases}$$

(20)

$$RLV(u, v, t_i) = \frac{2}{LV(u, t_i) + LV(v, t_i) + 1}$$

(21)

g) Page Rank (PR)

PageRank algorithm [28] has been widely used to measure the importance of web pages. Similarly, we employ the PageRank method to measure the importance of each service in the network and suppose that the two important services gain a high possibility to form a link.

$$PR_i(u, v) = p_i(u) + p_i(v)$$

(22)

Here $p_i(u)$ refers to the page rank score for service $u$ in the sum SSN.

h) Betweenness Centrality (BC)

Betweenness is a centrality measure in the traditional sense [29] and a higher centrality score for the service indicates that it can reach others in a relatively short path and it will gain a higher possibility to be reused in future. Thus we define the betweenness centrality score for the service pair $(u, v)$ as:

$$BC_i(u, v) = B_i(u) + B_i(v)$$

(23)

Here $B_i(u)$ refers to betweenness centrality score for service $u$ in the sum SSN.

Fig. 13. Fundamental features used for prediction. Topological features refer to the features which are just based on the structure of a single network while the temporal features are calculated based on the network series; Edge-centric features refer to the features which considers the service pair as a whole while the node-centric features will firstly consider each node’ pattern and then sum them up.

Obviously, SEW and RLE will assign the same score to all the service pairs without forming a link before, AA and CN are based on the common neighbors of the given service pair. However, VPA, PR, BC and RLV firstly consider the status of each service and then combine them up. Thus we term SEW,
RLE, AA and CN as edge-centric features and VPA, PR, BC and RLV as node-centric features.

Furthermore, SEW, CN, AA, VPA, PR, BC are calculated based on a single given sum SSN while RLE and RLV will take all the snap SSNs in PNS into account. So we name RLE and RLV as temporal features and the others as topological features.

2) Feature Influence to Evolution

In order to quantify the influence of the given features to the prediction task, for each feature we construct a simple binary classifier which splits the whole samples into two parts by a threshold:

Given a threshold \( x \), for each service pair \((u,v)\), if \( f(u,v) \geq x \) then \( y(u,v) = 1 \); else \( y(u,v) = 0 \)

Here \( f(u,v) \) refers to the score of a certain feature for \((u,v)\), \( y(u,v) = 1 \) means that we label the service pair \((u,v)\) as appearing in the next time interval while \( y(u,v) = 0 \) means conversely.

Then we consider the recall/fallout ratio [30] to quantify the influence which is defined as:

\[
\text{recall/fallout} = \frac{\text{true positives} / \text{positives}}{\text{false positives} / \text{negatives}} = \frac{\text{true positives} / \text{false positives}}{\text{positives} / \text{negatives}}
\]

(24)

Here true positives refers to the number of service pairs which are correctly predicted to appear, false positives is the number of service pairs which are erroneously predicted to appear, positives means the number of all the service pairs appear in future while negatives means conversely.

Due to the space limitation and the consistency of results, we just take \( G_{\text{sum}}(t_{156},10,0.0) \) as an example. As shown in Fig. 14, we get the consistently incremental curves for all the chosen features, which indicate that the service pair will gain a higher possibility to form a link in the next snap SSN if it owns a higher value. For example, if the service pair has a CN score larger than 5, it is 130 times more likely to be invoked in the same composition than not.

![Fig. 14. The Recall/Fallout Ratio Curves plotted on the log-log axes for the features we selected. The higher the ratio is, the better the feature can reflect the evolution of the network. Here we set \( i = 86, l = 10, \theta = 0, g = 1 \)](image)

B. Network Prediction based on Rank Aggregation (P5)

As SEW, CN, AA, VPA, PR, BC, RLE and RLV are all relevant to the evolution of the SSN, straightforwardly, for each factor, we can calculate the score for each service pair in \( G_{\text{sum}}(t, l, \theta) \) and rank them according to the score. The service pair with a higher score is considered as the one with a higher possibility to form a link in the future. In order to leverage the different features, we introduce a method based on rank aggregation [31] to study the link prediction problem.

