计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2018.20160521
55(3):613-628,2018

基于组合历史的交互式服务推荐方法

潘伟丰[1] 姜 波[1] 李 兵[2] 胡 博[3] 宋贝贝[1]
[1](浙江工商大学计算机与信息工程学院 杭州 310018)
[2](武汉大学计算机学院 武汉 430072)
[3](金蝶国际软件集团金蝶研究院 广东深圳 518057)
(wfpan1982@163.com)

Interactive Service Recommendation Based on Composition History

Pan Weifeng[1], Jiang Bo[1], Li Bing[2], Hu Bo[3], and Song Beibei[1]
[1](*School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou* 310018)
[2](*School of Computer Science, Wuhan University, Wuhan* 430072)
[3](*Kingdee Research, Kingdee International Software Group Co. Ltd, Shenzhen, Guangdong* 518057)

Abstract With the rapid increasing number of services and their types, how to discover the composible services which can meet uer's requirements is one of the key issues that need to be resolved. Service recommendation technique has become one of the effective methods to deal with the problem of service resource overload. However, the existing service recommendation techniques usually ultilize service data which are hard to be collected and they also neglect the usability and composiblity of the services to be recommended. To avoid these limitations, this paper, utilizing service composition histories, introduces the theory and methodology in the complex network research and proposes an interactive service recommendation approach. It uses an affiliation network to abstract service composition histories (i.e., composite services, atomic services, and the affiliation relationships between them), obtains the service composition relationships by one-mode projection, and introduces the backbone network extraction technology to filter out the invalid compostion relationships; it uses degree and degree distribution to mine the service usage patterns; it takes into account the situation of the failure of services and finally proposes several algorithms for service recommendation according to three usage scenarios. Real data of services crawed from ProgrammableWeb are used as subjects to demonstrate the correctness and feasibility of the proposed approach.

Key words service recommendation; service network; backbone network extraction; k-core decomposition; complex network

摘 要 随着服务种类和数量的飞速增长,如何发现满足用户需求的服务成为亟待解决的关键问题之一.服务推荐技术被认为是解决服务资源过载问题的有效方法之一.但是,现有的服务推荐方法存在数据难以获取和未考虑所推荐服务的可用性及与已有服务的可组合性等问题.有鉴于此,提出了一种基于

收稿日期:2016-07-15;**修回日期**:2017-05-03
基金项目:国家自然科学基金项目(61202048,61273216,61402406);浙江省自然科学基金项目(LY15F020004)
This work was supported by the National Natural Science Foundation of China (61202048, 61273216, 61402406) and the Natural Science Foundation of Zhejiang Province of China (LY15F020004).

服务组合历史的交互式服务推荐方法.该方法使用隶属网抽象服务组合历史(复合服务、原子服务及他们之间的隶属关系),通过单模投影获取服务间的组合关系,并利用骨干网挖掘过滤无效的服务组合关系;使用度和度分布分析服务的使用模式;考虑服务的失效问题,并根据服务的不同使用场景提出了相应的服务推荐算法.最后,使用 ProgrammableWeb 网站提供的真实服务数据验证了所提方法的正确性和有效性.

关键词 服务推荐;服务网络;骨干网挖掘;k-核分解;复杂网络

中图法分类号 TP311

面向服务的计算(service-oriented computing, SOC)[1-3]已成为工业界和学术界关注的热点.它倡导以服务及其组合为基础构造应用的开发模式,导致软件的形态、生产方式、运行方式和使用方式都发生了巨大变化[3].软件正处在一个由服务构成的开放、协同的环境中[4].随着由服务构成的应用的普及,服务的种类和数量(截止 2013 年 1 月 PWeb (ProgrammableWeb)① 上发布的开放 API 超过 6 400 个)急剧增加,以服务为中心的互联网正在形成[5].面对数量庞大的服务群,如何发现满足用户需求的服务已成为影响服务计算进一步发展的瓶颈[6].

推荐技术被认为是解决信息资源过载问题的有效方法之一[7].近年来,国内外学者将推荐技术引入服务计算领域,力图提高服务发现和选择的性能,主要包括[8]:基于语义的服务推荐方法[9]、基于情境的服务推荐方法[10]、基于语法的服务推荐方法[11]和基于协同过滤的服务推荐方法[12-21]等.现有的工作取得了一定的成果,但是仍有 2 点不足之处:

1) 数据难以获取.现有的方法基本都依赖事先收集的用户注册信息(兴趣、国籍、年龄等)、历史的服务调用信息、用户偏好信息、服务的 QoS 信息等数据,然而这些数据对于普通用户收集不易.

2) 未考虑所推荐服务的可用性及与已有服务的可组合性.现有的方法基本都是针对用户需求不明确时的服务推荐,针对复合服务开发(服务组合)时的服务推荐较少,基本都忽略了服务的可用性(推荐的服务是否可以访问)及推荐的服务与已有服务的可组合问题.

因此,如何在复合服务开发阶段(服务组合)时,在缺少用户注册信息、历史的服务调用信息、用户偏好信息、服务 QoS 信息等的情况下,为开发者推荐可用的、可组合的服务成为服务推荐领域的一个新问题.

针对这一新问题,我们提出了一种基于组合历史的交互式服务推荐方法(interactive service recommendation based on composition history, iSee).1)该方法依据服务注册库中(复合)服务元信息构建服务网络模型;2)使用复杂网络中的方法(如单模投影[22]、骨干网挖掘等[23])挖掘服务组合关系及使用模式;3)基于服务网络,并在服务使用模式及服务失效数据的支持下,针对服务的 3 种使用场景,提出了相应的服务推荐算法.我们以 PWeb 上 mashup 应用和开放 API 的真实数据为例进行验证,并与相关方法进行对比研究.结果表明:本文的方法在实现服务推荐方面是正确和有效的.

1 问题定义

在给出 iSee 方法的详细过程之前,首先通过一个实际场景说明本文拟解决的主要问题,并引出问题的定义.

Unmeshm 是一家社交网络公司的程序员,他曾通过组合 Facebook, Flickr, Google Maps 这 3 个开放 API 开发了一个 Blog on a Map 应用,使用户可以在地图上发布博客.后来,Unmeshm 将该应用发布在网络上,并提供了该应用的元信息(如名称、描述、标签、使用的开放 API 等).Mrowe 是另一家公司的程序员,正在利用开放 API 开发新的应用,为用户定位其 Facebook 上好友的位置.Mrowe 通过搜索已找到了一个开放 API Facebook,那么接下来他应该选择哪个 API 与 Facebook 组合呢? 网络上的开放 API 数量庞大,通过人工查看 API 文档寻找可以与 Facebook 组合的 API 是极其困难的.但是,Mrowe 预开发的应用与 Unmeshm 已开发的 Blog on a Map 应用在功能上有类似之处.Mrowe 可否利用 Unmeshm 的成功经验来寻找下一个可以

① http://www.programmableweb.com

与 Facebook 可组合的 API 呢?

Mrowe 可以利用的信息主要包括开放 API 的元信息(如名称、描述、标签等),应用(复合服务)的元信息(如名称、描述、标签、使用的开放 API 等).他无法得到用户的注册信息、偏好信息、服务的 QoS 信息等数据.因此,现有的很多服务推荐方法[8-18]都将无能为力.

因此,我们将本文拟解决的问题定义为:在复合服务开发阶段,仅给定复合服务及原子服务的一些元信息,如何利用历史的服务组合信息为开发者推荐满足需求的可用、可组合的服务.

为了解决这一问题,有 2 个关键点:

1) 如何抽象和表达蕴含在复合服务和原子服务元信息中的知识,并通过分析服务组合历史信息,挖掘服务的使用模式.

2) 如何针对复合服务开发的不同场景为用户推荐服务.

2 iSee 方法

服务的组合历史包含了服务使用的隐含知识[3].iSee 方法使用这种经过检验的知识为用户推荐服务,从而提高应用开发的效率.图 1 是 iSee 方法的框架图.

