2017 IEEE 24th International Conference on Web Services

Recommendation for Newborn Services by Divide-and-Conquer

Junqi Zhang
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China
jq-zhang15@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

Jia Zhang
Carnegie Mellon University Silicon Valley
jia.zhang@sv.cmu.edu

***Abstract*—Service recommendation plays a critical role in fostering the growth of service ecosystems. However, existing methods are mainly in favor of a small number of popular services while newly emerged ones (i.e., newborn services) are largely ignored, which hurts the systems in two aspects. First, the potential of many services, especially the newborn ones, is wasted. Second, service ecosystems highly depending on a few kernel services are not diversified nor robust. To address this issue, we propose to proactively recommend collaborative services for newborn ones. The aim is to illuminate how to use the newborn services and fertilize their proper usages. While this is a cold start problem, frequent collaboration among newborn or dissimilar services makes it more difficult. In this situation, a Divide-and-Conquer approach is adopted utilizing category tags and collaboration records (DCCC). For each newborn service, the approach first produces one ranked list of old services and one list of newborn services, separately. DCCC then merges the two lists into one for recommendation. Experiments over a real-world dataset from ProgrammableWeb demonstrate that the proposed approach achieves significant improvement in recommendation accuracy compared with baseline methods.**

***Keywords*-recommendation; collaboration; newborn services; LDA; divide-and-conquer**

I. INTRODUCTION

With the wide adoption of Service-Oriented Architecture and Cloud Computing, many web service ecosystems (such as ProgrammableWeb) have emerged in recent years [1]. Mashups are created by reusing and assembling existing web services to meet complex functional requirements [2]. To facilitate locating desired services, many service recommendation methods have been developed and proven effective [3], [4], [5], [6]. In spite of such encouraging facts, however, we concern about two phenomenons, to which should be paid great attention, in service ecosystems.

Firstly, the majority of mashup-service usage records are related to a few kernel services [7]. Taking ProgrammableWeb as an example, only 10% published services are ever used in any mashup and 3.8% published services contribute to 92.6% usage records. Such undue centralization hurts service ecosystems in two ways. On one hand, the potential of many services, especially the newly emerged ones, is wasted. On the other hand, the systems are not diversified nor robust. For example, *Yahoo Search*, which was once a popular API in ProgrammableWeb, was deprecated in 2015. As a terrible result, 146 mashups (2% of the total number of mashups) containing *Yahoo Search* were forced to be deprecated as well.

Furthermore, existing recommendation methods may exacerbate the unbalance of service usage, which aggravates the waste of potential and the lack of robustness rather than eliminating them. Most methods take advantage of usage records in order to get better performance [6], [8], [9] and they tend to recommend popular services (e.g., *Google Maps*). As a result, popular services will become even more popular while long-tail services are usually despised. Especially, newly emerged services may be completely ignored, since they do not have any usage record.

From the perspective of service ecosystem operators, the potential of every service should be fully exploited, and service ecosystems should not centralize unduly. Therefore, we propose a novel idea, *to proactively recommend collaborative services for newborn services*. Services emerged in the latest month and unused since then, are tagged *newborn services* in this paper. In more detail, for each newborn service, we proactively recommend old services and other newborn ones, separately. Such a recommendation process starts once a newborn service emerges. Our core idea is to exploit the functional potential of each newborn service in time and to illuminate how to use it with other collaborative services. In this way, we aim to benefit enhancing the diversity and robustness of service ecosystems.

However, recommendation for newborn services is a difficult task and no existing methods can be directly applied due to three significant issues. Firstly, this is obviously a cold start problem. Secondly, newborn services could also collaborate with other newborn ones, which means it may become a both-side cold start problem. Thirdly,

* Corresponding Author

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.16

two collaborated services may be totally dissimilar in their functions or descriptions, which happens frequently. For example, location-related APIs (e.g., *Google Maps*) and social network APIs (e.g., *Facebook*) are often combined to realize location-aware social network mashups [10].

Most existing service recommendation methods are based on the collaborative filtering techniques, which in general completely ignore cold services [11]. As a result, these methods cannot predict the collaboration among newborn services. Some existing methods are based on content matching, which recommend according to semantic similarity [3]. They are not able to discover potential collaboration among dissimilar services with complementary functional descriptions. A few approaches have been proposed to learn interactions among services. Most of them [12], [13], [14] use the Apriori algorithm. [15] adopts a link prediction approach, [16] mines negative rules among services, and [10] mines the latent service co-occurrence topics. However, all of them only examine existing service interactions. Thus, they cannot solve our cold start problem.

To tackle the aforementioned three issues, we propose a Divide-and-Conquer approach (DCCC) as illustrated in Fig. 1, which takes advantage of both category tags and collaboration records. To address the first issue on cold start problem, category tags are utilized as a complement to text descriptions. Since category tag is a kind of large-granularity information, the category tags of a cold service may have appeared several times. To address the second issue about the collaboration among newborn services, we divide our problem into two sub-problems. For each newborn service, we recommend old services and other newborn ones separately, in two ranked lists. To address the third issue on predicting future collaboration among dissimilar services, collaboration records are utilized as a complement to mashup-service usage records. In a service ecosystem, not only do services collaborate, categories also collaborate with each other.

