基于软件网络的服务自动分类和推荐方法研究

潘伟丰[1),2)] 李 兵[1),3)] 邵 波[1)] 何 鹏[1)]

[1)](武汉大学软件工程国家重点实验室 武汉 430072)

[2)](浙江工商大学计算机与信息工程学院 杭州 310018)

[3)](武汉大学复杂网络研究中心 武汉 430072)

摘 要 服务的分类和推荐方法对于服务管理和组合都具有重要意义.文中利用服务组合历史,从面向服务软件的结构角度研究了服务分类和推荐方法:提出面向服务软件的软件网络模型,并用于抽象面向服务软件的拓扑结构;利用复杂网络社区发现算法对软件网络进行聚类,从而实现服务的自动分类;利用软件网络描述服务之间的可组合关系及其强度,并基于服务的使用场景提出了相应的服务推荐算法.使用 ProgrammableWeb 上 API 服务和 mashup 应用的真实数据来说明该文方法的有效性.从实验结果可以看出,该方法可以准确地实现服务分类,整个聚类的纯净度达到了 86.8%,同时推荐的服务也比较有意义,这对于解决服务计算中的一些问题,包括服务的管理、检索、组合等都具有一定的意义.

关键词 服务分类;服务组合;社区发现;复杂网络;服务计算

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02355

Service Classification and Recommendation Based on Software Networks

PAN Wei-Feng[1),2)] LI Bing[1),3)] SHAO Bo[1)] HE Peng[1)]

[1)](*State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072)

[2)](*School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou* 310018)

[3)](*Complex Networks Research Center, Wuhan University, Wuhan* 430072)

Abstract Service classification and recommendation are two important problems in service management and composition. This paper, utilizing service composition histories, investigates the method for service classification and recommendation from a topological perspective of service-oriented software (SOS). It proposes a software network model for SOS to abstract its topological structures; it uses a community detection algorithm to cluster software networks and realizes service classification; it uses software networks to describe compositional strength between services and proposes an algorithm for service recommendation. API services and mashup applications in ProgrammableWeb are used as subjects to demonstrate the feasibility of the proposed approach. Experimental results show a high cluster purity of 86.8%. Finally we discuss the implications of our approach on service computing.

Keywords service classification; service composition; community detection; complex networks; service computing

收稿日期:2011-08-05;**最终修改稿收到日期**:2011-10-29.**本课题得到国家"九七三"重点基础研究发展规划项目基金**(2007CB310801)、**国家自然科学基金**(60873083,61003073)、**中国博士后科学基金**(20090460107,201003794)**和中央高校基本科研业务费专项资金**(114013,3101032)**资助**.**潘伟丰**,**男**,1982 **年生**,**博士**,**讲师**,**中国计算机学会**(CCF)**会员**,**主要研究方向为软件工程**、**复杂网络和智能计算**. E-mail: wfpan@mail.zjgsu.edu.cn.**李 兵**,**男**,1969 **年生**,**教授**,**博士生导师**,**中国计算机学会**(CCF)**高级会员**,**主要研究领域为软件工程**、**服务计算和云计算**.**邵 波**,**男**,1988 **年生**,**硕士研究生**,**主要研究方向为服务计算**.**何 鹏**,**男**,1988 **年生**,**硕士研究生**,**主要研究方向为软件工程和复杂网络**.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

1 引 言

面向服务的计算(Service-Oriented Computing,SOC)是针对分布式系统的新型计算模式,为解决分布异构环境中跨组织敏捷应用集成问题带来了诸多便利,已成为软件领域最热门的话题之一[1]. SOC倡导以服务及其组合为基础构造应用的开发模式,导致软件系统的主要形态、生产方式、运行方式和使用方式都发生了巨大变化,软件系统正处在一个由软件服务实体组成的开放协同的软件环境中[2]. 从传统的基于SOAP的Web服务到目前REST风格的Web服务以及mashup/API服务等,服务的种类日益丰富,应用领域也不断拓展,以服务为中心的互联网正在形成.

随着软件体系结构研究的深入,人们逐渐意识到软件结构是决定软件质量的重要因素[3]. 一些研究者基于软件的网络结构(软件网络),采取软件工程和复杂网络相关理论相结合的研究方法,以分析软件结构特征为着眼点,以理解软件的结构特征为软件开发和维护提供支持为目标,从表征、分析、度量和应用4个方面对软件的静态结构进行系统分析,取得了可喜的成果,这些成果对指导软件的开发和维护具有重要的指导意义[4-12]. 如今面向服务软件(Service-Oriented Software,SOS)日益成为主流,我们有必要研究SOS的结构,挖掘结构中蕴含的知识,从而更好地指导SOS的开发. 不断出现的在线服务和SOS注册中心(如ProgrammableWeb①、seekda②、myExperiment③等)为我们从结构角度研究SOS提供了一个契机.

服务发现和服务组合吸引了国内外众多学者的关注,成为SOC领域中的关键问题[13]. 但是目前的服务缺少分类信息,给服务发现、检索以及服务资源的管理带来不便. 如果采用手工的方式对服务分类,当服务数量很大时显然是不可取的,因此如何实现服务的自动分类构成SOC的挑战. 与此同时,服务组合中服务的可组合关系往往是根据服务间的输入输出来进行判断的,均遵循了文献[14]中的服务匹配原则,但这一原则过于严格. 那么除了输入输出匹配外,是否存在一种新的服务匹配方法能够挖掘服务间的潜在可组合关系?

本文针对上述两个问题展开研究,通过对服务组合历史,即已有SOS结构的分析,挖掘结构中蕴含的知识,最终实现服务的自动分类和服务推荐(文中将SOS看成大粒度的服务). 我们把这种方法称为从历史学习的方法(Learn From History,简称LFH方法),并以ProgrammableWeb上API服务和mashup应用的真实数据为例进行验证. LFH方法将面向对象(Object-Oriented,OO)软件的软件网络模型引入SOS研究中,提出了SOS的软件网络模型. 基于该软件网络模型,提出了基于复杂网络社区发现的服务自动分类方法. 同时,提出了一种服务推荐方法,该方法可以推荐与一个或多个服务匹配的其它服务. 实例研究表明,本文的方法在实现服务自动分类和服务推荐方面是有效的.

本文第2节介绍相关工作;第3节将详细地介绍LFH方法,包括SOS的软件网络模型、服务的分类方法、服务的推荐算法等;第4节将以ProgrammableWeb上的mashup、API服务的真实数据验证LFH方法的有效性;第5节介绍本文方法的应用平台;最后是结论与展望.

2 相关研究

复杂网络为我们研究系统结构及其动力学行为提供了有力的工具. 近年来,一些研究者将复杂网络相关理论引入软件工程中,构建软件网络模型,用于分析OO软件的结构特征及其动力学行为. 复杂网络理论强调从整体上认识、理解和控制系统,而非关注局部. 研究系统在整体上表现的特征可以为我们理解软件系统提供新的维度.

2002年,Valverde等人[4]首先将复杂网络方法引入软件结构分析中. 他们将系统的类图用无向网络来抽象,即网络中的节点表示系统中的类,节点间的边表示类间的关系,如继承(inheritance)、关联(association)等. 他们用这种方法抽象JDK 1.2和UbiSoft ProRally 2002,并用复杂网络中的相关方法分析这些软件网络的统计特性. 他们发现,这两个软件的软件网络具有与其它复杂网络类似的"小世界"和"无标度"特性. 同时,他们认为软件开发中的局部优化过程可能是导致软件网络呈现这些特性的原因.

在软件系统中,系统模块(类、方法等)之间的关系(协作、调用等)反映了系统的控制流. 因此,软件

① ProgrammableWeb官方网站. http://www.programmableweb.com
② seekda官方网站. http://webservices.seekda.com
③ myExperiment官方网站. http://www.myexperiment.org

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

网络中的边的方向是有意义的[5]. 2003 年，一些研究者使用有向网络研究软件结构. Myers 使用有向网络表示软件系统的结构，分析了 3 个 OO 软件的类协作图(class collaboration graph)和 3 个面向过程软件的静态过程调用图(static procedure call graph). 他们发现：(1) 这些有向软件网络尽管来自不同系统，甚至是用不同类型语言(面向过程/对象)开发的系统，但是都具有“小世界”和“无标度”特性；(2) 出入度分布的幂指数存在不平衡性，即出度分布的幂指数大于入度分布的幂指数；(3) 出度与入度之间存在负相关性，他们认为这可能与软件重用有关，即简单的类(出度小的类)容易被重用，因此入度大. Valverde 和 Sole 使用有向网络描述软件系统的类图(class diagram)对系统的结构进行研究. 他们发现，类粒度的软件网络除了具有“小世界”和“无标度”特性外，同时具有层次性(hierarchical)和模块性(modularity)[6].