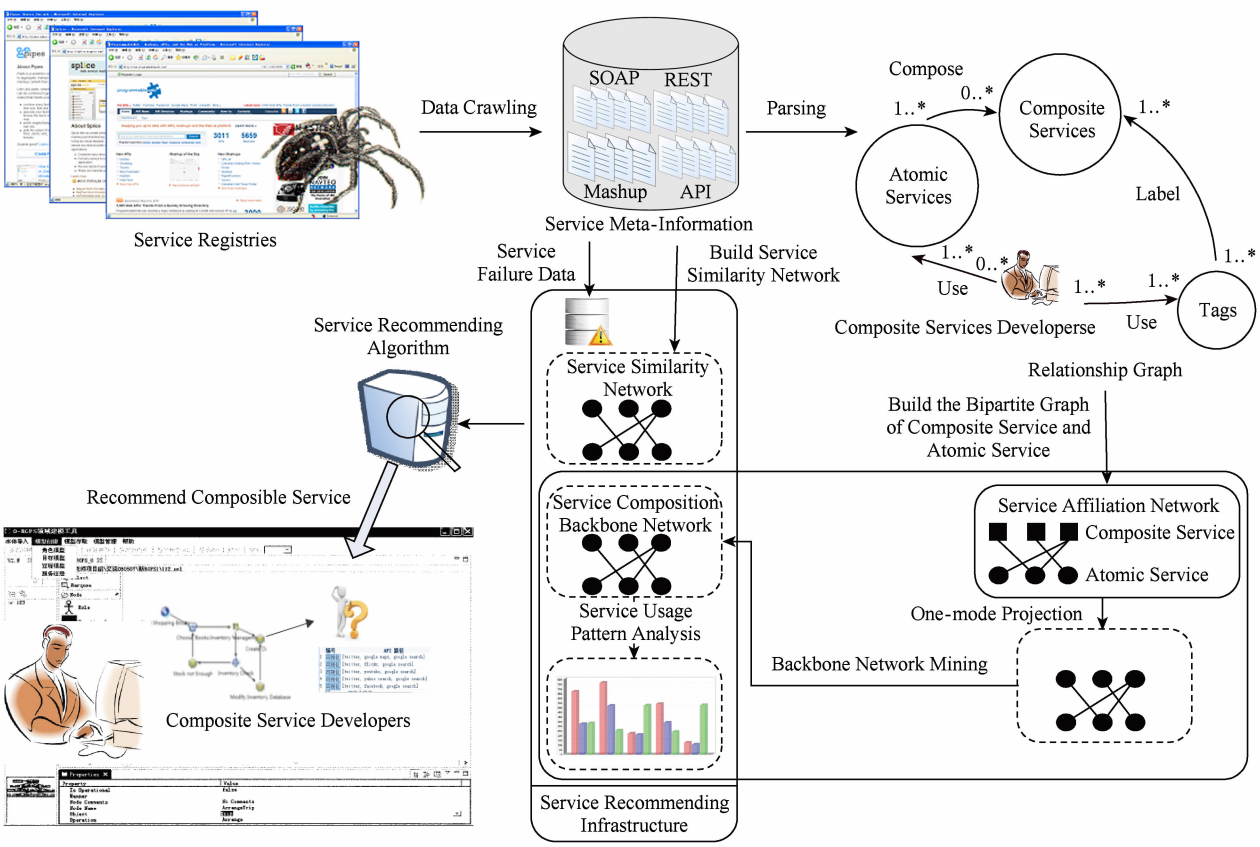


Fig. 1 Framework of iSee

图 1 iSee 方法框架

2.1 数据收集

互联网上分布着大量的服务,为了有效利用这些服务,必须使它们能够动态地找到彼此.服务注册中心为服务提供了一个宣传自己并了解其他服务的地方.为了使自己开发的(复合)服务尽可能地被找到,服务提供者一般会按特定的注册模型详细地描述其提供的服务,如原子服务的名称、描述、标签等(复合服务可能还包含其使用的原子服务集).iSee 方法主要使用这些易于获取的数据来为开发者推荐服务.因此,我们将爬取(复合)服务的这些元信息.同时,为了保证数据的准确性,我们还将对数据做降噪处理,如修正拼错的单词、去除重复的(复合)服务等[3].

2.2 服务网络模型

抽象和表达蕴含在复合服务和原子服务元信息中的知识是利用这些知识的前提[3].复合服务通常由>0 个原子服务辅以一定的编程组合而成.iSee 方法将复合服务和原子服务间的宏观组合关系用服

务网络模型抽象.下文首先给出相关服务网络的定义.

定义 1. 服务隶属网(service affiliation network, *SAN*).复合服务可由原子服务组合而成.复合服务使用原子服务的关系与社交网络中多个人员参与一个社会组织的隶属关系[22]类似.因此,iSee 使用服务隶属网 *SAN* 抽象复合服务对原子服务的使用关系.*SAN* 本质上是一个二部图,即 $SAN=(N_{cs},N_s,D_{use})$.其中:N_{cs} 和 N_s 分别为复合服务和原子服务的节点集;无向边集 $D_{use}=\{\{cs_i,s_j\}\}$,$cs_i\in N_{cs}$,$s_j\in N_s$ 抽象复合服务 cs_i 对原子服务 s_j 的使用.*SAN* 中复合服务和原子服务 2 类节点间的连接关系可由关联矩阵 $\boldsymbol{\psi}$ 描述,其元素:

$$\psi_{ij}=\begin{cases}1, & \{cs_i,s_j\}\in D_{use},\\ 0, & \text{其他},\end{cases} \tag{1}$$

其中,$\boldsymbol{\psi}$ 是一个 $|N_{cs}|\times|N_s|$ 的二值矩阵.若 $\psi_{ij}=1$,则 $\{cs_i,s_j\}\in D_{use}$,否则 $\{cs_i,s_j\}\notin D_{use}$.

通过对 *SAN* 在原子服务方向上做单模投影[22],可得到原子服务间的可组合关系.

定义 2. 服务组合网(service composition network, *SECON*).$SECON=(N_s,D_c)$ 可用于抽象原子服务间的可组合关系.其中:N_s 同 *SAN* 中 N_s 的定义;D_c 是一个无向边的集合,表示原子服务间的共现关系[3].*SECON* 的关联矩阵 $\boldsymbol{\psi}^s$ 可由 $\boldsymbol{\psi}$ 得到,其元素:

$$\psi^s_{ij}=\sum_k \psi_{ki}\psi_{kj}, \tag{2}$$

其中,ψ^s_{ii} 是使用原子服务 i 的复合服务数,ψ^s_{ij} 是同时使用原子服务 i 和 j 的复合服务数.因此,若 $\psi^s_{ij}>0$ 则 $\{s_i,s_j\}\in D_c$,否则 $\{s_i,s_j\}\notin D_c$.

我们用 ψ^s_{ij} 为 *SECON* 中原子服务 i 和 j 间的边 $\{s_i,s_j\}$ 赋权值.显然,ψ^s_{ij} 可以量度原子服务 i 和 j 可组合关系的强弱.ψ^s_{ij} 值越大,原子服务 i 和 j 组合的可能性越大.

SECON 包含了原子服务间是否可以组合的知识.但是,*SECON* 是由 *SAN* 单模投影得到的,单模投影会引入很多无效的组合关系.如图 2 所示,假设复合服务 CS 由原子服务 S_1,S_2,S_3 构成,可以构建相应的 *SAN*,如图 2 所示.同时,通过对 *SAN* 在原子服务方向作投影可得到相应的 *SECON*,如图 2 右边图所示.在投影时,因 S_1,S_2,S_3 在 CS 中共现,所以 *SECON* 中原子服务两两间都存在连边,表明它们之间可能可以组合.但是,若 S_1,S_2,S_3 间的真实组合关系如图 2 左 ω_s 所示,则 *SECON* 中 S_1 和 S_3 间的边是无效的.因此,原子服务间真实的组合网络 $\omega_s=(N_s,E_s)$ 和 *SECON* 满足 $E_s\in D_c$.随着复合服务数的增加,通过投影得到的 *SECON* 往往会比较稠密,大量的无效边使我们很难准确把握原子服务间真实的组合关系.因此,有必要尽量去除 *SECON* 中的无效边.

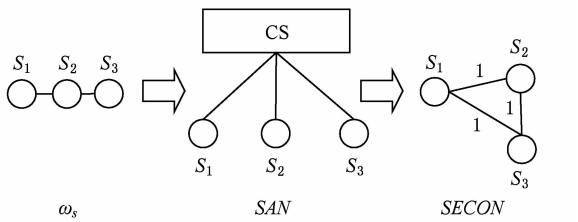


Fig. 2 Illustration of invalid compositions by one-mode projection

图 2 单模投影产生无效组合关系的例子

复杂系统和复杂性科学被认为是"21 世纪的科学",它的基本观点是结构决定功能[24].iSee 用网络模型抽象原子服务间的组合关系,启发我们用复杂网络研究中较成熟的理论及方法过滤 *SECON* 的无效边.

iSee 引入复杂网络研究中的骨干网(backbone network)挖掘方法[23]过滤 *SECON* 中的无效边,挖掘服务组合骨干网.骨干网挖掘在很多领域获得了成功的应用,如食谱分析[23]、航空网分析[25]、议会大选[26]、Internet 网分析[26]等,它主要基于各节点的权重及其分布特征,挖掘具有统计显著意义的边.我们使用 Barabási 研究组在文献[23]中提出的骨干网挖掘算法挖掘服务组合骨干网(service composition backbone network, *SCBN*).下文首先给出概念的定义.

定义 3. 节点权.节点 i 的节点权 s_i 定义为与该节点相连的所有边的权重和,即:

$$s_i=\sum_{j\in v_i} w_{ij}, \tag{3}$$

其中,v_i 是节点 i 的邻居节点集(网络中与节点 i 存在连边的节点集),w_{ij} 是节点 i 和 j 间边 (i,j) 的权值.如图 2 中右图所示,$s_{S_1}=w_{S_1S_2}+w_{S_1S_3}=2$.

定义 4. 归一化边权.边权 w_{ij} 相对于 s_i 的归一化边权 p_{ij} 定义为

$$p_{ij}=w_{ij}/s_i. \tag{4}$$

文献[23]仅将网络中满足 $(1-p_{ij})^{k_i-1}<\alpha$ 的边 (i,j) 保留下来(k_i 是节点 i 的度,α 是给定的一个过滤值).若节点 i 不存在这样的边,则保留边权最大的一条边.求解服务组合骨干网的过程详见算法 1.

其中,Network 是加权无向网络类(类型同 *SECON*),Node 是节点类,nodes 是 SECON 的节点集,getDegree(*i*)返回节点 *i* 的度,getWeight(*i*)返回节点 *i* 的节点权,getNeighbors(*i*)返回节点 *i* 的邻居节点集,w[*i*][*j*]存储边(*i*,*j*)的边权,pow(x,y)等价于 x^y,*alpha* 对应 α.

算法 1. *SCBN* 挖掘算法 *bne*.