The main contributions of this paper are three-fold:

1) We have introduced and studied a new research problem, *recommendation for newborn services*. We fully exploit functional potential of each newborn service and illuminate how to use it with other collaborative services, aiming at enhancing the diversity and robustness of a service ecosystem. As far as we know, this is the first effort in services computing.

42) We have proposed a *Divide-and-Conquer* approach to proactively recommend collaborative services for newborn services. For better performance, we take advantage of *category tags* as well as *collaboration records* in the process.

3) Comprehensive experiments over a real-world dataset from ProgrammableWeb show that our approach yields better precision than baseline methods. We confirm that not only our divide-and-conquer strategy but also category tags and collaboration records are helpful for solving this problem.

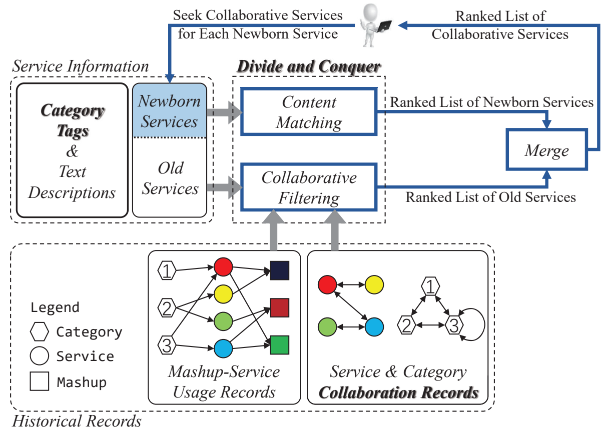

Figure 1. Framework of recommendation for newborn services by divide-and-conquer. For each newborn service, DCCC returns a ranked list of collaborative services. Separately, future collaboration among newborn services only is predicted by content matching, while collaboration among newborn and old services is predicted by collaborative filtering. Finally, those two parts are merged properly. Besides usage records, collaboration records are taken into consideration in collaborative filtering. Meanwhile, category tags are utilized to help solving this cold start problem. The details are presented in Sections II, III and IV.

The rest of this paper is organized as follows. Section II introduces fundamental definitions and formulates the problem. Model constructions and recommendation framework are described in Sections III and IV, respectively. Section V reports experimental results. Section VI compares with the related work and Section VII concludes the paper.

II. Problem Definitions

In this section, we first present several important definitions, and then formulate the recommendation problem: *recommendation for newborn services*.

Definition 1: ***Time Information***. Setting one month as a particular time granularity, a sequence of timestamps $TS = \{1, 2, \ldots, T\}$ represent the time information in a service ecosystem.

Definition 2: ***Topology***. The topology of an evolving service ecosystem is modeled as a sequence of undirected graphs. Specifically, at timestamp $t \in TS$, the service ecosystem is modeled as $G^t = (M^t \cup S^t, E^t)$. $M^t = \left\{m_1, m_2, \ldots, m_{N_m^t}\right\}$ is the set of mashups created before timestamp t, and $S^t = \left\{s_1, s_2, \ldots, s_{N_s^t}\right\}$ is the set of services emerged before timestamp t. $E^t \subseteq M^t \times S^t$ represents the mashup-service usage records, i.e., if a mashup invokes a service, an edge exists between the two nodes.

Definition 3: ***Category Tags***. Category tags of services and mashups are viewed as a kind of large-granularity information in a service ecosystem.

Many services or mashups carry tags from different categories. Assuming that there are N_{ca} different categories in a service ecosystem. At timestamp $t \in TS$, we have the corresponding G^t, in which there are N_s^t services and N_m^t mashups. The categories of services are represented by a service-category matrix $\mathbf{SCA^t} = (sca_{ij}^t)_{i=1,j=1}^{N_s^t \times N_{ca}}$. If service $s_i \in S^t$ has a tag of category j, then $sca_{ij}^t = 1$; otherwise, $sca_{ij}^t = 0$. Similarly, the categories of mashups are represented by a mashup-category matrix $\mathbf{MCA^t} = (mca_{ij}^t)_{i=1,j=1}^{N_m^t \times N_{ca}}$. $mca_{ij}^t = 1$ when mashup $m_i \in M^t$ has a tag of category j; and $mca_{ij}^t = 0$ otherwise.

Definition 4: ***Recommendation for Newborn Services***.

Given G^t, $\mathbf{SCA^t}$ and $\mathbf{MCA^t}$, at timestamp $(t+1)$, we obtain a set of N_{Snew}^{t+1} newborn services, denoted by S_{new}^{t+1}, and the corresponding category matrix $\mathbf{NSCA^{t+1}} = \left(nsca_{ij}^{t+1}\right)_{i=1,j=1}^{N_{Snew}^{t+1} \times N_{ca}}$. If a newborn service $ns_p \in S_{new}^{t+1}$ has a tag of category j, then $nsca_{pj}^{t+1} = 1$; otherwise, $nsca_{pj}^{t+1} = 0$.