1) Rank Aggregation based Link Prediction

Let \( p_i^j(u,v) \) denotes a vertex pair \((u,v)\) in \( G_{\text{sum}}(t, l, \theta) \). For simplicity but without confusion, we can use \( p_i^j \) for short. Here \( i \) is used to index different vertex pairs in this network. Obviously \( i = 1, \ldots, N \) where \( N = \frac{1}{2} |V| (|V| - 1) \) \( |V| \) is the number of services in \( G_{\text{sum}}(t, l, \theta) \). For each feature, we calculate the score for each service pair and then rank them based on the score. Thus we can get eight primary ranking for each service pair, \( r_i^j = \{p_i^j, \ldots, p_i^{j,N}\}, j = 1, \ldots, 8 \). Based on the position in the ranking, we assign an extended Borda’s score [32] for each service pair:

\[
B(r_i^j, p_i^j) = \frac{N - \pi(r_i^j, p_i^j)}{N}
\]

(25)

Where \( \pi(r_i^j, p_i^j) \) refers to the number of service pairs with a lower score than \( p_i^j \) in the primary ranker \( r_i^j \).

Finally we use the linear function \( f \) to combine all the primary rankers to get an aggregated score for each service pair:

\[
f' = R'W
\]

(26)

Where \( R' = B(r_i^j, p_i^j) \) and \( W = [w_1, w_2, \ldots, w_8]^T \). In this paper, we just set \( W = [\frac{1}{8}, \frac{1}{8}, \ldots, \frac{1}{8}]^T \). As the score is based on the aggregated ranker, we name this method as AR.

2) Performance Evaluation

In order to evaluate the effectiveness, we build the ROC (Receiver Operating Characteristic) curves by evaluating the service pairs from top to bottom of each ranker and then calculate the Area under the ROC curve (AUC) based on the Wilcoxon-Mann-Whitney algorithm [33]. Here AUC is adopted as the measure of performance because of its higher robustness to class distribution anomalies in the imbalanced dataset [34].

<table>
<thead>
<tr>
<th>ALGORITHMS FOR COMPARISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
</tr>
<tr>
<td>VPA</td>
</tr>
<tr>
<td>PR</td>
</tr>
<tr>
<td>AA</td>
</tr>
<tr>
<td>BC</td>
</tr>
<tr>
<td>CN</td>
</tr>
<tr>
<td>RLV</td>
</tr>
<tr>
<td>RLE</td>
</tr>
<tr>
<td>SEW</td>
</tr>
</tbody>
</table>

First of all, we study the effectiveness of the rank
aggregation. For each time interval, we calculate the AUC for each primary ranking and the aggregated ranking shown in Table IV. Finally we get the average AUC for each ranking. Here we take \( l = 10, \theta = 0 \) as an example and report the result in Table V.

As it can be seen, SEW and RLE achieve the worst performance because they assign the same score to all the previously non-existing service pairs that they cannot deal with the ESC patterns. VPA, PR, AA, BC, and CN gain the comparable performance because they just consider the topological information among the network. The aggregation ranker (first row) outperforms all the other primary rankers because it can leverage the topological and time-aware information, the edge-centric and the node-centric information together.

<table>
<thead>
<tr>
<th>( g )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>0.9125</td>
<td>0.897</td>
<td>0.9032</td>
<td>0.9064</td>
<td>0.8998</td>
<td>0.8951</td>
<td>0.8923</td>
<td>0.8897</td>
</tr>
<tr>
<td>VPA</td>
<td>0.894</td>
<td>0.8867</td>
<td>0.8829</td>
<td>0.8802</td>
<td>0.8781</td>
<td>0.8753</td>
<td>0.8724</td>
<td>0.8698</td>
</tr>
<tr>
<td>PR</td>
<td>0.8886</td>
<td>0.8804</td>
<td>0.8759</td>
<td>0.8728</td>
<td>0.8701</td>
<td>0.8669</td>
<td>0.8639</td>
<td>0.8611</td>
</tr>
<tr>
<td>AA</td>
<td>0.8961</td>
<td>0.8807</td>
<td>0.8778</td>
<td>0.8748</td>
<td>0.8725</td>
<td>0.8693</td>
<td>0.8659</td>
<td>0.8628</td>
</tr>
<tr>
<td>BC</td>
<td>0.8942</td>
<td>0.8764</td>
<td>0.8714</td>
<td>0.8682</td>
<td>0.8658</td>
<td>0.8626</td>
<td>0.8641</td>
<td>0.8575</td>
</tr>
<tr>
<td>CN</td>
<td>0.8841</td>
<td>0.8787</td>
<td>0.8786</td>
<td>0.8729</td>
<td>0.8706</td>
<td>0.8674</td>
<td>0.8664</td>
<td>0.8669</td>
</tr>
<tr>
<td>RLV</td>
<td>0.8218</td>
<td>0.811</td>
<td>0.8025</td>
<td>0.7972</td>
<td>0.7923</td>
<td>0.787</td>
<td>0.7833</td>
<td>0.7805</td>
</tr>
<tr>
<td>RLE</td>
<td>0.7667</td>
<td>0.7483</td>
<td>0.7359</td>
<td>0.7267</td>
<td>0.7193</td>
<td>0.7119</td>
<td>0.7054</td>
<td>0.6998</td>
</tr>
<tr>
<td>SEW</td>
<td>0.762</td>
<td>0.7429</td>
<td>0.7304</td>
<td>0.7212</td>
<td>0.7137</td>
<td>0.7064</td>
<td>0.6999</td>
<td>0.6943</td>
</tr>
</tbody>
</table>