之后，人们开始从不同粒度(方法、类、包、文件等)对大量开源软件的结构进行研究. Moura 等人将 C 和 C++语言开发的系统在文件级(头文件)抽象成网络，即头文件为节点，头文件的共现关系抽象成边. 他们发现这样的网络也具有“小世界”和“无标度”特性，并指出：“软件随时间增长是软件呈现无标度特性的原因，软件性能的优化是软件具有“小世界”特性的原因”[7]. LaBelle 和 Wallingford[8] 在包粒度将软件系统看成网络，对一些软件的依赖网络进行分析，发现这样形成的网络同样具有“小世界”和“无标度”特性. Valverde 和 Sole[9] 将类和类的方法都抽象为节点，方法访问类的关系抽象成类节点和方法节点间的边，他们发现这样构建的软件网络也同样具有“小世界”和“无标度”特性. 国内的研究人员在这方面也做了些实证研究，如李德毅、李兵等人[10-12] 对 Java 语言编写的若干软件系统以及 Linux 等进行分析，发现了其中的“小世界”和“无标度”特性.

以上这些工作都是针对面向过程或 OO 软件展开研究的，针对 SOS 的工作很少见到. 但是随着近年来 SOC 的快速发展，面向服务的思想已经被许多软件工程师所接受. 基于服务的应用正在获得强劲发展势头，因此有必要分析 SOS 的结构，以指导 SOS 的管理和开发等工作. 本文的工作正是围绕这点展开的，主要探讨服务的自动分类和推荐方法，因此与以前针对面向过程软件和 OO 软件所开展的工作有显著区别.

3　LFH 方法

从面向结构的程序设计到面向对象/构件程序设计，再到面向服务的软件开发方式，软件的设计和实现的重点已从实现局部的编程难题，转向了如何将代码有效地进行组织，特别是对于大规模复杂软件系统而言，结构的组织就尤为重要. 与此同时，使用自然语言或符号语言(如 OWL-S[15])这种一维的知识表示方式来描述服务，无法从全局的角度呈现 SOS 的复杂结构. LFH 方法使用软件网络这种二维知识表示方法抽象 SOS 的拓扑结构，并进而指导服务的自动分类和推荐. 图 1 给出了 LFH 方法的基本框架. 以下各小节将对框架中的部分内容进行说明，并给出相关概念的定义.

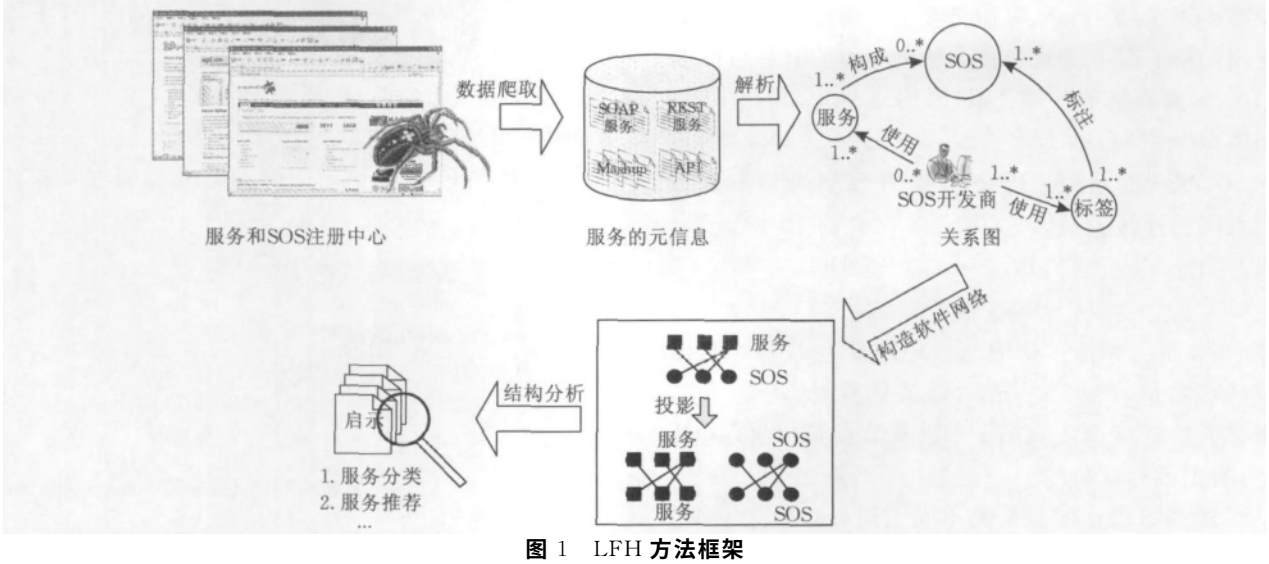


图 1　LFH 方法框架

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

3.1 数据收集

随着服务数量的增加，在线的服务注册中心为服务开发者和服务消费者提供了一个服务交易的场所，如 ProgrammableWeb、seekda、myExperiment 等. 注册中心通常是按照特定的注册模型进行注册的，包含了服务的元信息，如服务的名称、描述、地址、标签、提供者等. 对于一些由服务构成的 SOS（大粒度服务）甚至包含构成这个 SOS 的服务列表.

LFH 方法将收集注册中心中各类服务以及由这些服务构成的 SOS 的元信息（如名称、描述、标签等），并存于本地数据库. 这些爬下来的数据可能存在一些噪声，如作为标签的单词拼写错误、服务重复注册等. LFH 方法对收集到的数据进行预处理，通过一些算法修正单词的拼写错误、删除重复注册的服务等，尽量排除数据中的噪声.

数据的收集可以借助一些整站下载器，如 Teleport Ultra①、webzip②、WinHTTrack③ 等来完成，即：首先，用整站下载器将网站中的相关页面以 html 静态页面的形式爬到本地；然后，编写一些脚本解析 html 页面中的相关字段存于本地数据库中；最后，再对收集到的数据进行预处理以减少噪声.

3.2 SOS 的软件网络模型

SOS 通常由若干（≥ 0）服务辅以一定的编程组合而成，因此可以看成一种更大粒度的服务. 文中将由单个服务构成的软件看成一类特殊的 SOS. SOS 和服务之间的这种宏观组合关系可以使用网络模型来抽象. 下面首先给出 SOS 相应软件网络的定义.

定义 1（SOS-服务网络）. SOS-服务网络（简称 SOS2S）可以用一个二部图表示，即 $SOS2S=(N_{SOS},N_s,D)$. 其中，N_{SOS} 为 SOS 节点的集合，表示所有 SOS 的集合；N_s 是服务节点的集合，表示 SOS 使用的所有服务的集合；D 是一个无向边的集合，表示 SOS 对服务的使用关系. SOS2S 的边只存在于不同类型节点间，即 $D=\{\{sos_i,s_j\}\}$，$sos_i\in N_{SOS}$，$s_j\in N_s$. SOS2S 的连接矩阵 ψ 描述了 SOS 节点和服务节点间的连接关系：

$$\psi_{ij}=\begin{cases}1, & \{sos_i,s_j\}\in D\\ 0, & \text{其它}\end{cases} \tag{1}$$

其中，ψ 是一个 $A\times B$ 的二值矩阵，$A=|N_{SOS}|$ 表示 SOS 的数量，$B=|N_s|$ 表示服务的数量. 若 $\psi_{ij}=1$，则第 i 个 SOS 节点和第 j 个服务节点间存在有边相连，否则不存在边.

SOS2S 表示的是两类不同节点（SOS 节点和服务节点）间的关系. 同类型节点间的关系（SOS 和 SOS 的关系或服务和服务的关系）可以通过对 SOS2S 作投影得到（如图 2 所示）.