For a selected newborn service $ns_s \in S_{new}^{t+1}$, a ranked list of potential collaborative services denoted by $RL\left(ns_s\right)$ will be recommended. A service with higher rank in $RL\left(ns_s\right)$ has a higher probability to collaborate with ns_s in the future.

The recommendation problem is thus turned into finding $RL\left(ns_s\right)$. There are two groups of services in candidate list $CL\left(ns_s\right)$: all the old services $s_i \in S^t$, $i = 1, 2, \ldots, N_s^t$ and all the other newborn services $ns_p \in S_{new}^{t+1}$, $p \neq s$. We propose a divide-and-conquer approach to find the $RL\left(ns_s\right)$.

As shown in Fig. 1, our approach consists of two processes: divide-and-conquer and merging. In the process of divide-and-conquer, we deal with old services and other newborn ones separately. For all newborn services except the selected one, we produce a ranked list by *description&category-based content matching* (DCaCM). For all old services, we produce another ranked list by *mashup-service-usage-records-based collaborative filtering* (MURCF) combined with *collaboration-records-based collaborative filtering* (CRCF).

In the process of merging, those two independent ranked lists are merged into a unified one, which is the $RL\left(ns_s\right)$. Then we can recommend collaborative services for the selected newborn one according to $RL\left(ns_s\right)$.

III. Model Constructions

In this section, we introduce the constructions of three main components in our approach: DCaCM, MURCF and CRCF. Potential collaboration between a selected newborn service and other newborn ones is predicted by DCaCM. MURCF and CRCF are designed to predict the future collaboration between old services and the selected newborn one.

A. Description&Category-based Content Matching

Our earlier study over ProgrammableWeb has revealed an important observation: newborn services, which have similar functions, tend to collaborate with each other [11]. In this paper, both text descriptions and category tags are taken into consideration, as small-granularity and large-granularity information respectively, for calculating the similarity between two services.

Firstly, the similarity based on text descriptions is calculated. We apply the "Latent Dirichlet Allocation" (LDA) [17] model to obtain the topic distribution of every service and mashup, and then calculate the similarity between two services according to their distributions over topics.

Each service s comprises a collection of words $SW\left(s\right) = \left\{sw_1, sw_2, \ldots, sw_{N_W(s)}\right\}$ to describe its functional abilities. Similarly, each mashup m is associated with a collection of words $MW\left(m\right) = \left\{mw_1, mw_2, \ldots, mw_{N_W(m)}\right\}$ to describe its functions. We input all services and mashups with their associated sets of words $SW\left(s\right)$ and $MW\left(m\right)$ into an LDA model. Although we can obtain the distribution over topics of all services and mashups, in DCaCM, we are only interested in the distribution over topics of newborn services.

Let $z \in [1, K]$ be the topic indicator variable. At time $(t+1)$, the distribution over K composition topics of N_{Snew}^{t+1} newborn services can be represented by a $N_{Snew}^{t+1} \times K$ matrix Φ. In Φ, each row ϕ_p is a K-dimensional multinomial distribution of a newborn service ns_p with $\phi_{p,z} = P\left(z \middle| ns_p\right)$ and $\sum_{z=1}^{K} \phi_{p,z} = 1$. The similarity between text descriptions of a selected newborn service $ns_s \in S_{new}^{t+1}$ and another newborn service $ns_p \in S_{new}^{t+1}$, $p \neq s$ can be calculated by the following equation:

$$sim_d\left(ns_s, ns_p\right) = \frac{\phi_s \cdot {\phi_p}^{\mathrm{T}}}{\left\|\phi_s\right\| \left\|\phi_p\right\|} \tag{1}$$

Secondly, the similarity based on category tags is calculated. As stated in Section II, in $\mathbf{NSCA^{t+1}}$, each row $nsca_p^{t+1}$ represents the category vector of a newborn service ns_p. Thus, the similarity between categories of a selected newborn service and another newborn one can be calculated by the following equation:

$$sim_{ca}(ns_s, ns_p) = \frac{nsca_s^{t+1} \cdot {nsca_p^{t+1}}^{\mathrm{T}}}{\left\|nsca_s^{t+1}\right\| \left\|nsca_p^{t+1}\right\|} \tag{2}$$

Finally, DCaCM calculates the similarity between a selected newborn service and another newborn one as follows:

$$\begin{aligned} sim_{cm}(ns_s, ns_p) = (1-\lambda_{cm}) & sim_d(ns_s, ns_p) \\ + \lambda_{cm} & sim_{ca}(ns_s, ns_p) \end{aligned} \tag{3}$$

where $\lambda_{cm}$is a parameter to trade off text descriptions and category tags in DCaCM.

B. Mashup-Service-Usage-Records-based Collaborative Filtering

Collaborative filtering is one of state-of-the-art methods in the recommendation community [8]. We expand it to help recommending potential collaborative services for a newborn service. If an existing mashup $m \in M^t$ is partly similar with a newborn service $ns_s \in S_{new}^{t+1}$, the services invoked by m will tend to collaborate with ns_s in the future.