Fig. 15. Impact of window size. (a) Average AUC of each ranker for different \( l \). (b) Sample size (Number of service pairs) for different \( l \) in the study period.

Secondly, in order to study the impact of \( l \) to the link prediction, we set \( g = 1, \theta = 0 \) and assign \( l = 1, 2, 3, 4, 5, 10, 20, \infty \). For each \( l \), we calculate the average AUC during our study period and then inspect the marginal utility. As it can be seen in Fig 15, AR achieves the best performance for all the different window sizes. Also for most of the fundamental features, larger \( l \) brings a higher AUC which indicates that taking more historical information can help for evolution prediction.

<table>
<thead>
<tr>
<th>( g )</th>
<th>AR</th>
<th>SEW</th>
<th>CN</th>
<th>AA</th>
<th>VPA</th>
<th>RLE</th>
<th>RLV</th>
<th>PR</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.064</td>
<td>0.063</td>
<td>0.110</td>
<td>0.109</td>
<td>0.031</td>
<td>0.076</td>
<td>0.268</td>
<td>0.286</td>
<td>0.059</td>
</tr>
<tr>
<td>5</td>
<td>0.022</td>
<td>0.030</td>
<td>0.031</td>
<td>0.030</td>
<td>0.025</td>
<td>0.024</td>
<td>0.083</td>
<td>0.020</td>
<td>0.029</td>
</tr>
<tr>
<td>10</td>
<td>0.013</td>
<td>0.026</td>
<td>0.017</td>
<td>0.017</td>
<td>0.020</td>
<td>0.013</td>
<td>0.042</td>
<td>0.012</td>
<td>0.013</td>
</tr>
<tr>
<td>\infty</td>
<td>0.006</td>
<td>0.016</td>
<td>0.002</td>
<td>0.001</td>
<td>0.016</td>
<td>-0.005</td>
<td>0.020</td>
<td>0.001</td>
<td>0.000</td>
</tr>
</tbody>
</table>

However, the number of services in \( G_{\text{num}}(t, l, \theta) \) is increasing with the growth of \( l \), which results in a rapidly growth of the samples for the prediction, while the marginal utility is decreasing for all the features. From Table VI we can see that increasing \( l \) from 20 to \infty gains very little improvement for AR, CN and BC, it evenly reduces the accuracy for AA, VPA and PR. This indicates that the older snap SSN network will bring more noise for the prediction. Thus there should be a trade-off for the window size selection.

<table>
<thead>
<tr>
<th>( g )</th>
<th>AR</th>
<th>SEW</th>
<th>CN</th>
<th>AA</th>
<th>VPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.9125</td>
<td>0.762</td>
<td>0.8841</td>
<td>0.8861</td>
<td>0.894</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9123</td>
<td>0.762</td>
<td>0.8836</td>
<td>0.8855</td>
<td>0.8928</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9122</td>
<td>0.762</td>
<td>0.8828</td>
<td>0.8848</td>
<td>0.8916</td>
</tr>
<tr>
<td>0.15</td>
<td>0.912</td>
<td>0.762</td>
<td>0.8821</td>
<td>0.8841</td>
<td>0.8906</td>
</tr>
<tr>
<td>0.20</td>
<td>0.9119</td>
<td>0.762</td>
<td>0.8812</td>
<td>0.8833</td>
<td>0.8896</td>
</tr>
</tbody>
</table>

Thirdly, we study the impact of \( \theta \) to the prediction. Similarly, we set \( g = 1, l = 10 \) and then assign different value to \( \theta \). The result is reported in Table VII. As it can be seen, \( \theta \) has a very slight affection to the performance. Note that we don’t consider PR, BC, RLV and RLE because these features are not based on the weight of edge so \( \theta \) will not affect the performance.