输入:SECON、过滤值 *alpha*;

输出:*SCBN*.

```
Network bne(Network SECON,float alpha){
① 构建一个与 SECON 同节点,但没有边的网络 NewNetwork;
② for (Node i:nodes) {
③   int k=getDegree(i);
④   int kBak=k;
⑤   float maxWeight=-1.0;
⑥   if (k>1) {
⑦     float s=getWeight(i);
⑧     for (Node j:getNeighbors(i)) {
⑨       double p=1.0×w[i][j]/s;
⑩       if (pow(1-p,k-1)<alpha)
⑪         在 NewNetwork 中加入边(i,j);
⑫       else{
⑬         if (w[i][j]>maxWeight)
⑭           maxIndex=j;
⑮         kBak--;
⑯       }
⑰     }
⑱     if(kBak=0)
⑲       将边(i,maxIndex)加入 NewNetwork;
⑳   }
㉑   else if(k=1)
㉒     直接在 NewNetwork 中加入这条边;
㉓   else 不做任何处理;
㉔ }
㉕ 返回 NewNetwork.
㉖ }
```

算法 1 的关键步骤是步骤②~步骤㉔的 for 循环,故其时间复杂度是 $O(n^2)$(n 是 *SECON* 的节点数).

服务推荐时,若所推荐的原子服务是不可用的,就得寻找与该原子服务功能相似的可用原子服务进行替代.为此,iSee 引入服务相似网表征原子服务及原子服务间的相似关系.

定义 5. 服务相似网(service similarity network,*SANE*).*SANE* 可以表示为 $SANE=(N_s,D_{sim})$.其中:N_s 为原子服务节点集,D_{sim} 是无向边集,表示边 2 端原子服务间存在相似性,边上权值表示相似性大小.

原子服务的元信息包含了对其功能的描述.通过元信息各个部分(如描述、标签等)相似度的计算可以得到原子服务间功能的相似度.原子服务元信息的各部分本质是长短不一的文档,相似度的计算可以使用向量空间模型[27].

我们以原子服务元信息中描述的相似性计算过程为例说明向量空间模型的实施步骤.首先,使用词向量为每个原子服务的描述信息建立 N 维空间向量:

$$\boldsymbol{d}_j=(w_{1,j},w_{2,j},\cdots,w_{N,j}), \tag{5}$$

其中,$\boldsymbol{d}_j$ 为原子服务 j 的描述信息所对应的空间向量,N 是描述信息中的不同词语(短语)数,$w_{i,j}$ 表示第 i 个词语(短语)的权重.

然后,使用 2 个原子服务描述向量的余弦系数表示原子服务间由于描述而产生的相似性:

$$Sim(\boldsymbol{d}_i,\boldsymbol{d}_j)=Cos(\boldsymbol{d}_i,\boldsymbol{d}_j)=\frac{\sum_{k=1}^{|T|}w_{k,i}\times w_{k,j}}{\sqrt{\sum_{k=1}^{|T|}w_{k,i}^2\times\sum_{k=1}^{|T|}w_{k,j}^2}}, \tag{6}$$

其中,$Sim(\boldsymbol{d}_i,\boldsymbol{d}_j)$ 是 $\boldsymbol{d}_i$ 和 $\boldsymbol{d}_j$ 间的相似性,$Cos(\boldsymbol{d}_i,\boldsymbol{d}_j)$ 是 $\boldsymbol{d}_i$ 和 $\boldsymbol{d}_j$ 间的余弦系数,T 是词(或短语)的集合,$|T|$ 返回 T 中元素个数.

原子服务间的相似性为其元信息各部分(描述、标签等)相似性的线性加权和:

$$Sim(i,j)=\sum_{k=1}^{K}\alpha_k Sim(i_k,j_k), \tag{7}$$

其中,i 和 j 代表 2 个不同的原子服务,$Sim(i,j)$ 是 i 和 j 间的相似度,K 是参与 i 和 j 相似度计算的元信息类别数,i_k 和 j_k 分别代表 i 和 j 的第 k 个类别(如描述、标签等),$Sim(i_k,j_k)$ 代表 i 和 j 在第 k 个类别上的相似度,α_k 是权重值 $\left(\sum_{k=1}^{K}\alpha_k=1\right)$.

同时,原子服务的元信息各部分是长短不一的文本.短文本包含的词语较少且重复率低,长文本包含的词语较多且重复率高.我们分别使用布尔向量空间模型和 TF-IDF 模型[27]为其中的短文本(标签等)和长文本(描述等)设置词语权重 $w_{i,j}$.为了客观地设置式(7)中的权值 α_k,我们引入统计实践中常用的变异系数法(coefficient of variation method,*CV*)[28].

需要指出的是,本文主要以离线的方式构建各类服务网络,即首先从服务注册中心爬取服务的元信息,并存于本地数据库;然后从本地数据库读取信息构建各类网络.我们并未考虑在构建服务网络的过程中又有新服务加入的情况,因为我们是定期爬取服务信息的.当然,我们构建的服务网络也极易动态扩展.一旦有新的复合服务进来,只需对新的复合服务构建 SAN 和 $SECON$,然后将新构建的 $SECON$ 并入原来的 $SECON$ 中(增加新的节点、新的边或改变已有边的边权);骨干网更新的时候也只需重新考虑是否将受影响的边(新加入的边和边权改变的边)及节点加入到骨干网中;服务相似网更新的时候,只需依次计算新加入的原子服务与网络中原有原子服务间的相似度,并将新原子服务及相应的边加入相似网络.

2.3 服务使用模式

SAN 抽象了复合服务对原子服务的使用历史,从 SAN 投影得到的 $SECON$ 和 $SCBN$ 自然包含了原子服务间的可组合关系.实际上,$SECON$ 和 $SCBN$ 的任意路径都是一个潜在的复合服务.但是原子服务间是如何组合在一起的呢?为了回答这个问题,我们有必要对服务的使用模式进行分析.

我们主要围绕 2 个问题对服务使用模式进行分析:

问题 1. 一个复合服务通常由多少个原子服务构成.

问题 2. 原子服务是否更倾向于和流行的原子服务连接.

iSee 使用网络模型抽象复合服务对原子服务的使用关系及原子服务间的组合关系.我们可以借鉴复杂网络中的结构分析方法揭示服务的使用模式.为此,我们将首先引入复杂网络研究中的 2 个统计指标.

定义 6[29]. 度数中心度(degree centrality, DC).复杂网络研究中,节点的度数中心度描述了网络中与该节点相连的节点数.

在 SAN 中,复合服务的 DC 描述了其使用的原子服务数.因此,我们可以通过分析复合服务 DC 的分布规律得到构成一个复合服务所需原子服务数的范围$[A,B]$.例如,若 95% 的复合服务使用的原子服务数均在$[1,5]$内,便可将上述$[A,B]$定为$[1,5]$.

定义 7[30]. 度分布.度分布 $P(k)$表示一个随机选择的节点度恰好是 k 的概率,常用累积度分布来定义,即 $P_{\text{cum}}(k)=P(K>k)\sim k^{-\beta}$.

当网络的度分布符合幂律分布时,那么这个网络就是"无标度"网络[30].在无标度网络中,大部分节点只与少数节点相连,而极少的节点与大量节点相连.因此,若原子服务的度分布是无标度的,则表明开发者"择优"[30]选择原子服务构造复合服务.换言之,开发者倾向于选择流行的原子服务构建复合服务.

2.4 服务推荐算法

复合服务开发过程中,原子服务的使用场景大致可分为 3 种[3]:1)开发者还未选择原子服务;2)开发者已选择了 1 个原子服务;3)开发者已选择了 $multi(multi>1)$个原子服务.对于情况 1,一旦开发者选定了一个原子服务,就自动转化为情况 2.iSee 将服务的推荐问题转化为在 $SCBN$ 中搜索包含给定节点的路径的过程.路径中开发者已选节点外的节点代表的原子服务即为推荐的原子服务.然而,若 $SCBN$ 规模大且稠密,则其包含指定节点且长度任意的路径数会很庞大,难以推荐开发者真正需要的组合方案.因此,本文主要关注下一个可组合服务的推荐,而完整的复合服务则通过不断地交互进行构造.

但是,原子服务处在一个开放、动态的环境中,服务的可用性不能持续保证.若用户选择的原子服务失效,我们必须为其推荐功能相似的可替换服务,而原子服务间的功能相似性蕴含在 $SANE$ 中.因此,iSee 方法将 $SCBN$,$SANE$,服务的使用模式、服务失效数据等作为服务推荐的基础设施,为开发者推荐服务.

通过 $SANE$ 可以找到与失效服务相似的服务集.但是这仅仅利用的是服务间的相似性,忽略了服务流行性上的差异,即在相似服务集中,有些服务可能已经被广泛使用(在 SAN 中 DC 值大),有些很少使用(在 SAN 中 DC 值小),有些甚至从未被使用过.因此,在计算可替换服务集时,为了综合考虑服务间的相似度及服务的流行性,我们提出了相似服务的排序指标 $sRank$:

$$sRank_j=\alpha_1 w_{ij}+\alpha_2\frac{k_j}{maxDeg},\tag{8}$$

其中,$sRank_j$ 表示节点 j 的 $sRank$ 值,w_{ij} 是节点 i 与 j 间的相似度,k_j 是节点 j 在 SAN 中的度,$maxDeg$ 是 SAN 中原子服务节点的最大 DC 值,α_1 和 α_2 是相应分量的权值,且 $\alpha_1+\alpha_2=1$,代表相应分量对 $sRank$ 的贡献程度.若 $sRank$ 与服务的流行性无关(网络非无标度),则可设置 $\alpha_1=1,\alpha_2=0$.

iSee 按如下步骤计算可替换服务集:1)在 *SANE* 中寻找与失效服务相邻的所有邻居节点;2)依据 *sRank* 值降序排列这些邻居节点;3)返回 *sRank* 值最大的若干邻居节点作为可替换服务集.算法 2 所示的是单个失效服务的可替换服务推荐算法 *resReal*(replaceable service recommendation algorithm).其中,Network 是加权无向网络类,Node 是节点类,*nodes* 是 *SANE* 的节点数组,*j.sRank*表示节点 *j* 的 *sRank* 成员值,*j.k* 是节点 *j* 的度,α_1 和 α_2 是通过 2.2 节所述 CV 求得的加权系数,*w*[*ss*][*j*]是边(*ss*,*j*)上的权值.