Firstly, we calculate the similarity between a newborn service ns_s and an existing mashup m according to their text descriptions and category tags. Based on DCaCM, We calculate their similarity as follows:

$$\begin{aligned} sim(ns_s, m) = (1 - \lambda_{mcf}) sim_d(ns_s, m) \\ + \lambda_{mcf} \cdot sim_{ca}(ns_s, m) \end{aligned} \tag{4}$$

where λ_{mcf} is also a parameter to trade off text descriptions and category tags, but in MURCF. For convenience, we define $msim_m(ns_s)$ as the highest similarity between ns_s and every mashup m_i:

$$msim_m(ns_s) = \max_{m_i \in M^t} sim(ns_s, m_i) \tag{5}$$

Afterwards, $M_{\eta_{mcf}}^t$ is defined as follows:

$$M_{\eta_{mcf}}^t = \left\{ m \middle| \begin{array}{c} m \in M^t, \\ sim(ns_s, m) \geq \eta_{mcf} msim_m(ns_s) \end{array} \right\} \tag{6}$$

At timestamp $(t+1)$, the probability of future collaboration between a selected newborn service ns_s and an old service $s_i \in S^t$ can be calculated as follows:

$$p_{mcf}(s_i | ns_s) = \sum_{m_j \in M_{\eta_{mcf}}^t} sim(ns_s, m_j) y(m_j, s_i) \tag{7}$$

in which $y(m_j, s_i) = 1$, if $(m_j, s_i) \in E_t$, and $y(m_j, s_i) = 0$ otherwise. $p_{mcf}(s_i|ns_s)$ is one part of the probability for collaboration between ns_s and s_i.

C. ***Collaboration-Records***-based Collaborative Filtering

CRCF, as a complement to MURCF, utilizes collaboration records among services as well as collaboration records among categories. CRCF is designed to facilitate predicting the collaboration among dissimilar services.

On one hand, similar services tend to collaborate with same services. On the other hand, if there are two services and their categories collaborate frequently according to collaboration records, we believe they tend to collaborate with the same services.

Firstly, given a timestamp $t \in TS$ and the corresponding G^t, we can derive a **service-collaboration** matrix $\mathbf{SC^t} = (sc_{ij}^t)_{i=1,j=1}^{N_s^t \times N_s^t}$. For each service $s_i \in S^t$, if it collaborates with another service $s_j \in S^t$ for k times by the end of time t, then $sc_{ij}^t = k$. Collaboration with one service itself is not counted. Thus, $sc_{ii}^t = 0$, $i = 1, 2, \ldots, N_s^t$. Using $\mathbf{SC^t}$ and $\mathbf{SCA^t}$ (defined in Section II), we can derive a **category-collaboration** matrix $\mathbf{CC^t} = (cc_{ij}^t)_{i=1,j=1}^{N_{ca} \times N_{ca}}$. $cc_{ij}^t = k$ when category i and j collaborate for k times by the end of time t. Different from $\mathbf{SC^t}$, collaboration with one category itself is counted in $\mathbf{CC^t}$. Then $\mathbf{SC^t}$ and $\mathbf{CC^t}$ are normalized into $\mathbf{NSC^t}$ and $\mathbf{NCC^t}$ as follows:

$$nsc_{uv}^t = \frac{sc_{uv}^t}{\sum_w sc_{uw}^t}, ncc_{uv}^t = \frac{cc_{uv}^t}{\sum_w cc_{uw}^t} \tag{8}$$

Secondly, a generalized similarity between a selected newborn service $ns_s \in S_{new}^{t+1}$ and an old one $s_i \in S^t$ is calculated. Part of this similarity is called category-collaboration similarity. In $\mathbf{SCA^t}$, each row sca_i^t represents the category vector of s_i. In $\mathbf{NSCA^{t+1}}$ (defined in Section II), $nsca_s^{t+1}$ represents the category vector of ns_s. The category-collaboration similarity between ns_s and s_i can be calculated as follows:

$$cacosim(ns_s, s_i) = \sum_{u=1}^{N_{ca}} \sum_{v=1}^{N_{ca}} (nsca_{su}^{t+1} \cdot ncc_{uv}^t \cdot sca_{iv}^t) \tag{9}$$

Then the generalized similarity can be calculated as follows:

$$\begin{aligned} sim_g(ns_s, s_i) = (1 - \lambda_{crcf}) sim_d(ns_s, s_i) \\ + \lambda_{crcf} \cdot cacosim(ns_s, s_i) \end{aligned} \tag{10}$$

where λ_{crcf} is another parameter to trade off text descriptions and category tags. Similar with what we have done in MURCF, for convenience, $msim_s(ns_s)$ are defined as the highest generalized similarity between ns_s and every old service s_i. Afterwards, $S_{\eta_{crcf}}^t$ is defined as follows:

$$S_{\eta_{crcf}}^t = \left\{ s \middle| \begin{array}{c} s \in S^t, \\ sim_g(ns_s, s) \geq \eta_{crcf} msim_s(ns_s) \end{array} \right\} \tag{11}$$

Finally, the other part of probability for future collaboration between ns_s and an old service s_i can be calculated as follows:

$$p_{crcf}(s_i | ns_s) = \sum_{s_j \in S_{\eta_{crcf}}^t} sim_g(ns_s, s_j) nsc_{ji}^t \tag{12}$$

IV. Recommendation Framework

Based on previously introduced components, DCaCM, MURCF and CRCF, in this section, we show how to integrate them for recommendation.