Fourthly, for the top-\( k \) service pairs, we get the number of unique services as well as their status and then calculate the percentage of the obsolete services (POS). Here we only take the VPA as the baseline because it achieves the best
performance in the primary rankers. Fig 16 (a) reports the result for \( i = 87 \) because we only get the services’ status in \( t_\theta = \) August-2012. Setting \( l = \infty \), \( \theta = 0 \) which means that we take all the historical information into account, \( AR \) gains a 50% reduction of \( POS \) comparing with \( VPA \). This indicates that \( AR \) could effectively filter out the obsolete services. After setting the window size as 10, the \( POS \) for \( AR \) and \( VPA \) are both decreased. \( VPA \) gains a better performance for the top 390 unique services and \( AR \) achieves a lower \( POS \) for the long tail.

Finally, in order to show the performance of the rank aggregation method, we consider several baselines:

(1) \( FS \) [35]: This method constructs a link occurrence frequency series for each service pair which has been used before and then employs the ARIMA model to get the time series forecast score for each service pair as the final score.

(2) \( RF \) [36]: This method firstly calculates the factor values for each service pair and then uses a random forest to combine them to get the final score.

(3) \( NB \): Instead of using a random forest in \( RF \), for \( NB \), we use the naive Bayes for the combination and consider the positive possibility for each service pair as the final score.

Due to the computational complexity, we use R\(^4\), a famous software for statistical computing and graphics to build the ARIMA model for the frequency series. Also we employ Weka [37] to build the random forest and naive Bayes. As it can be seen in Fig 16 (b), our method \( (AR) \) achieves a better performance than all the other methods. Particularly, \( AR \) gains about 20% improvement than \( FS \). This is because \( FS \) only considers the evolution of the link occurrence frequency which indicates that it cannot predict ESC in the SSN. Comparing with \( RF \), \( AR \) achieves a 50% improvement because it re-ranks the service pairs based on their positions in each primary ranking so that it can eliminate the difference between the features. Also, \( AR \) gains a 5% improvement than \( NB \).

C. Predicted SSN Construction (P6)

In order to construct the predicted SSN, we get the service pairs with the top-\( k \) aggregated ranking and build the combined network. Thus the predicted SSN network is an undirected network with a possibility for each edge. Here we employ the \( AR \) method to calculate the possibility for each service pair. Fig. 17 shows the illustrated predicted SSN for the snap SSN in \( t_i = t_{\text{stag}} \), when we set the window size \( l = 10 \), the decay control \( \theta = 0 \) and only consider the top \( k = 1000 \) service pairs.

VI. NETWORK BASED RECOMMENDATION

We develop a prototype system based on JUNG\(^5\) to implement the services and compositions recommendation based on the predicted SSN for the developers which consists of three parts:

(1) Parameter Setting: Initial the parameters such as \( l, \theta, t_i, k \) etc. for the prediction and recommendation.

(2) Graph View: Show the visualization of the predicted SSN.

(3) Recommend Result: Recommend potential compositions, top services, and service chains for the developers.

\(^{4}\)http://www.r-project.org/

\(^{5}\)http://jung.sourceforge.net/
possibility. The lower part is the figure for the selected composition. The width of the edge refers to the possibility that the service pair will appear in future.

B. Top Services Recommendation

For the services in the predicted SSN, we sort them by their network degrees and the service with a higher network degree is considered to be more popular in the future.

Fig. 20. Services in the predicted SSN sorted by their network degree.

C. Service Chains Recommendation for Compositions

Given selected services in the predicted SSN, the ServiceMap [19] approach is employed to find the service paths with the highest possibility to help the developers to finish their compositions. For example, as shown in Figure 21, the developer selects the services “Amazon eCommerce”, “del.icio.us”, “Google Places”, “Dropbox” and “Flickr”, the ServiceMap approach will suggest the service paths as well as other latent services such as “foursquare”, “Google Maps” and “Twitter” to construct a local service network.