定义 2（SOS-SOS 网络）. SOS-SOS 网络（简称 SOS^2）可以用一个单模无向图表示，即 $SOS^2=(N_{SOS},D_{SOS})$. 其中，N_{SOS} 为 SOS 节点的集合，表示所有的 SOS 的集合；D_{SOS} 是一个无向边的集合，表示 SOS 间共用服务这种耦合关系，即若两个 SOS 共用服务，则代表这两个 SOS 的节点间存在一条无向边. SOS^2 的连接矩阵 ψ^{SOS} 可由 ψ 得到，即

$$\psi_{ij}^{SOS}=\sum_k \psi_{ik}\psi_{jk} \tag{2}$$

其中，ψ_{ij}^{SOS} 表示第 i 个 SOS 和第 j 个 SOS 共有的服务的数量，ψ_{ii}^{SOS} 表示第 i 个 SOS 使用的服务的数量. 因此，若 $\psi_{ij}^{SOS}>0$ 则第 i 个 SOS 和第 j 个 SOS 之间存在共用服务的耦合关系（$\{sos_i,sos_j\}\in D_{SOS}$），它们共用 ψ_{ij}^{SOS} 个服务；否则这两个 SOS 间没有共用任何服务（$\{sos_i,sos_j\}\notin D_{SOS}$）.

同时，我们给 SOS^2 中每条边赋予一个权值，用于刻画这条边两端节点代表的 SOS 间的相似度. 第 i 个 SOS 节点和第 j 个 SOS 节点之间的边权 w_{ij}^{SOS} 定义如下：

$$w_{ij}^{SOS}=a\times w_{ij}^s+b\times w_{ij}^t \tag{3}$$

其中，w_{ij}^s 是通过两个 SOS 共同使用的服务计算得到的相似度；w_{ij}^t 是通过使用的标签计算得到的相似度；a 和 b 是相应分量的权重，且满足 $0\leq a,b\leq 1$，$a+b=1$. 通过调节 a 和 b 的值，我们可以满足不同的服务分类需求. SOS 使用的服务代表了其功能，如果我们主要从 SOS 的功能构成角度对其分类，则 a 可以设置得相对大些；SOS 使用的标签代表了提供者对该 SOS 的认识，如果我们主要从提供者认知角度对其进行分类，则 b 可以设置得相对大些. 本文中 $a=0.9$，$b=0.1$.

我们使用 Jacard 相似度系数（Jacard similarity coefficient）来计算两个 SOS 之间的相似度. 给定两个集合 E 和 F，则它们之间的 Jacard 相似度系数计算如下：

$$simlarity(E,F)=\frac{|E\cap F|}{|E\cup F|} \tag{4}$$

所以 w_{ij}^s 和 w_{ij}^t 可以定义为

$$w_{ij}^s=\frac{|N_s^i\cap N_s^j|}{|N_s^i\cup N_s^j|} \tag{5}$$

① Teleport Ultra 官方网站 http://www.tenmax.com/teleport/ultra/home.htm
② webzip 官方网站 http://www.spidersoft.com/webzip
③ WinHTTrack 官方网站 http://www.httrack.com

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

$$w_{ij}^{t}=\frac{|N_t^i\cap N_t^j|}{|N_t^i\cup N_t^j|} \tag{6}$$

其中，N_s^i和N_s^j(N_t^i和N_t^j)分别表示第i个和第j个SOS使用的服务集合(标签集合).

定义3(服务-服务网络). 服务-服务网络(简称S^2)可以用一个单模无向图表示，即$S^2=(N_s, D_s)$. 其中，N_s为服务节点的集合，表示所有SOS所使用服务的集合；D_s是一个无向边的集合，表示服务间的共现关系，即若两个服务共同作为某个SOS的构成部分，则代表这两个服务的节点间存在一条无向边. S^2的连接矩阵ψ^s可由ψ得到，即

$$\psi_{ij}^{s}=\sum_k \psi_{ki}\psi_{kj} \tag{7}$$

其中，ψ_{ij}^s表示服务i和j共同参与的SOS的数量，ψ_{ii}^s表示服务i参与的SOS的数量. 因此，若$\psi_{ij}^s>0$则服务i和j之间存在共现关系($\{a_i,a_j\}\in D_a$)，否则不存在共现关系($\{a_i,a_j\}\notin D_a$).

同时，我们也给S^2中每条边赋予一个权值，用于刻画这条边两端节点代表的服务间可以组合的可能性. 服务节点i和j间边的权值w_{ij}^s定义如下：

$$w_{ij}^{s}=\frac{1}{N_s^k-1}\psi_{ij}^{s} \tag{8}$$

其中，N_s^k是第k个SOS使用的服务的数量. 显然，w_{ij}^s描述了服务i和j间可以组合的可能性大小. w_{ij}^s值越大，则服务i和j间可以组合的可能性越大. 因此，S^2实质上包含了从已有SOS中获得的服务之间是否可以组合的知识. 可以想象，SOS越多，服务之间可以组合的知识就越丰富，当SOS的数量趋于无穷时，S^2中将包含服务间的所有可组合关系. 但是，SOS的数量往往是有限的，所以已有SOS中服务之间真实的连接网络$\omega_s=(N_s,E_s)$和S^2满足

$$E_s\cap D_s\neq\varnothing \tag{9}$$

图2显示的是从我们收集的真实API服务和mashup应用数据(详见第4节)中构建的SOS2S、SOS^2和S^2. 其中，mashup应用对应SOS，API服务对应服务.

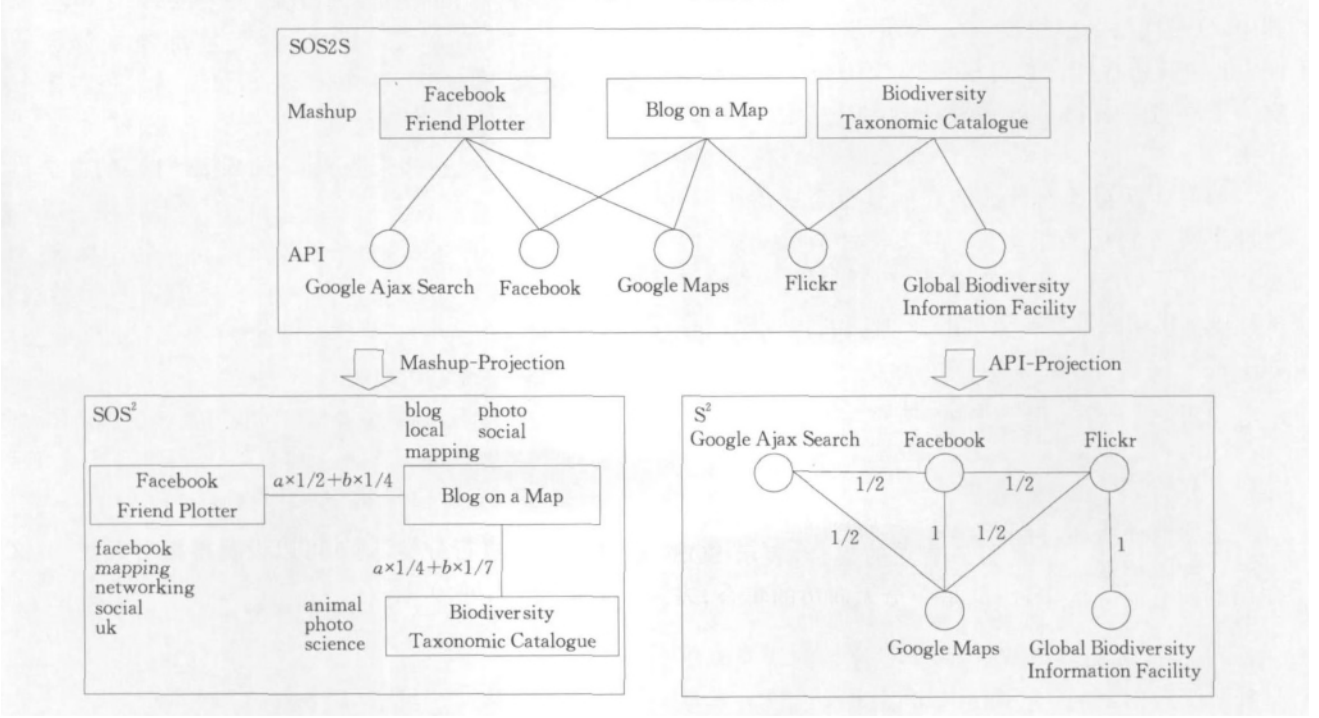


图2　SOS2S、SOS^2和S^2构建举例

3.3　SOS和服务的自动分类

3.3.1　SOS的自动分类

社区结构是复杂网络中一个重要的特性，它是一种介于宏观和微观之间的网络特征，是真实世界中许多复杂网络所具有的一种普遍性质. 研究表明，社区结构和网络的功能(鲁棒性、高速传播性等)有着紧密的关系.