A. The Process of Divide-and-Conquer

At timestamp $(t+1)$, a newborn service $ns_s \in S_{new}^{t+1}$ is selected. Other newborn services $ns_p \in S_{new}^{t+1}$, $p \neq s$ and all the old services $s_i \in S^t$ are in candidate list $CL(ns_s)$. Separately, each candidate newborn service ns_p receives a point calculated by DCaCM:

$$p_{ns}(ns_p | ns_s) = sim_{cm}(ns_s, ns_p) \tag{13}$$

while each candidate old service s_i receives a point calculated by MURCF and CRCF as follows:

$$\begin{aligned} p_s(s_i | ns_s) = (1 - \mu) p_{mcf}(s_i | ns_s) \\ + \mu \cdot p_{crcf}(s_i | ns_s) \end{aligned} \tag{14}$$

in which μ is a parameter to trade off MURCF and CDCF.

So far, every candidate service in $CL(ns_s)$ gets a point but in two standards, and two independent ranked lists can be produced according to these points. We will unify the points in the next process and merge those two lists into a unified and ranked one for recommendation.

B. The Process of Merging

One important observation is that collaboration between two newborn services is less than that between a newborn service and an old one. So we select Top N_{new} candidate newborn services according to p_{ns}, and adjust their points by a factor σ. Then the unified point for each candidate service $s_c \in CL(ns_s)$ is determined by the following equation:

$$p(s_c|ns_s) = \begin{cases} p_s(s_c|ns_s), & \text{if } s_c \in S^t \\ \sigma \cdot p_{ns}(s_c|ns_s), & \text{if } s_c \text{ is Top } N_{new} \text{ in } S_{new}^{t+1} \\ 0, & \text{others} \end{cases} \tag{15}$$

Finally, the ranked recommendation list $RL(ns_s)$ can be produced according to $p(s_c|ns_s)$ in a descending order.

V. Experiments

In this section, we present our empirical experiments on a real-world dataset from ProgrammableWeb, evaluating the performance of our proposed approach (DCCC) against the state-of-the-art methods. The effects of our divide-and-conquer strategy, category tags and collaboration records are demonstrated.

A. Dataset

ProgrammableWeb has been accumulating a variety of services and mashups since its establishment in 2005 [18]. We crawled the metadata of all services and mashups from September 2005 to August 2016. Metadata includes name, creation date, description, category and mashup-service usage records. Numerical properties of the dataset are summarized in Table I.

Table I
Numerical Properties of Dataset

Number of services	15,386
Number of mashups	7,822
Number of services collaborating with other services	1,289
Number of mashups containing more than one service	3,564
Number of categories	437
Size of vocabulary	24,209

B. ***Experiment Preparation***

1) Preprocess: Mashups containing less than two services and categories appearing less than twice were removed, because they cannot offer any collaboration records. We applied word stemming and removed stop words in text descriptions.

As stated in section III, we applied LDA to obtain distributions over K latent topics for every service and mashup. α and β are two hyper-parameters in LDA [17] . In our experiments, we set $K = 60$, $\alpha = 50/K$, and $\beta = 0.01$.

2) ***Training and Test Sets****:* In our experiments, we adopted a time granularity of one month, and there are 144 months from September 2005 to August 2016. To test the performance of our approach (DCCC), we divided the dataset into training and test sets by a moving timestamp $t \in TS$. Given a cutoff timestamp t, we regard the data before it $[1, t]$ as a training set, and the data in the following ten months $[t + 1, t + 10]$ as a test set. We moved the timestamp from August 2007 to October 2015, $t \in [24, 134]$, and obtained 111 training and test sets. We did experiments on overall 111 training and test sets. In other words, we tested our approach and baseline methods in more than a **nine-year** period **month by month**. For each newborn service at timestamp $(t + 1)$, we will recommend proper services to collaborate with it. The collaboration records in the following ten months $\{\mathbf{SC^{t+1}}, \mathbf{SC^{t+2}}, \ldots, \mathbf{SC^{t+10}}\}$ act as ground truth.

C. Evaluation Metric

Similar with [4], two widely accepted metrics, Mean Average Precision @ top N (MAP@N) and Normalized Discounted Cumulative Gain @ top N (NDCG@N), are used in our experiments.

Both MAP@N and NDCG@N are real numbers between 0 and 1. The higher MAP@N or NGCD@N indicates a better accuracy of the recommendation. Different from MAP@N, NDCG@N emphasizes on the precision of the first few (1^{st}, 2^{nd},3^{rd},...) recommendations.

By moving the cutoff timestamp t , we can calculate MAP@N and NGCD@N for each test set and use the average value of MAP@N and NDCG@N as the evaluation metric to compare DCCC with baseline methods.

D. Baseline Methods

Some complex methods for service recommendation [10], [15], [19] were proposed in recent years, however, none of them can be directly applied to our problem. Therefore, three typical approaches were selected as baselines. Another three baselines were generated by reducing components in DCCC, to test our approach more meticulously.