算法 2. 可替换服务推荐算法 *resReal*.

输入:*SANE*、存储失效服务数据的数组 *fs*、单个失效服务 *ss*、待推荐的服务数 *resNum*;

输出:推荐的服务数组.

```
Node[] resReal(Network SANE,Node[] fs,
Node ss,int resNum) {
① Node[] nodeBak=new Node[nodes.length];
② int maxDeg=0;
③ for (Node i:nodes) {/*nodes 为 SAN 节点
    集*/
④   int deg=getDegree(i);
⑤   if (deg≥maxDeg) /*求节点最大度*/
⑥     maxDeg=deg;
⑦   if (ss 与 i 在 SANE 中有边相连)
⑧     if (i 不在 fs 中)
⑨       将节点 i 加入 nodeBak 数组;
⑩ }
⑪ for (Node j:nodeBak) {
⑫   j.sRank =α1×w[ss][j]+α2×
      j.k/maxDeg;
⑬ }
⑭ BubbleSort(nodeBak); /*按节点 sRank 值
    冒泡排序(降序)*/
⑮ 返回 sRank 值 Top-resNum 的服务集进行
    推荐;
⑯ }
```

算法 2 的关键步骤是 for 循环(步骤③~步骤⑩)的步骤④,故其时间复杂度是 $O(n^2)$(n 是 *SANE* 网络节点数).

iSee 为上述 3 种原子服务的不同使用场景都提供了相应的推荐方法.

场景 1. 开发者还未选择原子服务.

开发者还未选择原子服务时的服务推荐类似于推荐系统中的"冷启动"问题,这时我们将向开发者推荐"平台"原子服务[3].“平台”原子服务流行度很高,参与了大部分复合服务的构建.因此,新构建的复合服务也很可能选择这些"平台"原子服务.为此,iSee 将 *SAN* 中的服务按照 *DC* 值降序排列,并将 Top-*psNum* 个服务推荐给用户(*psNum* 是待推荐的服务数).

算法 3. 平台服务推荐算法 *psReal*.

输入:*SAN*、待推荐的服务数 *psNum*;

输出:度 Top-*psNum* 的服务数组.

```
Node[] psReal(Network SAN,int psNum) {
① for (Node i:nodes1) /*nodes1 为 SAN 原
    子服务节点数组*/
②   for (Node j:nodes2) /*nodes2 为 SAN
      复合服务节点数组*/
③     if (i 与 j 相连)
④       i.degee++;
⑤ BubbleSort(nodes); /*按度值冒泡排序(降
    序)*/
⑥ 返回度值 Top-psNum 的服务集进行推荐;
⑦ }
```

算法 3 的关键步骤是 for 循环(步骤①~步骤④)内的步骤③,故其时间复杂度是 $O(nm)$(n,m 分别是 *SAN* 的原子服务节点数和复合服务节点数).

场景 2. 开发者已选择了 1 个原子服务.

iSee 主要是基于 *SCBN* 为用户推荐服务,而原子服务只有与其他原子服务在复合服务中共现才能出现在 *SCBN* 中.对于一些仅参与一个复合服务构建(该复合服务仅由一个原子服务构成)或未参与任何复合服务构建的原子服务是不可能出现在 *SCBN* 中的.因此,对于选择了 1 个原子服务的服务推荐问题,iSee 将分 2 种情况:1)已选服务不属于无效服务集 *fs*,并且存在于 *SCBN* 中;2)情况 1 之外的情况.对于情况 1,我们可以直接在 *SCBN* 中搜索与该服务直接相连的服务来推荐.对于情况 2,iSee 将首先为用户推荐可替换的 Top-*resNum* 服务,一旦用户选择了某个服务,再按照情况 1 为用户推荐服务.iSee 将不在 *SCBN* 中的服务也看成失效服务,因为它们没有历史使用记录可以利用.

那么,如何对待推荐的服务按某种优先级排序呢?如 2.3 节所述,服务的排序应融入服务使用模式的知识.因此,若服务的度分布是幂律分布,则服务更倾向于和流行的服务连接.为了综合考虑 *SCBN* 中服务连接强度和度分布对服务优先级的

影响,我们引入曾在文献[31]中提出的加权 k-核分解方法(weighted k-core decompostition method, $W_{k\text{-core}}$)分析服务节点的优先级.

$W_{k\text{-core}}$基于节点的加权度从整体角度分析节点重要性.加权度指标综合考虑了节点的度和边权对节点重要性的影响,符合本文对服务优先级的定义.节点 i 的加权度 $wDeg(i)$定义为

$$wDeg(i)=\left[Deg(i)^{\alpha}\left(\sum_{j=1}^{Deg(i)}w_{ij}\right)^{\beta}\right]^{\frac{1}{\alpha+\beta}},\quad(9)$$

其中,$wDeg(i)$是节点 i 的加权度,$Deg(i)$是节点 i 的度,$\sum_{j=1}^{Deg(i)}w_{ij}$ 是与节点 i 相连的边的边权和.α 和 β 是相应分量的权值,且 $\alpha+\beta=1$,代表相应分量对 $wDeg(i)$的贡献程度.本文不区分节点度和节点边权和对 $wDeg(i)$的贡献大小,设置 $\alpha=\beta=0.5$,即 $wDeg(i)=\sqrt{Deg(i)\sum_{j=1}^{Deg(i)}w_{ij}}$.因此,对于无权网络($w_{ij}=1$),$wDeg(i)$等价于 $Deg(i)$.对于加权网络,$wDeg(i)$往往是一个正实数.我们将其离散化成与其最接近的整数.

假设网络 $G=(V,E)$是由 $|V|$个节点和 $|E|$条边组成的一个加权无向网络,则与 $W_{k\text{-core}}$相关的概念定义如下:

定义 8. 网络的加权 k-核(weighted k-core).是指反复去掉加权度值小于或等于 k 的节点及其连边后剩余的子图,并且该子图是具有这一性质的最大子图.

定义 9. 节点的加权核数(weighted coreness) k.表示该节点存在于加权 k-核中,但是在加权($k+1$)-核中被删除,则该节点的加权核数为 k.节点加权核数的最大值 $k_{\max}$称为网络的加权核数.

本文将节点的加权核数作为节点排序的优先级,值越大排名越靠前.对于无效的服务其加权核数值为0.已选1个原子服务的推荐过程如算法4所示.

算法 4. 已选 1 个原子服务的推荐算法 *sigReal*.

输入:*SCBN*,*SANE*、存储失效服务数据的数组 *fs*、用户已选服务 *selS*、待推荐的方案数 *sigNum*、算法 2 参数 *resNum*;

输出:推荐的服务集合.

Node[] *sigReal*(Network *SCBN*, Network *SANE*, Node[] *fs*, Node[] *selS*, int *sigNum*, int *resNum*) {

① boolean *flag*; int $i=0$;

② Node[] *resNodes*; /* 存储替换服务集 */

③ if (*selS*[0]属于 *fs* || *selS*[0]不在 *SANE*)

④ *flag*=true;

⑤ if (*flag*=true) {

⑥ *resNodes*=*resReal*(*SANE*, *fs*, *selS*[0], *resNum*); /* 调用算法 2 */

⑦ 输出 *resNodes* 中 Top-*resNum* 的可替换服务,等待用户选择;

⑧ *selS*[0]=*resNodes*[*s*]; /* 若用户选择了第 *s* 个服务 $s\in[0,resNum]$ */

⑨ }

⑩ for (Node *i*; *nodes*)

/* *nodes* 为 *SCBN* 的节点集 */

⑪ if (*i* 与 *selS*[0]相连) {

⑫ 将节点 *i* 放入数组 *sibNodes*;

⑬ 计算节点 *i* 的加权核数值;

⑭ }

⑮ *BubbleSort*(*sibNodes*); /* 按加权核数冒泡排序(降序) */

⑯ 返回 *sibNodes* 中 Top-*sigNum* 的服务集,等待用户选择.

⑰ }

算法 4 的关键步骤是循环(步骤⑩~步骤⑭)中的步骤⑬.步骤⑬的时间复杂度是 $O(n+e)$[31],故算法 4 的时间复杂度是 $O(n^2+ne)$(n, e 分别是 *SCBN* 的节点数和边数).

场景 3. 开发者已经选择了 $multi(multi>1)$个原子服务.

对于选择了 $multi(multi>1)$个原子服务的服务推荐问题,iSee 将分 2 种情况:1)已选原子服务不属于无效服务集 *fs*,并且存在于 *SCBN* 中;2)情况 1 之外的情况.对于情况 1,我们可以直接在 *SCBN* 中搜索包含已选服务的路径来推荐服务.对于情况 2,我们必须按照算法 2 为失效的服务推荐可替换服务.同时,如 2.3 节所述,复合服务一般由$[A,B]$个以内的原子服务构成,所以 iSee 限定推荐的简单路径节点数在 B 以内.推荐算法如算法 5 所示.

算法 5. 已选 *multi* 个服务的推荐算法 *mulReal*.

输入:*SCBN*, *SANE*, 存储失效服务数据的数组 *fs*、存储用户已选服务的数组 *selS*、待推荐的方案数 *mulNum*、算法 2 参数 *resNum* 及路径最大节点数 B;

输出:推荐的服务集合.