网络社区结构的检测是揭示网络结构和功能之间关系的重要基础，成为了最近几年的研究热点. 复杂网络采用图作为表示形式，因此很多在计算机领域被广泛应用的图分割算法都可以直接应用于复杂网络的社区检测，如Kernighan-Lin算法[16]、谱分割法[17-18]等. 但传统图分割算法倾向于把网络分成规模相近的子图或要求网络的社区数目已知，而这两点往往是很难满足的，因此复杂网络领域出现了很多新的社区发现方法，按照边的连接权值可以分为：

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

无权网络的社区发现方法[19-20]和加权网络的社区发现方法[21-22]；按照节点的隶属度可以分为：硬划分的社区发现方法[20-22]和模糊社区发现方法[23-24]；按照发现的策略可以分为：层次算法[20-25]（分裂和凝聚）和最优化算法[26]；或者分为全局社团发现方法[20-22,25-26]和局部社区发现方法[27-28]等.

本文使用复杂网络社区发现算法，对 SOS^2 进行社区划分，可以对大量 SOS（大粒度服务）进行聚类，进而对服务进行聚类. 目前用于评价社区划分质量的指标不少：Mancoridis 等人提出了模块化质量 MQ（Modularization Quality）；Tucker 等人提出 EVM 函数（EVM Function）；Newman 提出了模块度 Q[29]. 本文使用应用比较广的模块度 Q 作为评价指标. 鉴于本文中待划分的网络 SOS^2 是一个加权网络，我们使用针对加权网络的 Q 来评价社区划分的质量.

$$Q=\sum_i (we_{ii}-wa_i^2) \tag{10}$$

其中，Q 是网络的模块度；we_{ii} 表示两个端点都在社区 i 的边其边权和占总边权和的比例；wa_i 表示至少有一个端点在社区 i 的边其边权和占总边权和的比例.

如何计算 Q 值对算法的性能具有重要的影响，特别当网络的规模很大时. 本文将计算 Q 的增值 ΔQ，而不是 Q，这一思路与文献[29]同. 将一个节点从社区 i 移动到社区 j 造成的 Q 值的变化可以计算如下：

$$\Delta Q=\begin{cases} we_{ij}+we_{ji}-2wa_i wa_j, & \text{社区 } i \text{ 和 } j \text{ 是相连的} \\ 0, & \text{否则} \end{cases} \tag{11}$$

正如前面所述，目前用于社区划分的算法很多，本文从算法复杂性角度出发，选择使用 Newman 提出的快速算法对 SOS^2 进行聚类，其时间复杂度为 $O(n\cdot\log n)$，n 为网络节点数. 但是原始的快速算法是针对无权网络的，为了使其可以处理边权为实数的网络，我们对其做了必要的修改. 修改后的快速算法 MFA 描述如下.

算法 1. 修改后的快速算法 MFA.

输入：弱连通网络

输出：Q 和聚类结果

1. 假定 SOS^2 中每个节点都是一个社区，并根据式(10)计算该划分的模块度 Q；
2. WHILE 社区总数大于 1 {
3. 根据式(11)依次计算有边相连社区合并后的 ΔQ，并取使 ΔQ 最大的两个社区合并；
4. 更新 we_{ij}、wa_{ij} 和 Q；
5. }
6. 选择使 Q 最大时的网络划分，输出相关结果；
7. RETURN.

社区发现算法往往是针对弱连通图进行的，MFA 以 SOS^2 的弱连通子图（Weakly Connected Graph，WCG）作为输入，输出模块度 Q 和聚类结果.

MFA 针对的是已经收集到的 SOS 的分类，那么对于后续新加入的 SOS 我们该如何确定其类别呢？LFH 将分两种情况实现新增服务分类，即：(1) 如果这个 SOS 与现有 SOS 有联系（共用部分服务），即在 SOS^2 中代表这两个应用的节点间有边相连，我们将该 SOS 分类到与其相连的那些 SOS 中产生最大 ΔQ 的那个 SOS 所在的类别中. 这种方法与我们以前用社区发现算法实现 OO 软件类重构的方法类似，详见文献[30]. (2) 如果这个新增 SOS 与现有 SOS 没有任何联系（没有共用任何 API 服务），我们先将该 SOS 归为未分类. 随着更多的 SOS 通过(1)、(2)的方式不断加入，先前那些未分类的 SOS（包括不在 WCG 中的 SOS 和新加入但是未分类的 SOS）这时可能与其它已分类 SOS 有联系，这时我们可以对这些未分类的 SOS 按照(1)的方法再进行分类.

如图 3 所示，每个黑色节点代表一个 SOS，边上的文字是其名称，虚线圈起来的 SOS 属于一个类别. 在 LFH 方法中，对每个新增的 SOS（记为 SOS_e）分类时将完成以下步骤：

1. 判断节点 SOS_e 与哪些节点有直接的联系：节点 SOS_e 与类别 Cx 内的 SOS_c 和类别 Cy 内的 SOS_d 有直接联系，而与类别 Cu 中的 SOS_f 没有直接的关系（肯定不会移动至没有直接关系的类别 Cu）.

2. LFH 方法会试探性地将节点 SOS_e 移至有直接关系的其它类，即：从类别 Cz 分别移至类别 Cx 和类别 Cy，然

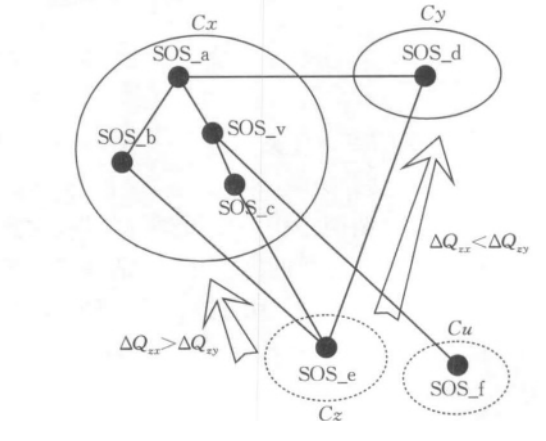


图 3 新增 SOS 分类示意图

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

后计算在这两次移动过程中产生的 ΔQ，即 ΔQ_{zx} 和 ΔQ_{zy}. 如果 $\Delta Q<0$，则令 $\Delta Q=0$.

3. 比较 ΔQ_{zx} 和 ΔQ_{zy} 的大小：如果 $\Delta Q_{zx}>\Delta Q_{zy}$，则将节点 SOS_e 从类别 Cz 移至类别 Cx；如果 $\Delta Q_{zx}<\Delta Q_{zy}$，则将节点 SOS_e 从类别 Cz 移至类别 Cy；如果 $\Delta Q_{zx}=\Delta Q_{zy}$，则将节点 SOS_e 随机移至类别 Cx 或类别 Cy 之一，即若 $rand(0,1)>1/2$，则将节点 SOS_e 移至类别 Cx，否则将节点 SOS_e 移至类别 Cy.

3.3.2　服务的自动分类

我们认为，服务的类别并不是先验的，而是由用户的使用及其应用情境所决定. 也就是说，服务的类别由其被使用的 SOS 的类别来分类. 在具体实施服务分类前，我们首先定义服务相对于社区(类别)的权值. 第 j 个服务 s_j 相对于 SOS 的第 k 个类别 C_k 的权值 W_j^k 定义为

$$W_j^k=\sum_{sos_i\in C_k}\psi_{ij} \tag{12}$$

公式中的标识符与其第一次出现时的意思相同.

LFH 方法根据算法 2 来对 API 服务进行分类，其时间复杂度为 $O(N_s^2)$. 算法 2 描述如下.

算法 2.　服务分类算法 SCA.