1) ***Baseline Method 1****:* **A Probabilistic Approach (PA)**

The Probabilistic Approach [5] applies LDA to calculate the semantic similarity between the selected service $ns_s \in S_{new}^{t+1}$ and any other candidate service $s_c \in CL(ns_s)$. Gibbs sampling is applied to get probability distribution of services over topics $p(z|s)$ and topics over words $p(w|z)$. The description of ns_s is $SW(ns_s)$ and the similarity is calculated according to:

$$p_{pa}(s_c|ns_s) = \sum_{w \in SW(ns_s)} \sum_{z=1}^{K} p(w|z)p(z|s_c) \tag{16}$$

The recommendation list $RL(ns_s)$ is then ranked in a descending order w.r.t $p_{pa}(s_c|ns_s)$.

2) Baseline Method 2: **TopPopN**

TopPopN approach [9] recommends the top N popular services for each selected newborn one.

3) Baseline Method 3: **Mashup-Description-based Collaborative Filtering (MDCF)**

Collaborative Filtering is widely acknowledged as the most important recommendation algorithm and applied in many methods [6], [8]. The semantic similarity of an mashup $m \in M^t$ and a selected newborn service $ns_s \in S_{new}^{t+1}$ is calculated according to $sim_d(ns_s, m)$ in (1). The recommendation list $RL(ns_s)$ is ranked in a descending order w.r.t $p_{md}(s_c|ns_s)$, which can be calculated as follows:

$$p_{md}(s_c|ns_s) = \sum_{m_j \in M^t_{\eta_{mcf}}} sim_d(ns_s, m_j) y(m_j, s_c) \quad (17)$$

where $M^t_{\eta_{mcf}}$ is defined the same as (6).

4) Baseline Method 4: **A Divide-and-Conquer Approach without category tags and collaboration records (DC)**

DC ignores category tags and collaboration records in a service ecosystem. In other words, DC is degenerated from DCCC by setting λ_{cm}, λ_{mcf} and μ all to 0, and $RL(ns_s)$ is ranked in a descending order w.r.t $p(s_c|ns_s)$ in (15). Comparing DCCC with DC, we can demonstrate the effect of category tags together with collaboration records.

5) Baseline Method 5: **DCaCM**

Similar with PA, DCaCM recommends potential collaborative services according to the similarity between the selected newborn service ns_s and any other one s_c. The difference is that DCaCM takes category tags into consideration and calculates the similarity according to (3). $RL(ns_s)$ is then ranked in a descending order w.r.t $sim_{cm}(ns_s, s_c)$ in (3).

6) Baseline Method 6: **A Complex Collaborative Filtering combining MURCF with CRCF (CCF)**

This baseline method only comprises two components in our complete approach and it recommends potential collaborative services only through collaborative filtering. In other words, CCF is a degenerated DCCC in which $\sigma = 0$ and it ignores the collaboration between newborn services. Recommendation list $RL(ns_s)$ is ranked in a descending order w.r.t $p_s(s_c|ns_s)$ in (14).

E. Experiments Results

1) Parameters Settings: We tuned the parameters of our approach and every baseline method, respectively, to get the best performance of each method. In DCCC, $\lambda_{cm} = 0.5$, $\lambda_{mcf} = 0.4$, $\eta_{mcf} = 0.4$, $\lambda_{crcf} = 0.35$, $\eta_{crcf} = 0.45$, $\mu = 0.4$, $\sigma = 2.1$, $N_{new} = 9$. In MDCF, $\eta_{mcf} = 0.4$. In DC, $\eta_{mcf} = 0.4$, $\sigma = 1.7$, $N_{new} = 9$. In DCaCM, $\lambda_{cm} = 0.35$. In CCF, $\lambda_{mcf} = 0.4$, $\eta_{mcf} = 0.4$, $\lambda_{crcf} = 0.35$, $\eta_{crcf} = 0.45$, $\mu = 0.4$.

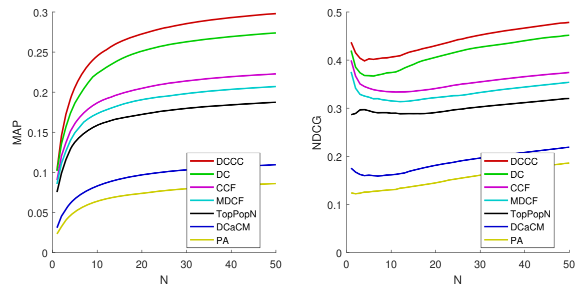

Figure 2. MAP@N and NDCG@N of all approaches.

2) Performance Comparison: Fig. 2 illustrates the MAP@N and NDCG@N of our approach and all baseline methods. PA only considers the semantic similarity between services. However, collaborative services may have little semantic similarity or even be totally dissimilar. As a result, PA gets the lowest MAP and NDCG. TopPopN recommends services simply according to their popularity. Our experiments confirm that it is effective in some degree. MDCF, as a typical method based on collaborative filtering, utilizes mashup-service usage records and performs better than other two published methods. DCaCM takes advantage of category tags as large-granularity information, and it performs better than PA. Similarly, CCF utilizes not only category tags but also collaboration records as a complement to mashup-service usage records, and it performs better than MDCF. By adopting the divide-and-conquer strategy, DCCC and DC achieve much better performance than CCF and MDCF.