Node[] *mulReal*(Network *SCBN*, Network

SANE, Node[ ] *fs*, Node[ ] *selS*, int *B*, int *resNum*) {

① boolean *flag*[ ] = new boolean[*selS*.*length*];
② int *cnt*=0;Node[] *resNodes*;
③ for (int *i*=0;*i*<*selS*.*length*;*i*++) {
④ if (*selS*[*i*]不属于 *fs* ‖ *selS*[*i*]不在 *SANE*)
⑤ *flag*[*cnt*++]=*i*;
⑥ }
⑦ for (int *i*=0;*i*<*cnt*;*i*++) {
⑧ *resNodes* = *resReal* (*SANE*, *fs*, *selS*[*flag*[*i*]],*resNum*);
⑨ 输出 *resNodes* 中 Top-*resNum* 的可替换服务,等待用户选择;
⑩ *selS*[*flag*[*i*]]=*resNodes*[*s*]; /* 若用户选择第 *s* 个服务 *s*∈[0,*resNum*] */
⑪ }
⑫ if (*isInOneComp*(*selS*)) {/* *isInOneComp*()判断节点集是否在同一连通图 */
⑬ 求*selS* 中任一节点对间长度在[|*selS*|,*B*]内、包含 *selS* 中全部服务节点的所有简单路径,并存储在 Node[][] *path* 中;
⑭ if (*path* 中没有任何路径)
⑮ 输出提示信息"所选节点间不存在长度[|*selS*|,*B*]内的历史组合信息!"
⑯ else {
⑰ *BubbleSort*(*path*); /* 按路径加权核数值和冒泡排序(降序) */
⑱ 按长度分类输出 Top-*mulNum* 的路径供用户选择; /* 长度为|*selS*|,|*selS*|+1,…,*B* 分类输出 */
⑲ }
⑳ }
㉑ else 输出提示信息"所选节点间不存在历史组合信息!"
㉒ }

算法 5 的关键步骤是 for 循环(步骤⑦~⑪)中的步骤⑧,故其时间复杂度是 $O(n^2)$(n 是 *SANE* 网络节点数).

因此,总的服务推荐算法 *IsReal* 如算法 6 所示.

算法 6. 服务推荐算法 *IsReal*.

输入: *SANE*, *fs*, *ss*, *resNum*, *SAN*, *psNum*, *SCBN*, *selS*, *sigNum*, *mulNum*, *B*;

输出:推荐的服务集合.

① while (true) {
② 接受用户输入的服务,存于 *selS* 数组;
③ if (*selS* 元素个数=0) {
④ 按照算法 3 的 *psReal* 推荐服务;
⑤ }
⑥ else if (*selS* 元素个数=1) {
⑦ 按照算法 4 的 *sigReal* 推荐服务;
⑧ }
⑨ else if (*selS* 元素个数≤*B*){
⑩ 按照算法 5 的 *mulReal* 推荐服务;
⑪ }
⑫ else {
⑬ 输出提示信息"您选择的服务过多,请重新输入",同时清空 *selS* 数组;
⑭ }
⑮ }

算法 6 的时间复杂性分析详见各种场景相应的算法部分,此处不再赘述.

3 实例分析

为了说明 iSee 的具体实施过程,同时验证其正确性和有效性,本文将结合 PWeb 上 mashup 应用和开放 API 的真实数据,进行实证分析.

3.1 数据来源

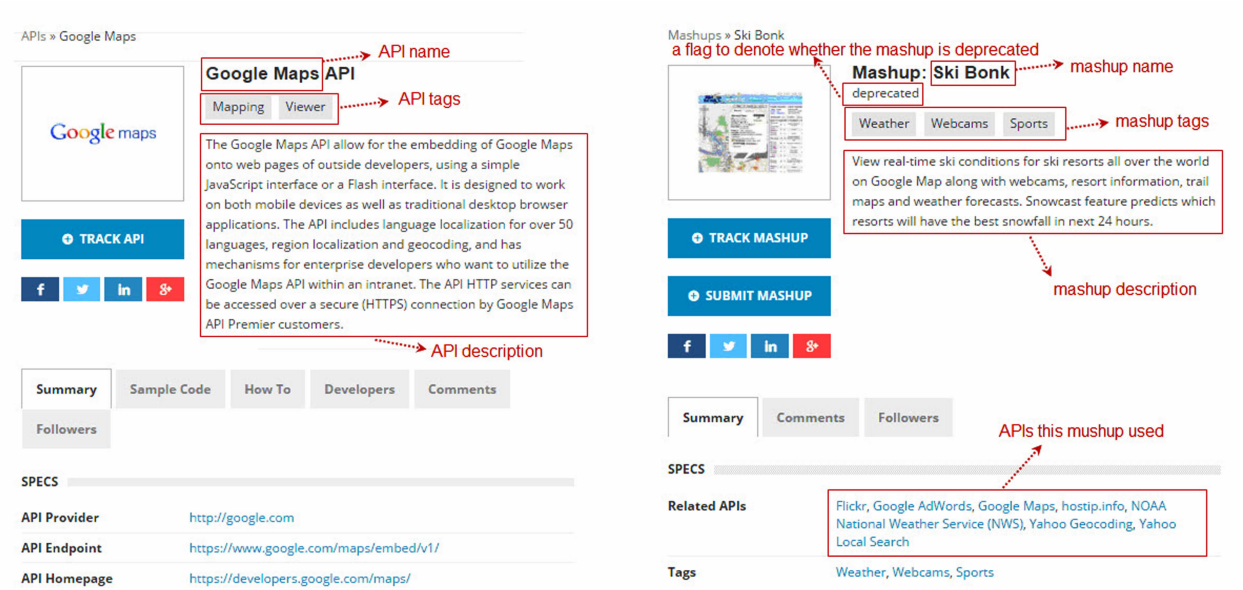
PWeb 是目前最大和最活跃的 mashup 应用和开放 API 注册库,提供了 mashup 和 API 的一些注册元信息,如图 3 所示. mashup 是由 API 组合而成的,包含了对 API 的使用历史. 因此,可以将 API 看成原子服务,mashup 看成复合服务. 同时,若某一 API 失效了,在其名字旁和描述中都会出现"Deprecated". 因此,失效 API 服务集也容易获得. PWeb 数据集符合 iSee 方法对数据的要求.

我们使用的数据集是 PWeb 上截止 2011 年 12 月 14 日的所有 mashup 应用和开放 API 的元信息(图 3 中矩形框内信息),并按文献[3]中的方法对数据降噪. 最终,实验数据集含复合服务 6 362 个(mashup 应用)、原子服务 4 506 个(开放 API).

3.2 实验过程及结果

如图 3(b)所示,每个 mashup 应用都有一个构成该 mashup 的开放 API 列表. 我们据此构建了所有 mashup 应用和其所用 API 间的 *SAN*,共包含

6 362 个 mashup 节点和 982 个 API 节点. 如前所述, 爬取的数据集中 API 共有 4 506 个, 但最终 *SAN* 中的 API 只有 982. 可见, PWeb 上 API 的使用率比较低, 被使用过的 API 仅占总 API 数的 21.8%.



(a) The meta-information of open API (Google Maps①)　　(b) The meta-information of mashup (Ski Bonk②)

Fig. 3 Meta-information of open API and mashup

图 3 开放 API 及 mashup 应用的元信息页

一旦构建了 *SAN*, 就可以根据 *SAN* 分析开放 API 的使用模式. 图 4 所示的是 mashup 的 *DC* 分布. 从图 4 可见, 95.8%的 mashup 应用其使用的开放 API 数≤5, 仅有 4.2%的 mashup 应用使用的开放 API 数≥10. 这说明大部分 mashup 应用仅由少量的开放 API 构成. 因此, 我们将 2.3 节中提到的范围[*A*, *B*]确定为[1, 5]. 同时, 将 *DC* 排名 Top-10 的 API, 如表 1 所示, 作为"平台"API[3].

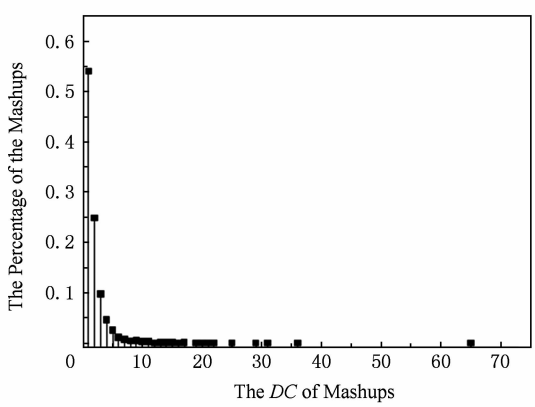


Fig. 4 *DC* distribution of mashup

图 4 mashup 应用的 *DC* 分布

Table 1 Top-10 Open APIs with high *DC*

表 1 度数中心度 Top-10 的开放 API

Names of Open APIs	*DC*
Google Maps	2 303
Twitter	661
Flickr	590
YouTube	589
Amazon eCommerce	403
Twilio	333
Facebook	320
eBay	214
Last. fm	206
Google Search	178

图 5 所示的是开放 API 在双对数轴上的 $P_{cum}(k)$. 从图 5 可见, $P_{cum}(k)$接近直线, 评价拟合精度的指标 R^2 达到了 0.98 以上, 满足幂律分布. 幂律分布表明, 开发者倾向于选择流行的开放 API 构建 mashup. 因此, 式(8)中 $a_2 \neq 0$. 本文不区分服务的相似性与流行性对 *sRank* 的影响, 设置 $a_1 = a_2 = 0.5$.