输入：服务集合 N_s

输出：各服务对应的类别名

注：$cNum$ 对应 SOS 分类数；$W[k]$ 用于存储服务相对于社区的权值；服务类别数组 $sClu[]$.

```
1. WHILE Ns 中服务节点数大于 1 {
2.    取 Ns 中一个服务 sj，并将其从 Ns 删除；
3.    for k=0 to cNum {
4.       计算 sj 相对类别 k 的 Wjk，存入数组 W[k]；
5.    }
6.    取 max(W[k]) 对应的下标 k(若 W[k] 中有多个值取到最大值，则随机选择其中一个的下标 k)；
7.    令 sClu[j]=k；/* 将 sj 归入获得最大 Wjk 的那个类 k 中 */
8. }
9. 输出 sClu[] 中元素；
10. RETURN.
```

3.4　服务推荐

3.4.1　SOS2S 结构特征分析

现有的 SOS 包含了 SOS 对服务的使用模式，分析这些使用模式可以为我们推荐服务和构建 SOS 提供指导. 因此，本小节我们首先分析一下 SOS2S 的一些基本结构特征. 为了后面工作的需要，我们使用二部图研究中比较常用的度数中心度(Degree Centrality，DC)指标来分析 SOS2S 的结构特征.

定义 4(度数中心度[31]).　在二部图中，一个点的度数中心度是该节点所隶属的事件数，一个事件的度数中心度是该事件所拥有的行动者数.

本文中，服务的度数中心度描述了该服务参与的 SOS 的数量. SOS 的度数中心度描述了 SOS 使用的服务的数量. 这两个度数中心度反映了 SOS 和服务的使用模式，即 SOS 使用几个服务，服务被几个 SOS 使用. 这里我们主要使用 SOS 的度数中心度为推荐服务和构建 SOS 提供指导.

在得到 SOS 的度数中心度后，我们将分析这些数值的分布，从而确定大部分 SOS 使用服务数的一个较小范围 $[A,B]$. 譬如如果有 100 个 SOS，这 100 个 SOS 使用的服务数从 1 个到 30 个不等，但是若 90%(这个数字可以由用户设定，建议大于 80%)以上的 SOS 使用的服务数都小于 10，那么我们就设置 $[A,B]$ 为 $[1,10]$. $[A,B]$ 实际上是构成 SOS 服务数的一个范围，我们新开发的 SOS 所使用的服务数也在很大概率上是属于这个范围.

3.4.2　服务推荐算法

如前所述，SOS 包含了服务的使用模式，从 SOS2S 得到的 S^2 自然也包含了服务之间的使用模式. S^2 中任意长度的路径实际上就是一个潜在的 SOS. 因此，要构建新的 SOS，只需要依据一定的规则对 S^2 进行遍历即可. 同时，我们将度数中心度最大的几个服务称为平台服务，它们成为很多 SOS 的构成单元.

通过对 SOS2S 作投影操作可以得到 SOS2S 相应的 S^2. 我们将基于 S^2 来为构建 SOS 的用户推荐服务. 首先我们将 SOS 开发中，服务的使用场景分为以下 3 种类型：(1) 用户还未选择任何服务；(2) 用户已选了一个服务；(3) 用户已选了 $n(n>2)$ 个服务. 针对每一种应用场景我们都提出了相应的推荐方法. 服务的推荐算法 SRA 见算法 3，其时间复杂度为 $O(n\cdot\log n)$，n 为网络节点数. 由于网络的规模比较大，节点间的连接可能会比较稠密，为了提高推荐算法的效率，我们在算法中提供了一个边权过滤值 w_{th} ($w_{th}\in[0,1]$)，用于将那些边权值很小的边过滤掉. 同时，从 3.4.1 小节的分析我们可以知道，SOS 一般有 B 个以内的服务构成，所以在推荐算法中，针对应用场景 3，我们限定推荐的服务路径长度 l 在 B 个以内，即 $l\leq B$. 同时，SOS 是个大粒度的服务，SOS 的推荐可以由构成它的服务的推荐来实现，即将构成该 SOS 的所有服务作为推荐算法的输入. 服务推荐算法 SRA 描述如下.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

算法 3. 服务推荐算法 SRA.

输入：S^2、用户已选择的服务集合 S_{sel}、待推荐的服务个数 k、边权过滤值 w_{th}、路径长度 l

输出：推荐的服务集合

1. IF S_{sel} is NULL THEN //场景(1)
2. 推荐平台服务；
3. ELSE IF $|S_{sel}|=1$ THEN //场景(2)
4. IF $DC(k)\geq k$ THEN
5. 在 S^2 中选择与服务 k 相连的节点中，节点间边权 top-k 的服务进行推荐；
6. ELSE IF $1<DC(k)<k$ THEN
7. 在 S^2 中选择与服务 k 相连的节点中，节点间边权 top-$DC(k)$ 个服务进行推荐；
8. ELSE //场景(3)
9. For S_{sel} 中任一节点组合 Ns 和 Ne
10. 求 S^2 中以 Ns 为起点，Ne 为终点，包含 S_{sel} 中其它节点，经过的边权都不小于 w_{th} 且长度为 l，$l-1$，$l-2$，…，$DC(k)$ 的所有路径，并将其放入集合 S_{route}；
11. 按照路径长度对 S_{route} 中的路径归类，输出每一类中边权和 top-k 的路径进行推荐；
12. RETURN.

4 实例分析

为了说明 LFH 方法的具体实施过程，同时验证其有效性，本文将结合 ProgrammableWeb 上 mashup 应用和 API 服务的真实数据，进行实证分析.

4.1 数据来源

ProgrammableWeb 是著名的 mashup 应用和开放 API 服务目录，罗列了 5000 多个 mashup 应用和 3000 多个 API 服务，并提供了 mashup 和 API 的一些注册元信息，包括它们的名称、URL、提供者、标签等. API 是服务，mashup 是由服务组合得到的 SOS，也可以看成大粒度服务，它包含服务组合的历史信息，符合本文工作的要求. 因此，本文以 ProgrammableWeb 上的 mashup、API 服务的真实数据为例展开研究.

图 4 所示的是 ProgrammableWeb 上名称为 Ski Bonk① 的 mashup 应用的一些注册元信息，包括 mashup 应用的名称、描述信息、使用的 API 服务和标签等. 我们使用自己开发的网爬工具将 ProgrammableWeb 上从 2005 年(建站时)到 2011 年 1 月 12 日(本文工作开展时)所有 mashup 应用的名称、描述信息、API 服务和标签信息(图 4 中方框部分)爬了下来，存储在本地数据库中. 这些数据是后续工作的基础.

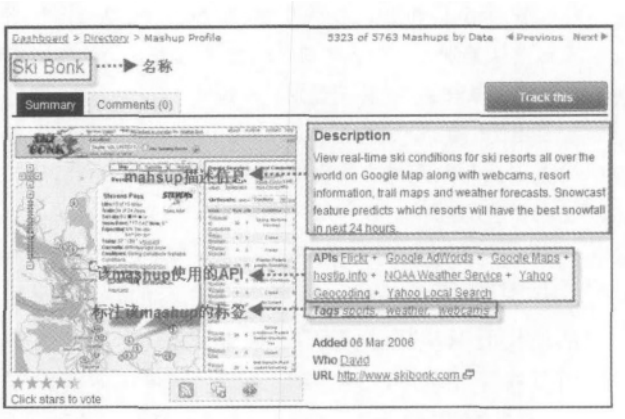


图 4 Ski Bonk 的元信息

ProgrammableWeb 上的数据都是由用户提交的，存在一定的随意性. 在数据收集的过程中，我们发现这些数据存在一些错误：(1) 有些 mashup 存在重复注册现象. 一些 mashup 虽然它们名称不一样，但是它们其它的信息都一样. 这些 mashup 在我们的数据集中只保存一份. (2) 有些 mashup 仅提供了名称，但是其它我们所需的注册信息缺失了. 这些 mashup 将不作为我们的实验数据. 同时，用于标识 mashup 的标签也存在不一致，同一种含义的标签有多种不同的表现形式，如“api”、“Api”和“APIs”都表示 API，但是形式不同，有些标签甚至拼写错误. LFH 方法使用文献[32]中的 Suffix Stripping 算法对标签进行预处理，将同意异形的词都转化成同一词性，尽量消除存在的不一致性. 最终，我们的数据集包含 5115 个 mashup 应用、750 个 API 服务和 1489 个标签，数据集可以从文献[33]下载.