The detailed performance of different approaches is summarized in Table II. In terms of MAP@N, DCCC has an approximately 9% better performance than the best published method MDCF. Comparison between DC and MDCF demonstrates a more than 6.5% promotion by adopting the divide-and-conquer strategy. It can be inferred by comparing DCCC with DC, category tags together with collaboration records contribute a 2.5% promotion furthermore.

NDCG@N emphasizes on the precision of the first few (1^{st}, 2^{nd},3^{rd},...) recommendations. Both Fig. 2 and Table II illustrate that DCCC performs pretty good on the

Table II
PERFORMANCE COMPARISON BETWEEN ALL APPROACHES ON MAP@20, MAP@50, NDCG@5 AND NDCG@50

Approaches	MAP@20	MAP@50	NDCG@5	NDCG@50
DCCC	**27.27%**	**29.80%**	**0.4026**	**0.4785**
DC	25.13%	27.39%	0.3677	0.4517
CCF	20.48%	22.28%	0.3415	0.3744
MDCF	19.06%	20.71%	0.3229	0.3539
TopPopN	17.23%	18.74%	0.2947	0.3202
DCaCM	9.70%	10.95%	0.1608	0.2189
PA	7.40%	8.60%	0.1258	0.1858

first few recommendations. DCCC gets a higher NDCG@5 than MDCF by 0.0796 (24.65% relatively), when category tags together with collaboration records contribute 0.0349 (10.81% relatively) and the divide-and-conquer strategy contributes the other 0.0447 (13.85% relatively).

As a conclusion, it improves the accuracy of recommendation significantly by adopting the divide-and-conquer strategy alone. Moreover, it leads to more improvement to take advantage of category tags together with collaboration records, especially when the length of a recommendation list is shorter than 10.

3) ***Effect of Category Tags****:* To demonstrate the effect of category tags more sufficiently, we analyzed the effects of λ_{cm}, λ_{mcf} and λ_{crcf}. Those parameters balance the influence of text descriptions and category tags (small-granularity and large-granularity information) in three main components of DCCC. λ_{cm}, λ_{mcf} and λ_{crcf} are all real numbers between 0 and 1, and when they are 0, category tags are ignored in DCCC. Since their values are almost the same in DCCC, for convenience, we set them equal $\lambda_{cm} = \lambda_{mcf} = \lambda_{crcf} = \lambda$ in this analysis.

Fig. 3 illustrates the MAP@20 and NDCG@5 of DCCC with different λ. Both MAP@20 and NDCG@5 reach the maximum when $\lambda \approx 0.5$, which indicates that taking category tags into consideration improves the quality of recommendation in our cold start problem.

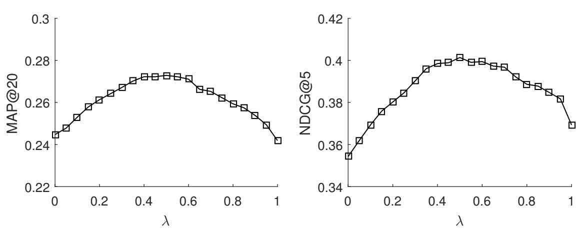

Figure 3. MAP@20 and NDCG@5 of DCCC with different λ

4) ***Effect of Collaboration Records****:* To further demonstrate the limitation of mashup-service usage records and the contribution of collaboration records on predicting the collaboration among dissimilar services, next a qualitative analysis of our experimental results is presented.

Top three services in the ranked recommendation list for several typical services are presented in Table III. At the times they emerged in the system, we recommend potential collaborative services for each of them twice. For example, *Facebook* is a social network API, emerged in August 2006. Without collaboration records, the top 3 services are all related to social network. *Flickr* is a photo sharing API, *Plazes* is a location-aware social network API, and *del.ico.us* is a social bookmarking API. However, only *Flickr* collaborated with *Facebook* in reality. With collaboration records added, a location-related API *Google Maps* appears in the top 3 list. Most importantly, *Google Maps* collaborated with *Facebook* in reality, which is a typical collaboration between dissimilar services. The situations are similar for other cases in Table III, potential collaboration among dissimilar services, such as *The Movie DB*, *Baidu* and *Twitter*, can be discovered by taking collaboration records into consideration.

Table III demonstrates that collaboration records are effective complement to mashup-service usage records. Meanwhile, it proves that collaboration records help to predict future collaboration among dissimilar services.

Table III
TOP 3 SERVICES IN RANKED RECOMMENDATION LIST FOR SEVERAL TYPICAL NEWBORN SERVICES

	Without Collaboration Records	**With Collaboration Records Added**
Facebook	Flickr, ~~Plazes~~, ~~del.icio.us~~	Flickr, ~~Plazes~~, ***Google Maps***
Twitter	Flickr, ~~Amazon Product Advertising~~, ~~Technorati~~	Flickr, ***Google Maps***, ~~Amazon Product Advertising~~
Map24 AJAX	Google Maps, ~~Plazes~~, ~~Yahoo Geocoding~~	Google Maps, ~~Plazes~~, ***Flickr***
The Movie DB	YouTube, Amazon Product Advertising, ~~Google Maps~~	YouTube, ***Twitter***, Amazon Product Advertising
Baidu	Google Maps, ~~YouTube~~, ~~Yahoo Search~~	Google Maps, ***Twitter***, ~~YouTube~~

[1] ***Italics*** indicate that service can only be recommended by utilizing collaboration records.
[2] A ~~strikeout~~ indicates that service never collaborates with the selected one in reality.