① http://www.programmableweb.com/api/google-maps

② http://www.programmableweb.com/mashup/ski-bonk

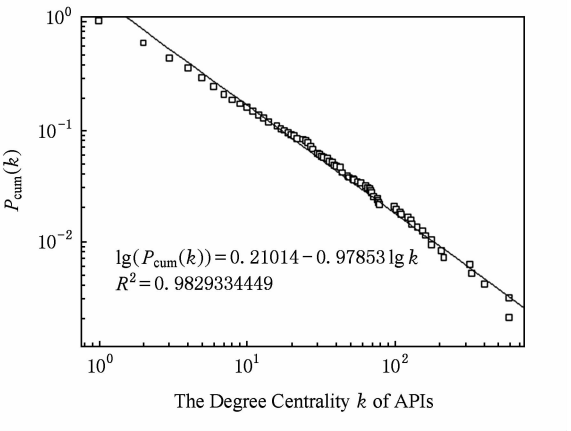


Fig. 5 Cumulative degree distributuion of open API (log-log scale)

图 5 开放 API 的累积度分布(双对数轴)

通过对 *SAN* 在 API 方向作投影操作,可以得到 *SAN* 相应的 *SECON*,包含 865 个节点和 10 966 条边.投影产生了 117 个孤立节点,它们没有与其他 API 在同一个 mashup 中共现. *SECON* 排除了这类孤立节点. *SECON* 的平均 *DC* 达到了 25,网络比较稠密,包含了很多无效边.为此,我们使用骨干网挖掘算法(算法 1)过滤 *SECON* 中的无效边.最终,我们得到的 *SCBN* 包含 865 个节点、7 702 条边.

同时,我们根据开放 API 的描述和标签计算了 API 间的相似度(其中:描述用 TF-IDF 模型计算、标签用布尔向量空间模型计算),构造了 *SANE*(包含 4 506 个节点和 8 629 141 条边).

3.3 实验结果分析

在本节中,我们将重点围绕 4 个问题对实验结果进行深入分析.

3.3.1 骨干网挖掘算法过滤无效边的能力检验

要检验骨干网挖掘算法在过滤无效边上的有效性,必须有服务间真实连接关系的数据. PWeb 上 mashup 内部 API 间真实的连接关系是无法直接获取的.为此,我们使用在文献[32]中构建的数据集.通过解析工作流源文件获取工作流与 Web 服务间的关系,我们构建了 *SAN*(包含 261 个工作流和 358 个 Web 服务).通过对 *SAN* 在 Web 服务方向作投影获得了相应的 *SECON*(包含 358 个 Web 服务和 4 252 条边).

我们首先以步长 0.01 在区间[0,1]之间取了 101 个点,然后分别以这 101 个点作为 *alpha*(算法 1 参数)的取值过滤 *SECON* 得到 101 个 *SCBN*.我们依次计算了这 101 个 *SCBN* 包含真实服务连边的比例(|(*SCBN* 边集∩真实服务间连边集)|/|真实服务间连边集|)及 *SCBN* 中无效边的比例(|*SCBN* 中边集-(*SCBN* 中边集∩真实服务间连边集)|/|无效边集|).结果如图 6 所示:

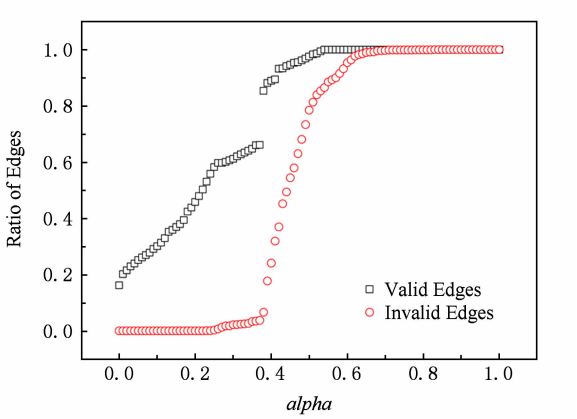


Fig. 6 The curve of valid and invalid edg ratio versus *alpha*

图 6 有效边与无效边比例随 *alpha* 的变化情况

从图 6 可见,当 $alpha\in[0.60,1.00]$,*SECON* 的边并没有明显的减少;当 $alpha\in[0.50,0.60)$,无效边的比例显著降低,而有效边的比例没有减少;当 $alpha\in[0.40,0.50)$,无效边的比例显著下降,有效边的比例缓慢下降;当 $alpha\in[0.00,0.40)$无效边的减少基本稳定,有效边的比例显著减少.由此可见,骨干网挖掘算法对于过滤无效边确实可以起到一定作用.

文中若没有特别说明,取 $alpha=0.45$.在这种设置下,整个网络的边数没有太明显的减少,可以尽量去除无效边,保留有效边. *alpha* 的最优设置方法此处不做讨论,因为在去掉部分无效边的 *SECON* 上做推荐始终优于在最原始的 *SECON* 上做推荐.

3.3.2 iSee 方法推荐有意义 API 的能力检验

我们设计了人工实验以检验本文方法的有效性.我们邀请了 10 人参与本次实验,包括 3 名同事、3 名博士研究生和 4 名硕士研究生.3 名同事从事服务计算相关研究已有 4 年之久,3 名博士研究生也在服务计算领域从事了 3 年的研究,3 名硕士研究生在服务计算领域也从事了 2 年的研究.他们对于 mashup 应用的开发及 PWeb 平台都比较熟悉,但是他们事先并不了解 iSee 方法.

在实验中,他们将各自独立开发一个与旅游相关的 mashup 应用,但是只能使用我们的平台(该平台实现了 iSee)及平台提供的 API 资源.他们可以根据自己的经验,从选择 0 个、1 个或多个 API 开始,增量式的开发 mashup 应用.根据开发者的开发

情境,我们将给用户提供 Top-10 的 API 推荐意见,同时将检测用户是否选择我们推荐的服务,并记录选择了哪几个服务、在推荐列表中的排名、推荐的时间(实施推荐到用户选择该推荐的总时间)等信息.若用户对我们推荐的结果不满意也可以通过关键词检索的方式查找相应的服务.

为了客观地量度 iSee 在服务推荐方面的准确度,我们计算了平均排序分(average rank score)[33].对于某一开发者 u 而言,服务 s 的排序分 RS_{us} 定义为

$$RS_{us}=\frac{l_{us}}{L_u}, \tag{10}$$

其中,L_u 表示可为开发者 u 推荐的服务总数,l_{us} 是开发者 u 选择的服务 s 在 L_u 中的排名.将所有的开发者的排序分求算数平均便能得到系统的排序分 RS.排序分值越小,说明系统越趋于把满足用户需求的服务排在前面;反之,说明系统把用户需要的服务排在了越后面.

在实验中,10 个参与人共进行了 28 次选择,我们发现他们选择的服务都排在推荐的服务列表的前 6,平均 RS 为 0.3255.这说明 iSee 方法在辅助开发复合服务方面还是可以起到一定作用的,其推荐的服务能满足开发者的需求.

3.3.3 iSee 方法与同类方法的对比

目前服务推荐方面的工作很多,但是从结构角度(服务网络)出发的工作并不多.在本节中,iSee 将仅与从结构角度出发的 LFH 方法[3]及 BIGSIR 方法[32]进行比较.

为了比较各个方法推荐效果上的差异,我们需要分别为这 10 个参与者的每次选择构造最优服务推荐列表.为此,我们又请这 10 位参与者不使用 iSee 的推荐功能,仅通过 PWeb 的搜索功能来寻找 API,构建满足其需求的 mashup 应用,以检验是否存在比 iSee 方法推荐的 Top-10 服务更好的服务.若存在这样的服务,我们便将其与 iSee 的推荐结果合并,然后由他们人工对这些服务进行打分排序,构建新的 Top-10 服务推荐列表.我们将由这 10 个参与者各自给出的 Top-10 服务推荐列表看成是最优的服务推荐列表.

在实验中,仅 2 个参与者各自发现了一个比 iSee 方法 Top-10 中更好的服务,而且这 2 个方案在最终人工构造的 Top-10 推荐列表中排名都比较靠后,分列 8,9 位.我们请这 10 个参与者人工对 iSee 方法的 Top-10 结果进行排序,以构造最优排序结果.对于其中 2 位发现了新服务的参与者,我们将新发现的服务与 iSee 方法为其推荐的 Top-10 服务一起参与排序.我们发现,iSee 方法排名 1～6 位的推荐服务集和排名 7～10 位的推荐服务集与最优排序 1～6 位的服务集和 7～10 位的服务集(新增 2 个除外)基本是一致的,只是服务的相对排名有差异.

为了得到 LFH 方法和 BIGSIR 方法的推荐结果,我们重复了 3.3.2 节中的实验,仅服务推荐分别使用 LFH 和 BIGSIR 方法.在实验中,我们发现:对于 LFH 方法,仅 17 个采纳了推荐的 Top-10 服务列表中的服务,而且选择的服务都排在 6～10 位间,其余 11 个通过人工搜索查找;对于 BIGSIR 方法,仅 10 个采纳其服务推荐意见,而且选择的服务都排在 6～10 位间,其余 18 个通过人工搜索查找.

服务推荐实质是对服务的排序.因此,比较 iSee,LFH,BIGSIR 之间的优劣,可以通过计算每种方法的排序结果与最优排序的相似度来衡量.Kendall 等级相关系数 τ_B[34]常用于量度 2 种排序结果的相似性及其强弱.本文将使用 τ_B 量度各种推荐方法排序结果与最优排序结果的相似关系.τ_B 定义为

$$\tau_B=\frac{C-D}{\sqrt{(N_3-N_1)(N_3-N_2)}}, \tag{11}$$

其中,C 和 D 分别为元素对集合 XY 中一致性的和不一致性的元素对数;$N_3=n(n-1)/2$;$N_1=\sum_{i=1}^{s}t_i(t_i-1)/2$;$N_2=\sum_{j=1}^{t}u_j(u_j-1)/2$.$n$ 是集合 X 和 Y 的元素个数.N_1 是根据 X 计算的:s 是 X 中的小集合(相同元素自成一小集合)数,t_i 是第 i 个小集合中的元素数.N_2 可根据 Y 类似地求得.