4.2 mashup 和 API 服务的自动分类

4.2.1 mashup 的自动分类

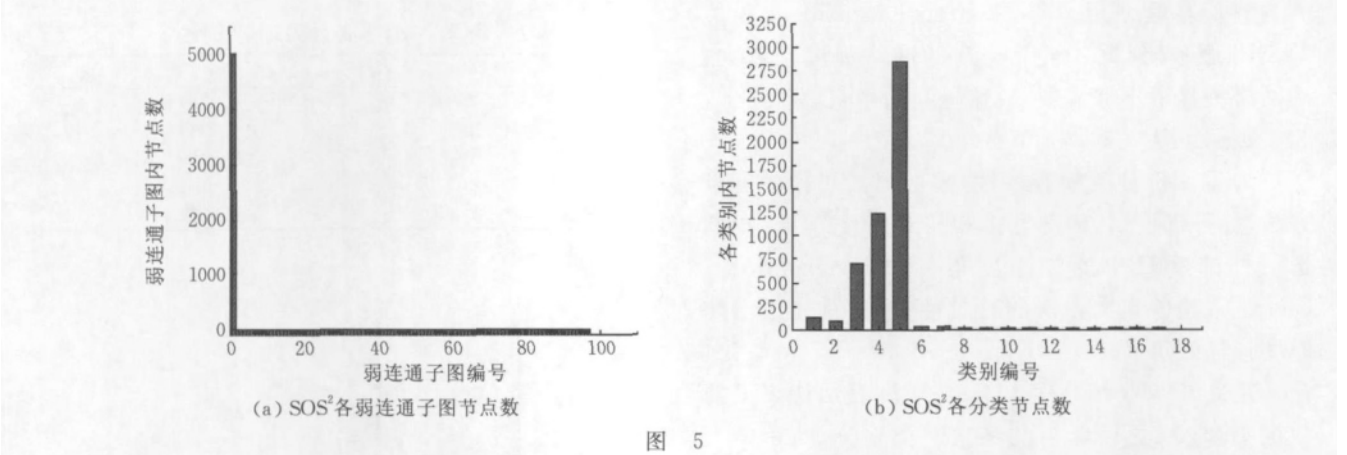
通过解析所获得的数据，我们构建了所有 mashup 应用和 API 服务之间的 SOS2S(mashup 是 SOS、API 是服务). 通过对 SOS2S 作投影操作，我们可以得到 SOS2S 相应的 SOS^2，该 SOS^2 拥有 5115 个节点和 2 660 793 条边. 但是这个 SOS^2 并非一个弱连通图，它由 97 个弱连通子图构成. 图 5 显示了各个弱连通子图包含的节点数，其中最大弱连通子图(Maximum Weakly Connected Graph，MWCG)有 4973 个节点，最小的子图节点数仅为 1(孤立点). MWCG 的节点数占了 SOS^2 总节点数的

① Ski Bonk 在 programmableWeb 上的网址. http://www.programmableweb.com/mashup/ski-bonk

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

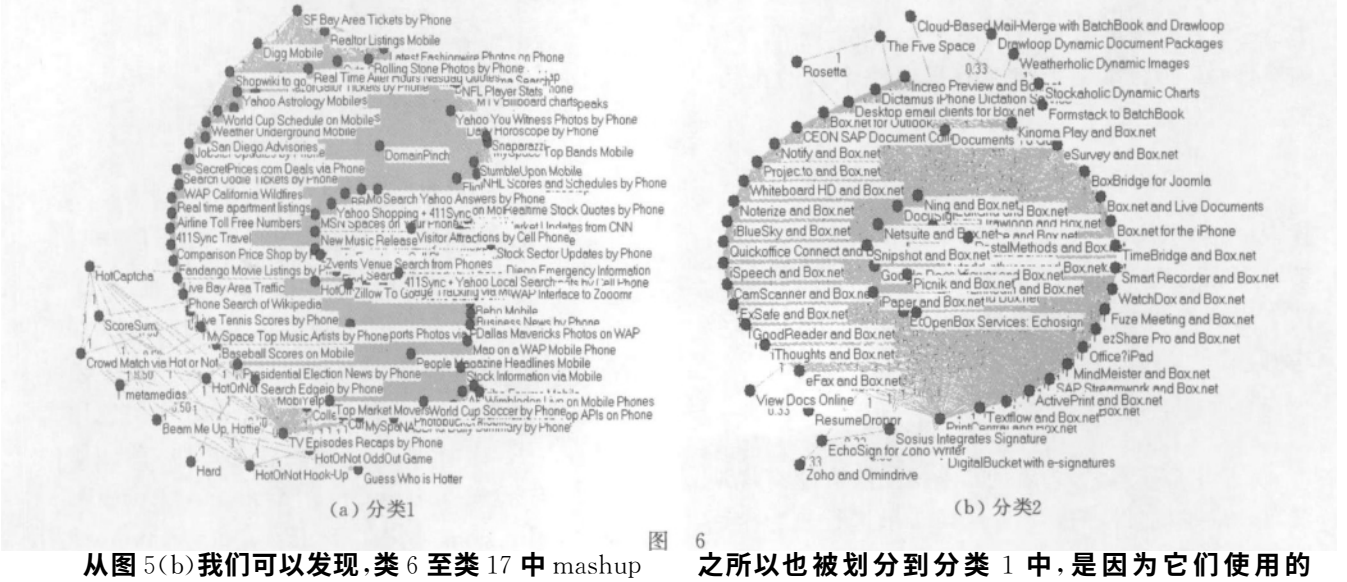
97.2%，而其余的弱连通图的节点数占了不到4%，统计上忽略它们不会对结果产生太大的影响. 因此，本文使用 SOS^2 的 MWCG 作为分析对象.

我们将 MWCG 作为 MFA 算法的输入，聚类后得到 17 个分类（社区），各分类的规模如图 5(b)所示，部分分类结果及其网络图见图 6 所示. 那些不在 MWCG 中的 mashup 应用，我们将其标识为未分类.



(a) SOS^2 各弱连通子图节点数　　(b) SOS^2 各分类节点数

图 5



(a) 分类1　　(b) 分类2

图 6

从图 5(b)我们可以发现，类 6 至类 17 中 mashup 应用节点数都在 20 以内（合计 58 个节点），相比网络规模总节点数 4973 而言很小，统计上我们可以将这些类忽略. 因此，我们这里主要考虑前 5 个分类. 为了验证分类结果是否有意义，我们手工对 5 个分类中共 4915 个 mashup 应用进行分析. 通过分析每个 mashup 应用的描述（description）信息，从描述信息判断其功能及所述的领域，最终将这 4915 个 mashup 进行手工分类.

在分类 1 中（如图 6(a)）除 Crowd Match via Hot or Not、Guess Who is Hotter、Hard、HotCaptcha、HotOrNot Hook-Up、HotOrNot Live Gadget、HotOrNot OddOut Game、Jesus Speaks、metamedias、ScoreSum、Snaparazzi 这 11 个 mashup 应用以外，其余都是跟手机移动通信密切相关的. 而这 11 个应用之所以也被划分到分类 1 中，是因为它们使用的 API 服务也被分类 1 中其它的 mashup 应用频繁使用. 在分类 2 中（如图 6(b)）除 The Five Space、Rosetta、Weatherholic Dynamic Images、Stockaholic Dynamic Chart 这 4 个 mashup 应用以外都是跟在线办公（在线文档处理）密切相关的. 而这 4 个应用被划分到分类 2 中，同样也是因为它们使用的 API 服务被分类 2 中其它的 mashup 应用频繁使用. 分类 3 中主要是一些与出行相关的 mashup 应用，包括道路、交通、住宿、餐饮、天气等. 当然，这些 mashup 应用中好多也用到了地图方面的 API 服务. 在分类 4 中主要是一些与地图相关的 mashup 应用，且大多数应用都使用了 Google Maps 这个 API 服务，甚至好多应用仅由这一个 API 服务构成，如 Openmaps、RateMyStreet 等. 在分类 5 中主要是一些与社交网

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

络(Flicker、Twitter、Facebook、Youtube、Delicious、wiki、amazon eBay)密切相关的 mashup 应用.此外,也包括了一些出行、桌面应用、手机软件、政治等相关的应用.这些应用之所以也分到了这个类中,是因为它们都是使用了与 Twitter、Facebook 等相关的 API 服务.分类 3 至分类 5 的网络规模比较大,用网络图显示不够清晰,故这里我们将它们省略了,相关数据可从文献[33]下载.

为了评价分类的有效性,人们提出了很多度量方法,如外部度量、内部度量和相对度量等[34].我们选用外部度量中的纯净度指标(Cluster Purity,CP)[35]来检验本文方法的有效性.下面首先给出纯净度指标的定义.