VI. RELATED WORK

A. Service Recommendation

With the explosive development of service ecosystems, service recommendation becomes a key problem. Early works used to employ keyword-based methods on the information from *Web Service Description Language (WSDL)* [3]. However, keyword-based methods suffer from poor performance in practice. In this situation, the LDA model is widely used in later work to characterize the latent topics between service descriptions and users' queries [5]. The LDA model is also used to extract temporal information in an evolving service ecosystem [11]. On the other hand, a lot of research work [6], [8] are supported by neighborhood-based collaborative filtering. [19] and [20] combine collaborative filtering and content matching for better performance. Unfortunately, all these approaches recommend services based on users' queries, which are commonly represented by the description of mashups. Most of them tend to recommend popular services [9].

B. Service Collaboration

In the research of service collaboration, most works are on the basis of Apriori algorithm [12], [13], [14]. Apriori algorithm can be used to mine association rules among services. [15] proposed a link prediction method to predict future collaboration between services. [16] came up with a new method to mine latent negative association rules among services. The SeCo-LDA approach [10] used the LDA model to mine the latent co-occurrence topics among services. However, all the approaches mentioned above can only be implemented on the set of old services, which have been used before. In other words, these approaches cannot solve a cold start problem.

VII. Conclusions and Future Work

In this paper, we have introduced and studied a novel recommendation problem, *proactively recommending potential collaborative services for a newborn service.* Our motivation is to fully exploit functional potential of every single service, and to illuminate how to use newborn services with the collaborative ones. We aim to enhance the diversity and robustness of a service ecosystem.

To solve this problem, we present a divide-and-conquer (DCCC) approach, which utilizes category tags and collaboration records as complements of text descriptions and mashup-service usage records. Through three main components DCaCM, MURCF and CRCF, DCCC produces one ranked list of old services and another list of newborn services separately for each newborn service. Finally, the two ranked lists are merged into one for recommendation.

Empirical experiments demonstrate that DCCC achieves significant improvement in recommendation accuracy. The effects of our divide-and-conquer strategy, category tags and collaboration records are also confirmed.

In the future, we plan to replace the LDA-based and collaborative-filtering-based components with other methods, such as deep learning. We also plan to further study how to predict future collaboration between newborn services.

Acknowledgment

This research is supported by the National Science Foundation of China (No. 61673230).

References

[1] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the evolution of services," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 609–628, 2012.

[2] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in *Services, 2007 IEEE Congress on*. IEEE, 2007, pp. 332–339.

[3] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 372–383.

[4] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware api clustering and distributed recommendation for automatic mashup creation," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 674–687, 2015.

[5] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A probabilistic approach for web service discovery," in *Services Computing (SCC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 49–56.

[6] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 2009, pp. 437–444.

[7] K. Huang, Y. Fan, and W. Tan, "An empirical study of programmable web: a network analysis on a service-mashup system," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 552–559.

[8] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 9–16.

[9] P. Cremonesi, M. Picozzi, and M. Matera, "A comparison of recommender systems for mashup composition," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*. IEEE Press, 2012, pp. 54–58.

[10] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai, and S. Chen, "Seco-lda: Mining service co-occurrence topics for recommendation," in *Web Services (ICWS), 2016 IEEE International Conference on*. IEEE, 2016, pp. 25–32.

[11] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 356–368, 2015.

[12] Q. A. Liang, J.-Y. Chung, S. Miller, and Y. Ouyang, "Service pattern discovery of web service mining in web service registry-repository," in *2006 IEEE International Conference on e-Business Engineering (ICEBE'06)*. IEEE, 2006, pp. 286–293.

[13] K. Goarany, G. Kulczycki, and M. B. Blake, "Mining social tags to predict mashup patterns," in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*. ACM, 2010, pp. 71–78.

[14] S. Bayati, A. F. Nejad, S. Kharazmi, and A. Bahreininejad, "Using association rule mining to improve semantic web services composition performance," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*. IEEE, 2009, pp. 1–5.

[15] K. Huang, Y. Fan, and W. Tan, "Recommendation in an evolving service ecosystem based on network prediction," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 906–920, 2014.

[16] Y. Ni, Y. Fan, W. Tan, K. Huang, and J. Bi, "Ncsr: Negative-connection-aware service recommendation for large sparse service network," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 579–590, 2016.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[18] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IT Professional Magazine*, vol. 8, no. 5, p. 31, 2006.

[19] B. Bai, Y. Fan, K. Huang, W. Tan, B. Xia, and S. Chen, "Service recommendation for mashup creation based on time-aware collaborative domain regression," in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 209–216.

[20] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 42–49.