Fig. 21. The illustration of the service chain recommendation. Upper part is the service list in the predicted SSN which can be selected by the developers. Lower part is the suggested service chains in which the dark gray nodes are the selected services and the light gray nodes refer to the internal services for the chains.

Semantic-based approaches recommend services based on semantic compatibility. MashupAdvisor employed an AI planner and the semantic matcher to compute the service composition [9]. C. Zhou etc. [39] constructed a semantic Bayesian network based on the semi-supervised learning method to recommend the services for compositions.

QoS-based approaches recommend the services with a high quality for the composition. J. Cao etc. [10] presented a hybrid collaborative filtering algorithm based on QoS to provide bidirectional recommendation for providers and consumers. Z. Zheng etc. [11] proposed two personalized QoS ranking prediction approaches to calculate the QoS of the services for different users and then recommended the services with a higher QoS quality for consumers.

Social network based approaches are becoming a hot research topic to recommend services for consumers based on the network perspective. W. Xu etc. [12] proposed a social-aware service recommendation approach which leveraged the multi-dimensional social relationships among potential users, topics, compositions and services. B. Cao etc. [13] balanced the user interest and the social network among services to recommend services. C. Zhang etc. [15] presented the correction policy and the precision policy for computing user similarity to improve the accuracy of the recommendation. A. Maaradji [14] introduced the SoCo framework based on the users and the social network built from the interactions between users and services, and the different services compositions.

<table>
<thead>
<tr>
<th>Information</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>QoS</td>
</tr>
<tr>
<td>[9]</td>
<td>√</td>
</tr>
<tr>
<td>[39]</td>
<td>√</td>
</tr>
<tr>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>[12]</td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td></td>
</tr>
<tr>
<td>[14]</td>
<td></td>
</tr>
<tr>
<td>[15]</td>
<td></td>
</tr>
</tbody>
</table>

Information: refers to the information employed for the recommendation, which includes Semantic, QoS, Network, User Interest, User Query and Evolution Pattern.

Property: refers to the properties the methods can offer. OSF is short for obsolete service filtering; NQ is short for non-query which means that no user input query is required; CS is short for cold start which means that the method could offer recommendation to the cold start developers.

As we discussed before, most of the services on the web nowadays only provide human-readable documentation [40] and the services are evolving [16] and some may become obsolete over time. Also most of the service ecosystem does not offer the QoS information and it is resource-intensive to fetch them. Furthermore, most of the developers in the service ecosystem are cold start users who rarely publish compositions before that the preference information of the developer is lack. In practice, they are not always clear about the goal and they may not offer a query for their compositions.

Thus from a different perspective, we propose a network prediction approach (NPA) which employs the network information among services as well as the evolution patterns for
the recommendation. This enables us to filter out obsolete services (OSF) and no composition queries are needed (NQ).

Note that NPA does not take the developers’ information into account thus we can offer recommendation for cold start developers (CS). In fact it can be straightforward to combine with collaborative filtering to offer personalized recommendation and we leave this for future work.

B. Network Analysis

Network analysis is a powerful tool to understand the insight of the networks [23]. Many studies have been done to analyze networks such as scientists’ correlation network, mobile phone calling network [41], the Internet [42], the email network [43] and the web resources [2]. For example, in [2], the authors build an association link network to present the associated relations among resources and study the scale-free properties, small-world feature and the average weight and weight distribution.

Recently because of the rapid increasing of services over internet, some works have been done to study the relationship between services. Yu and Woodard [44] presented a preliminary result in studying the network properties such as scale-free and small world in ProgrammableWeb. P. Kungsas and M. Matskin [45] constructed a service network to study the composability of the services and pointed out most of the compositions contain only a few services. Weiss and Gangadharan [46] built the API affiliation network and analyzed the complementary nature of APIs in ProgrammableWeb. They found that the complexity of the mashups drives the development of the mashup platform. Wang et al [47] emphasizes on mining mashup community from users’ perspective by analyzing the User-API and User-Tag network in ProgrammableWeb. Our previous works [50] introduce the network analysis method to study the usage pattern of services in the service-workflow system, i.e. myExperiment [20] and then a GPS-like assistance tool Service-Map [19] is used to help understanding the usage patterns as well as to recommend the service operation chains for the users.