定义 5(聚类纯净度[35]). 令 D 表示待聚类的数据集合,C 表示 D 上的某次聚类划分,$C_k \in C$ 为该聚类结果中的一个簇,S 表示 D 上的标准聚类结果,$s \in S$ 为该聚类结果中的一个标准簇,则 C_k 类的纯净度 $CP(C_k)$ 定义为该类数据占整个簇的比例,即

$$CP(C_k)=\frac{1}{|C_k|}\max_s(|C_k^s|) \tag{13}$$

其中 $|\cdot|$ 表示集合 $\cdot$ 中的元素个数.本文中标准聚类结果 S 是通过我们手工分类得到的.在此基础上,整个划分 C 的纯净度 $CP(C)$ 是所有簇的纯净度的均值,即

$$CP(C)=\sum_{k\in C}\frac{|C_k|}{|D|}CP(C_k) \tag{14}$$

CP 反应了聚类结果的准确性,其值越高说明聚类效果越好,纯净度能对聚类结果进行较为全面合理的评价.本文方法所得各个簇的纯净度如表 1 所示,整个聚类的纯净度为 86.8%.

表 1 mashup 各分类纯净度

类别	纯净度/%
第 1 类	90.3
第 2 类	93.9
第 3 类	82.5
第 4 类	82.7
第 5 类	89.3

4.2.2 API 服务的自动分类

在得到了 mashup 应用的分类后,我们按照 SCA 算法实现 API 服务的分类.各类别中的 API 服务数量如表 2 和图 7 所示.表 2 中显示的是 5 个主要分类的结果(对应 4.2.1 节 mashup 应用的 5 个主要分类),图 7 显示的是所有分类的结果.其中第 0 类表示这些 API 没有被分类.因为我们的分类是针对 mahsup 的分类结果来对 API 分类的.mashup 分类的时候针对的是 MWCG 进行的,忽略了那些不在这个 MWCG 中的节点.所以这些 mashup 应用使用的部分 API 就无法实现分类.

表 2 API 服务分类结果统计

类别	API 服务数
第 1 类	7
第 2 类	14
第 3 类	102
第 4 类	1
第 5 类	511

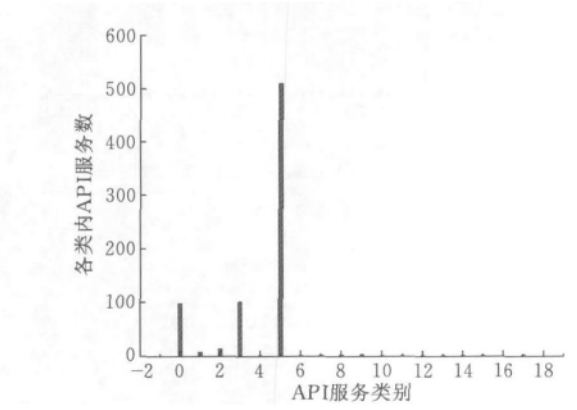


图 7 API 服务分类结果统计(全部类别)

为了检验 LFH 方法对新增 mashup 应用分类的有效性,我们又将 ProgrammableWeb 上 2011 年 2 月 1 日至 2011 年 2 月 28 日(本文工作结束时)新增的 85 个 mashup 应用的元信息爬了下来.在这 85 个 mashup 应用中,共有 30 个 mashup 应用使用了 2011 年 1 月 12 日(4.1 节数据收集之时)后注册的新 API 服务,如 mashup 应用 Compare Prices Discount-Malin.com 使用了 Kelkoo,player4you 使用了 Google Font,mapTheGraph!使用了 Facebook Graph 等.这些 mashup 应用在与其它 mashup 应用计算相似度的时候,这些新 API 服务没有参与计算,因此会对相似度值产生影响.所以使用了这些新 API 服务的 mashup 应用并不作为我们的检验数据.因此我们的检验数据集由剩下的 55 个 mashup 应用构成.

我们按照 3.3.1 节介绍的方法对这新增的 55 个服务进行自动分类,然后将自动分类的结果与手工分类的结果进行对比,我们发现 55 个 mashup 应用中,其中 48 个分到了正确的类别中,只有 7 个 mashup 应用的分类错误.可见 LFH 方法对新增 mashup 应用的分类具有很好的效果.

4.3 API 服务推荐

通过解析所获得的数据,我们构建了所有

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

mashup 应用和 API 服务之间的 SOS2S(mashup 是 SOS、API 是服务). 如图 8 所示，其中圆形节点对应 mashup 应用(共 5115 个)，方形节点对应 API 服务(共 750 个)，两类节点间的边代表 mashup 应用使用 API 服务的关系.

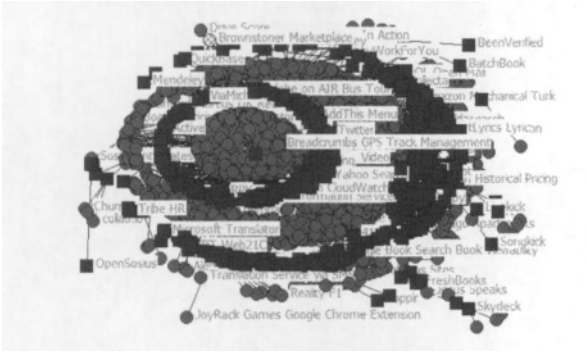


图 8　SOS2S 图示

我们使用度数中心度指标 *DC* 来分析该 SOS2S 的结构特征. 图 9 显示了具有某一度数中心度的 mashup 应用和 API 服务的数量. 可以发现，大部分 mashup 应用具有较小的度数中心度(80.1% 的 mashup 应用的度数中心度仅为 1 和 2)，只有 1.1% 的 mashup 应用度数中心度大于 10，平均每个 mashup 应用度数中心度仅为 13.8. 这说明大部分 mashup 应用仅由少量的 API 服务组合而成. 与此同时，大部分的 API 服务具有较小的度数中心度(50.7% 的 API 服务度数中心度仅为 1 和 2)，只有 17.9% 的 API 服务度数中心度大于 10，平均每个 API 服务的度数中心度为 9.5. 这说明大部分 API 服务仅参与了少量的 mashup 应用. 度数中心度最大的 10 个 mashup 应用和 API 服务(平台 API 服务)如表 3 所示.

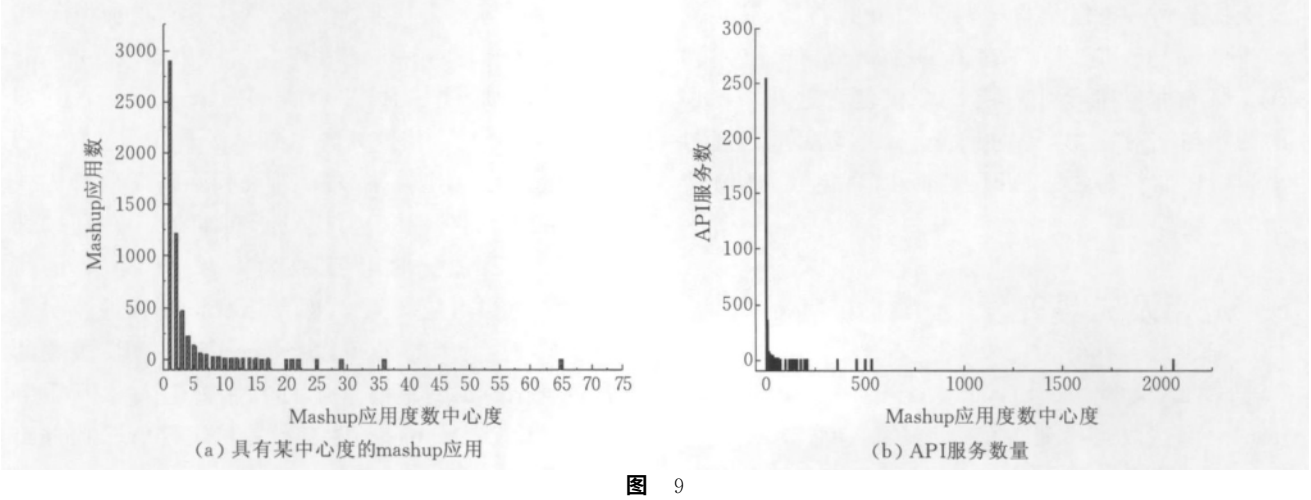


图　9

表 3　度数中心度 top-10 的 mashup 和 API 服务

类型	名称	度数中心度
mashup 应用	Tagbulb	65
mashup 应用	Headup	36
mashup 应用	Pixelpipe	29
mashup 应用	Sociotoco Search	25
mashup 应用	Gawkk.com	25
mashup 应用	G4ng	22
mashup 应用	ConnectorLocal	22
mashup 应用	Pageflakes	21
mashup 应用	Maps Shown To Me	21
mashup 应用	KwiClick	20
API 服务	Google Maps	2060
API 服务	Flickr	533
API 服务	YouTube	499
API 服务	Twitter	453
API 服务	Amazon eCommerce	358
API 服务	Facebook	205
API 服务	eBay	194
API 服务	Microsoft Virtual Earth	173
API 服务	Last.fm	158
API 服务	Google Search	155

通过对 SOS2S 作投影操作，可以得到 SOS2S 相应的 S^2(如图 10). S^2 包含 750 个节点，9415 条边，由 97 个弱连通子图构成，其中最大的弱连通子图有 651 个节点，最小的子图节点数仅为 1(孤立点). 节点边上的标签是该节点所属分类的一个标识符. 为了使图显示的清晰，我们忽略了每个节点对应 API 服务的名称.