使用 τ_B 计算 2 个排序的相似性有一个前提条件,即 2 个排序是对相同的元素集合进行排序.iSee 的 Top-10 服务集与最优服务集元素基本一致.LFH 和 BIGSIR 的推荐服务集与最优服务集之间元素差别比较大.LFH 的结果集与最优服务集平均只有 2.4 个元素相同;BIGSIR 的结果集与最优服务集平均只有 1.2 个元素相同.因此,我们无法用 τ_B 量度 LFH 和 BIGSIR 的排序结果与最优排序的相似性.因此,我们使用 τ_B 仅量度 iSee 推荐服务集与最优排序集的相似关系(结果如表 2 所示).因每个参与者多次选择的 τ_B 存在显著度不一致的问题,无法求其均值.表 2 中显示的是参与者在多次选择中 τ_B 的最低值.在计算 2 个排名的相关性系数时,对于新增了 2 个推荐意见的参与者,我们将新增加

的推荐意见放在第 11 位,而最优排序将其分别排在了第 8 和第 9 位.

Table 2 τ_B Between Results of iSee and Optimum

表 2 iSee 方法排序结果与最优排序的 τ_B

Participants	τ_B	Participants	τ_B
Participant 1	0.556*	Participant 6	0.467
Participant 2	0.511*	Participant 7	0.564*
Participant 3	0.467	Participant 8	0.600*
Participant 4	0.491*	Participant 9	0.644*
Participant 5	0.289	Participant 10	0.422

Note: * Correlation is significant at the 0.05 level (two-tailed).

从表 2 可见,iSee 方法给 10 位参与者的推荐结果在多数情况下都与其相应的最优推荐至少存在显著性水平 0.05 的正相关性,相关系数在 0.4 以上,仅在某几次选择中不存在明显的相关性.但是至少它们的元素基本是一致的.可见,iSee 方法在推荐效果上比 LFH 和 BIGSIR 均要好.

在实验中,一位参与者选择了一个无效的开放 API—alexa Web search,一个参与者选择了一个没有被任何复合服务使用过的服务——500px.iSee 均可以为这 2 位参与者提供可替换的服务,从而顺利的完成推荐.而这 2 种情况,LFH 和 BIGSIR 都是无能为力的.因此,从为失效服务及无历史使用历史的服务实施推荐来看,iSee 要优于 LFH 和 BIGSIR.

3.3.4 iSee 方法的扩展性如何

作为一个实际的服务推荐方法,iSee 将被运用于不同规模的数据集.若一个服务推荐方法时间开销很大,用户往往是无法忍受的.因此,我们将检验 iSee 在不同规模数据集上实现服务推荐的时间开销.

我们设计了 2 个实验以检验 iSee 的扩展性.iSee 涉及很多与用户的交互操作,如失效服务的选择、推荐服务的选择等.为了能高效地获取算法运行的时间开销,我们假设:若服务失效,默认选择排名第一的可替换服务;用户每次都选排名第一的推荐服务;推荐的服务数为 10.

实验 1. 数据规模对算法运行时间的影响.

SAN,*SECON*,*SANE* 等网络都是根据 mashup 和构成 mashup 的 API 列表定义的.网络的规模往往以网络中的节点数来描述.为了分析数据规模对算法性能的影响,我们以 600 的倍数随机采样得到的原始数据集,构建各种规模的网络(*SCBN* 是在 $alpha=0.45$ 时构建的),并考虑用户的不同输入情况,测试算法的运行时间,结果如图 7 所示:

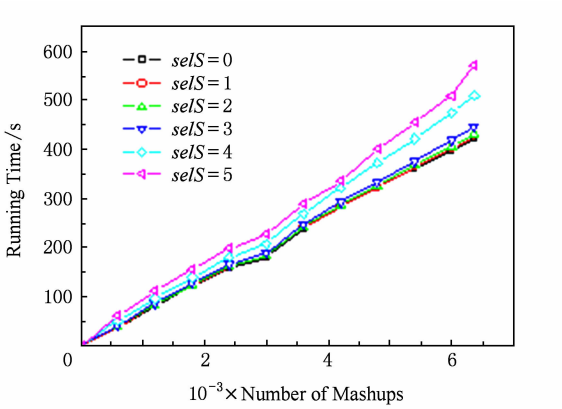


Fig. 7 The curve of running time versus data scale

图 7 运行时间与数据规模的变化曲线

为了提高结果的稳定性,图 7 中的每个数据点是 100 次独立实验得到结果的平均值.图 7 中不同颜色的曲线代表不同的用户输入情况(*selS* 是算法 4 和算法 5 中的参数,代表用户已选择的服务数).算法的运行时间包含了各种网络的构建时间,这 2 部分的时间与 $selS=0$ 时的情况基本一致.然而网络并不需要每次都动态构建,一般通过离线方式定期更新即可(随着大量的新 mashup 的加入才需要更新各网络).因此,若除去网络构建的时间,推荐算法还是比较高效的.从图 7 我们可以发现,算法运行时间随着 mashup 数量和 *selS* 值的增加呈现线性增长的趋势.但是,即使 mashup 规模增长了 5 倍,*selS* 增长到 5,算法运行时间仍然低于 2.5 min.在大部分情况下,用户选择的服务数都不大于 3,算法基本都可以在 20 s 内完成推荐,这么短的时间完全可以给用户以实时推荐.iSee 适用于大规模服务的情况.

实验 2. 已选服务数对算法运行时间的影响.

用户已选服务数会影响算法的搜索策略.为了分析用户已选服务数对算法性能的影响,我们以3.2节构造的数据集为实验对象(*SCBN* 是在 $alpha=0.45$时构建的),并考虑用户的不同输入情况,测试算法的运行时间,结果如图 8 所示.

图 8 中的每个数据点都是 100 次独立实验所得结果的平均值.*selS* 代表用户已选服务数.算法的运行时间同样也包含了各种网络的构建时间.从图 8 可见,算法运行时间随着 *selS* 值的增加呈现不断增长的趋势.当 $selS\leqslant 3$ 时,时间增长缓慢.当 $selS>3$ 时,时间增长快速.因为随着 *selS* 的增大,API 间组合的可能性成倍增加.

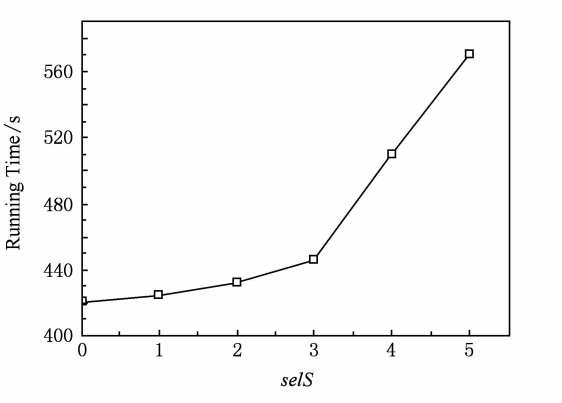


Fig. 8 The curve of running time versus *selS*

图 8 运行时间与用户已选服务数(*selS*)的变化曲线

4 相关研究

服务推荐技术是解决服务资源过载问题的有效方法之一,近年来已涌现了大量的研究工作.

Blake 和 Nowlan[11]通过对用户操作会话的监控获取文本信息,并量度该文本信息与 Web 服务描述文档的相似性,从而定位用户感兴趣的服务集,实现推荐.

Shao 等人[12]通过用户对相同 Web 服务的体验评价用户间的相似性,并使用基于用户的协同过滤算法预测其未使用 Web 服务的 QoS 信息,从而向用户推荐具有较高 QoS 的服务.

Zheng 等人[13]将基于用户的推荐算法和基于商品的推荐算法系统的融合起来预测 Web 服务的 QoS 信息,并实现了推荐系统 WSRec.

Zhao 等人[35]通过语义技术发现服务间隐含的关联,并使用网络模型抽象服务及它们之间的关联,提出了一种通过服务间关联发现用户所需服务的方法.用户通过关键字搜索便可获得相关的服务集.

Chen 等人[14]发现用户的地理位置与 Web 服务的 QoS 间存在高相关性.因此,他们构建了一种融入用户地理位置信息的 RegionKNN 算法来预测服务的缺失 QoS 信息,进而指导服务推荐.相比于以往的方法,该方法预测的 QoS 信息更加准确,推荐效果更好.

Deng 等人[18]基于网络建模和分析技术研究了用户和服务之间的信任关系,提出了基于信任协同过滤的服务推荐算法提供个性化的服务推荐.

Jiang 等人[32]构建复合服务与原子服务的二部图,并引入物质流动算法实现服务推荐.

Su 等人[19]认为,现有的基于协同过滤的方法在预测服务 QoS 时忽略了服务历史 QoS 信息的可靠性.因此,他们提出了一种信任感知的服务 QoS 预测方法,进而向用户推荐具有较高 QoS 的服务.

Kang 等人[20]提出了一种融入用户潜在 QoS 偏好及用户对服务兴趣多样性特征的 Web 服务推荐方法.

Xu 等人[21]综合考虑用户端和服务端的情境信息,提出了一种情境感知的 Web 服务 QoS 预测方法,并应用于服务推荐和选择.

我们曾在文献[3]中将“软件网络观”引入服务分类及推荐中,提出服务分类及推荐的 LFH 方法.但是该工作主要关注服务的分类,并未对推荐工作展开深入的探讨,只是给出在服务推荐方面可能的应用方向.本文的工作是文献[3]及文献[32]的深入研究,是复杂网络思想在服务推荐领域的进一步实践,与其他的服务推荐工作存在很大差异.本文提出的系统方法以及算法和评估策略,均可以应用在其他服务推荐研究.