However few papers pay attention to the dynamic patterns of the ecosystem. In this paper we introduce the network series model to formally define the evolution of the service ecosystem and transform the prediction problem into a network prediction problem. We focus on the usage patterns and the evolution phenomenon in the service ecosystem to extract features for the evolution prediction.

C. Link prediction

Since introduced by Liben-Nowell and Kleinberg [24], link prediction has become an active research area in computer science. Liben-Nowell and Kleinberg [24] studied different predictors, such as graph distance, preferential attachment, PageRank for the prediction of social network. Lichtenwalter, Lussier and Chawla [48] introduced a random forest-based approach to combine the static topological features. Z. Huang and D. K. Lin [35] constructed the link occurrence frequency series for each service pair and then deploy the ARIMA model on the series to the temporal evolutions pattern into account so that it’s more appropriate for the evolving network. Soares and Prudencio [49] considered the evolution of the static topological features for each non-connected nodes and used the forecast scores as the final scores.

In this paper, based on the network analysis, we extract 8 features including temporal, topological, node-centric and edge-centric features. Then the rank aggregation method is introduced to leverage the different features to gain a higher performance for link prediction.

VIII. Conclusion

The providers, services, compositions and developers form a service ecosystem that is constantly evolving. Understanding the usage pattern and the evolution mechanism in the service ecosystem can help to recommend services and compositions for the developers, and promote the growth of the service ecosystem.

In this paper, we propose a three-phase network prediction based approach for the service and composition recommendation and then report the result on the real-world service ecosystem: ProgrammableWeb.

1) Network Analysis: we construct the composition-service network (CSN) and the service-service network (SSN) based on the historical information, and then introduce the network series model (NSM) including the snap CSN series, the snap SSN series and the temporal collapsed sum SSN series so that we can transform the evolution prediction into a temporal link prediction task. Finally we study the usage patterns and the evolution phenomenon based on the comprehensive network analysis.

2) Network Prediction: we extract eight features from the network series, which cover the topological, temporal, node-centric and edge-centric features for the service ecosystem. Furthermore we study their influence to the ecosystem evolution, and then introduce a rank aggregation method to holistically leverage these features for the prediction. Experiments on ProgrammableWeb data shows that the rank aggregation method achieves a superior performance in predicting the future behavior of the network, than those which do not take historical information into account.

3) Network based Recommendation: Based on the predicted service network, we present three kinds of recommendation including potential composition recommendation, top service recommendation and service chain. A prototype system is implemented for these three recommendations.

As our method only employs the evolution mechanism into account, it can filter out the obsolete services, recommend based on up-to-date usage pattern, and offer recommendation for the cold start developers. It is not user-specific and requires no user query; however it can be easily used in combination with those user- and query-specific approaches such as [12, 15]. In future work we will investigate the cold start prediction issue we left behind, so as to offer a more comprehensive method to recommend services and compositions. Also we will study the performance of our method for the other service ecosystem, such as the workflow-service ecosystem in myExperiment⁶ and Biocatalogue⁷.

⁶ www.myexperiment.org
⁷ www.biocatalogue.org
ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. 61174169) and the National Key Technology R&D Program (No. 2012BAF15G00).

REFERENCES


Research and development in information retrieval, 2001, pp. 276-284.

Wei Tan received the B.S. degree and the Ph.D. degree from Department of Automation, Tsinghua University, China in 2002 and 2008, respectively. He is currently a research staff member in IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. From 2008 to 2010 he was a researcher at Computation Institute, University of Chicago and Argonne National Laboratory, USA. At that time he was the technical lead of the caBIG workflow system. His research interests include NoSQL, big data, cloud computing, service-oriented architecture, business and scientific workflows, and Petri nets. He has published over 50 journal and conference papers, and he has authored ten books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and Web Service-Oriented Approach. Wiley-IEEE Press, 2013

Keman Huang received a B.S. degree in control theory and application as well as another B.S. degree in economics from Tsinghua University in 2002, respectively. He is currently a Ph.D student in Automation Department in Tsinghua University, Beijing, China. His research interests include services computing, web service composition, social network analysis, data mining and service recommendation.

Yushun Fan received the Ph.D. degree in control theory and application from Tsinghua University, Beijing, China, in 1990. From September 1993 to 1995, he was a Visiting Scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He is currently a Professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. He has authored ten books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and workshop management and control.