5 结论与展望

针对现有服务推荐方法中存在的数据难以获取和未考虑所推荐服务的可用性及与已有服务的可组合性等问题,提出了一种基于服务组合历史的交互式服务推荐方法,给出了服务隶属关系、服务组合关系和服务相似关系的服务网络模型,并引入学科交叉思想,用复杂网络相关原理及方法分析服务使用模式、去除无效服务关系等,并进而指导服务推荐,为从网络科学角度解决服务计算相关问题提供了一种思路.本文的主要工作有 3 个方面:

1) 将复合服务、原子服务及它们间的使用关系用隶属网抽象,并通过单模投影得到原子服务间的可组合关系,进而借助复杂网络中的骨干网挖掘技术挖掘原子服务间真实的组合关系,并通过度和度分布挖掘原子服务使用模式,指导服务推荐算法的设计.

2) 依据服务的 3 种使用场景设计了相应的推荐方法.方法考虑了服务的失效问题.

3) 以 PWeb 上真实的服务数据验证所提方法的正确性和有效性.

下一步的研究工作主要包括:提供自适应的参数设置方法,同时与基于语义、语法、协同过滤等的推荐方法融合起来,进一步提高服务推荐的效率和精准度.

参 考 文 献

[1] Papazoglou M P. Service-oriented computing: Concepts, characteristics and directions [C] // Proc of the 4th Int Conf on Web Information Systems Engineering (WISE 2003). Piscataway, NJ: IEEE, 2003: 3-12

[2] Li Gang, Ma Xiujun, Han Yanbo, et al. Transparent service composition in dynamic network environments [J]. Chinese Journal of Computers, 2007, 30(4): 579-587 (in Chinese)
(李刚, 马修军, 韩燕波, 等. 动态网络环境下的透明服务组合[J]. 计算机学报, 2007, 30(4): 579-587)

[3] Pan Weifeng, Li Bing, Shao Bo, et al. Service classification and recommendation based on software networks [J]. Chinese Journal of Computers, 2011, 34(12): 2355-2369 (in Chinese)
(潘伟丰, 李兵, 邵波, 等. 基于软件网络的服务自动分类和推荐方法研究[J]. 计算机学报, 2011, 34(12): 2355-2369)

[4] Cardoso J. Quality of service and semantic composition workflows [D]. Athens, GA: University of Georgia, 2002

[5] Li Zeng, Wang Jian, Zhang Neng, et al. A topic-oriented clustering approach for domain services [J]. Journal of Computer Research and Development, 2014, 51(2): 408-419 (in Chinese)
(李征, 王健, 张能, 等. 一种面向主题的领域服务聚类方法[J]. 计算机研究与发展, 2014, 51(2): 408-419)

[6] Zhang Liangjie, Zhang Jia, Cai Hong. Service Computing [M]. Berlin: Springer, 2007

[7] Jannach D, Zanker M, Felfernig A, et al. Recommender Systems: An Introduction [M]. Cambridge: Cambridge University Press, 2010

[8] Sun Huifeng, Zheng Zibin, Chen Junliang, et al. Personalized Web service recommendation via normal recovery collaborative filtering [J]. IEEE Trans on Services Computing, 2013, 6(4): 573-579

[9] Klusch M, Kapahnke P. iSeM: Approximated reasoning for adaptive hybrid selection of semantic services [C] // Proc of the 4th IEEE Int Conf on Semantic Computing (ICSC 2010). Piscataway, NJ: IEEE, 2010: 184-191

[10] Liu Liwei, Lecue F, Mehandjiev N, et al. Using context similarity for service recommendation [C] // Proc of the 4th IEEE Int Conf on Semantic Computing (ICSC 2010). Piscataway, NJ: IEEE, 2010: 277-284

[11] Blake M B, Nowlan M F. A Web service recommender system using enhanced syntactical matching [C] // Proc of the 5th IEEE Int Conf on Web Services (ICWS 2007). Piscataway, NJ: IEEE, 2007: 575-582

[12] Shao Lingshuang, Zhang Jing, Wei Yong, et al. Personalized QoS prediction for Web services via collaborative filtering [C] // Proc of the 5th IEEE Int Conf on Web Services (ICWS 2007). Piscataway, NJ: IEEE, 2007: 439-446

[13] Zheng Zibin, Ma Hao, Lyu M R, et al. WSRec: A collaborative filtering based Web service recommender system [C] //Proc of the 7th IEEE Int Conf on Web Services (ICWS 2009). Piscataway, NJ: IEEE, 2009: 437-444

[14] Chen Xi, Liu Xudong, Huang Zicheng, et al. RegionKNN: A scalable hybrid collaborative filtering algorithm for personalized Web service recommendation [C] //Proc of the 8th IEEE Int Conf on Web Services (ICWS 2010). Piscataway, NJ: IEEE, 2010: 9-16

[15] Zheng Zibin, Ma Hao, Lyu M R, et al. QoS-aware Web service recommendation by collaborative filtering [J]. IEEE Trans on Services Computing, 2011, 4(2): 140-152

[16] Zhang Qiong, Ding Chen, Chi Chihung. Collaborative filtering based service ranking using invocation histories [C] //Proc of the 9th IEEE Int Conf on Web Services (ICWS 2011). Piscataway, NJ: IEEE, 2011: 195-202

[17] Chen Xi, Zheng Zibin, Liu Xudong, et al. Personalized QoS-aware Web service recommendation and visualization [J]. IEEE Trans on Services Computing, 2013, 6(1): 35-47

[18] Deng Shuiguang, Huang Longtao, Wu Jian, et al. Trust-based personalized service recommendation: A network perspective [J]. Journal of Computer Science and Technology, 2014, 29(1): 69-80

[19] Su Kai, Xiao Bin, Liu Baoping, et al. TAP: A personalized trust-aware QoS prediction approach for Web service recommendation [J]. Knowledge-Based Systems, 2017, 115(1): 55-65

[20] Kang Guosheng, Tang Mingdong, Liu Jianxun, et al. Diversifying Web service recommendation results via exploring service usage history [J]. IEEE Trans on Services Computing, 2016, 9(4): 566-579

[21] Xu Yueshen, Yin Jianwei, Deng Shuiguang, et al. Context-aware QoS prediction for Web service recommendation and selection [J]. Expert Systems with Applications, 2016, 53: 75-86

[22] Newman M E J. The structure of scientific collaboration networks [J]. Proceedings of the National Academy of Sciences, 2001, 98(2): 404-409

[23] Strogatz S H. Exploring complex networks [J]. Nature, 2001, 410(6825): 268-276

[24] Ahn Y Y, Ahnert S, Bagrow J P, et al. Flavor network and the principles of food pairing [OL]. [2016-07-01]. https://www.nature.com/articles/srep00196

[25] Serrano M A, Boguna M, Vespignani A. Extracting the multiscale backbone of complex weighted networks [J]. Proc of the National Academy of Sciences, 2009, 106(16): 6483-6488

[26] Lee S H, Kim P J, Ahn Y Y, et al. Googling social interactions: Web search engine based social network construction [J]. Plos One, 2010, 5(7): e11233

[27] Salton G, Wong A, Yang Chungshu. A vector space model for automatic indexing [J]. Communications of ACM, 1975, 18(11): 613-620

[28] McClave J T, Benson P G, Sincich T. Statistics for Business and Economics [M]. 10th ed. Upper Saddle River, NJ: Prentice Hall, 2008

[29] Wikipedia. Centrality definition [EB/OL]. [2016-05-05]. http://en.wikipedia.org/wiki/Centrality

[30] Barabási A L, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286(5439): 509-512

[31] Pan Weifeng, Hu Bo, Jiang Bo, et al. Identifying important packages of object-oriented software using weighted *k*-core decomposition [J]. Journal of Intelligent Systems, 2014, 23(4): 461-476

[32] Jiang Bo, Zhang Xiaoxiao, Pan Weifeng, et al. BIGSIR: A bipartite graph based service recommendation method [C] // Proc of the 9th IEEE World Congresss on Services (SERVICES 2013). Piscataway, NJ: IEEE, 2013: 363-369

[33] Zhou Tao, Ren Jie, Medo M, et al. Bipartite network projection and personal recommendation [J]. Physical Review E, 2007, 76(4): 046115

[34] Pan Weifeng, Li Bing, Ma Yutao, et al. Identifying the key packages using weighted PageRank algorithm [J]. Acta Electronica Sinica, 2014, 42(11): 2174-2183 (in Chinese)
(潘伟丰, 李兵, 马于涛, 等. 基于加权 PageRank 算法的关键包识别方法[J]. 电子学报, 2014, 42(11): 2174-2183)

[35] Zhao Chenting, Ma Chun'e, Zhang Jing. HyperService: Linking and exploring services on the Web [C] // Proc of the 8th IEEE Int Conf on Web Services (ICWS 2010). Piscataway, NJ: IEEE, 2010: 17-24

Pan Weifeng, born in 1982. PhD, associate professor and master supervisor. Member of CCF. His main research interests include software engineering, service computing and complex networks.

Jiang Bo, born in 1970. PhD, professor and master supervisor. Member of CCF. Her main research interests include service computing and cloud computing (nancybjiang@zjgsu.edu.cn).

Li Bing, born in 1969. PhD, professor and PhD supervisor. Senior member of CCF. His main research interests include complex networks, service computing and cloud computing (bingli@whu.edu.cn).

Hu Bo, born in 1983. PhD and researcher. His main research interests include software enginnering, cloud computing and big data (bob_hu@kingdee.com).

Song Beibei, born in 1989. Master candidate. Her main research interests include software engineering, evolutionary algorithm and complex networks (betty8408@qq.com).