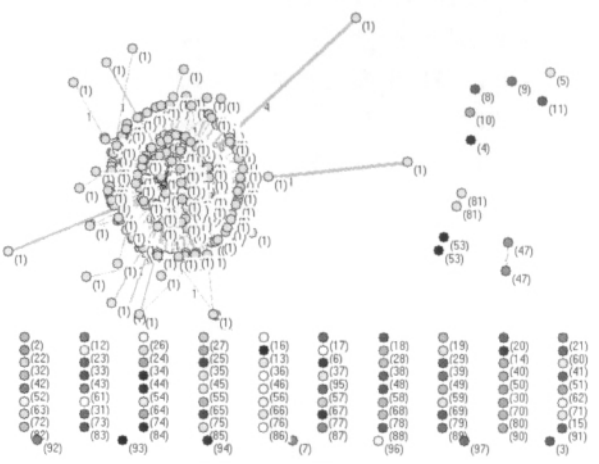


图 10　S^2 图示

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

为了检验本文构建的 S^2 是否有意义以及 API 服务推荐方法是否有效，我们仍然使用 4.2.2 节收集的 55 个 mashup 应用进行验证. 在这 55 个 mashup 应用中，mashup 应用使用的 API 服务的个数从 1（如 Color Pal）到 20（如 The Music Feed）不等，但 98.2% 的 mashup 应用其使用的 API 服务个数仍在 10 个以内. SRA 按 3 种使用场景进行推荐，使用场景不同，推荐过程存在差异. 这里我们以每个 mashup 应用使用的 API 服务集合作为 SRA 的输入，看 S^2 中是否存在一条仅包含这几个服务的路径，以此作为检验 S^2 和算法有效性的标准.

我们发现，在这 55 个 mashup 应用中，仅 Rhythm and Bluesmashup 这个 mashup 应用对应的 MTV，Google Custom Search 两个 API 服务在 S^2 中找不到一条仅包含这两个服务的路径，其余的 54 个 mashup 应用中，它们每个 mashup 使用的 API 服务集合在 S^2 中都有一个仅包含这几个 API 的路径与之对应，这说明推荐 SRA 确实可以通过 S^2 为用户推荐有效的 API 服务，以简化 mashup 应用的构建.

5 应用及工具开发

S2R2(Software Service Registry and Repository)[①]是武汉大学研发的一个语义互操作性软件服务注册管理平台. 该平台通过研究互联网上软件服务的基础理论与方法、技术标准，开发实现软件服务资源管理与服务的基础设施平台及其关键技术，提供一个统一的软件服务注册管理平台，有效管理各种已注册软件服务，方便用户找到所需服务. 以这一平台为基础，建立软件服务开发的基础设施，改善软件服务质量，同时，推动面向服务的软件开发新技术的应用，实现软件服务资源的交流与共享.

目前该平台主要由服务（Web 服务、mashup 应用、API 服务等）注册子系统、服务管理子系统、本体注册管理子系统和语义查询子系统 4 个部分组成（如图 11 所示）. S2R2 系统的详细描述请见文献[36].

我们已经将 LFH 方法应用于 mashup 应用注册和 API 服务查询 2 个模块中，提供对 mashup 应用和 API 服务分类以及 API 推荐的支持. 当用户通过 S2R2 平台注册一个 mashup 应用时，在输入该 mashup 应用使用的 API 后，后台程序会通过算法 2 为其推荐可能的类别. 图 12(a)显示的是注册 mashup 应用 Localley[②] 时，在输入 Facebook、Google Maps

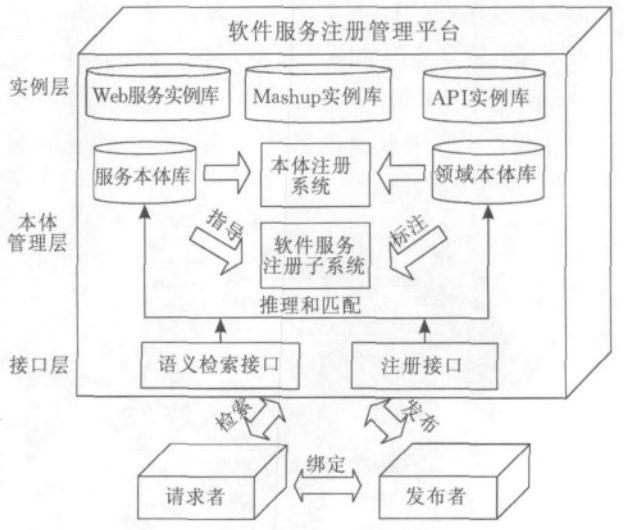


图 11 S2R2 结构图

和 Twitter 后，为其推荐“社交网络”类别的例子. 当用户在某个 mashup 应用开发平台中开发 mashup 应用时，当通过 S2R2 平台查询一个或多个 API 服务时，S2R2 也会按照算法 3 推荐若干可与已有 API 服务组合的 API 服务，并以列表和网络图两种方式显示给用户. 图 12(b)对应场景 1，即在用户还未选择任何 API 服务时，为其推荐平台 API 服务的情形. 图 12(c)对应场景 2，即在用户已输入一个 API 服务 23 时，为其推荐可以组合的其它 API 服务的情形. 图 12(d)对应场景 3，即在用户输入 API 服务 23 和 Google Maps 时，为其推荐可组合 API 服务路径的情形.

LFH 方法提供了一种利用历史信息从结构角度挖掘知识的方法，虽然本文以 API 服务和 mashup 应用为载体来进行说明，但是只要有 SOS，可以获得服务和软件之间的这种构成关系，我们就可以用 LFH 方法来对服务和软件（大粒度服务）进行分类，并进行服务的推荐.

本文提出的服务分类方法可以用于服务注册管理系统中，为服务提供分类信息，提高服务发现、检索以及服务资源管理的效率. 与此同时，目前服务组合中服务的可组合关系往往是根据服务间的输入输出来进行判断的，本文提供了一种从结构角度挖掘服务组合关系的方法，为服务组合开辟了新的思路.

SOS2S 可以推广至一般的 SOS 和服务之间关系的网络，从而可以让我们获得一些知识，如软件是如何使用服务的、服务是如何在不同的软件中使用

① s2r2 官方网站. http://www.s2r2.org
② Localley 在 programmableWeb 上的网址. http://www.programmableweb.com/mashup/localley

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

图 12　功能示例图

的、多个软件如何使用相同的服务等. 这些知识包含了一些服务使用的正确经验，可以作为推荐系统的一部分，从而指导面向服务的软件的开发.

SOS^2 可以推广至 SOS 之间的网络，从而为我们提供了一种基于关系的软件查找方式. 我们借助 SOS 构成的网络来查找功能相似的软件，从而为用户提供多种选择可能.

同样的，S^2 也可以推广到一般的服务网络，可以帮助我们挖掘服务之间可能的组合关系，并可以通过图的遍历推荐潜在的服务组合方案，也就是潜在的 SOS，从而为 SOS 的开发提供支持.

因此，本文提出的 LFH 方法对于服务的管理、查找、推荐和自动组合都有一定的意义，可以为实现 SOS 开发提供部分支持.

6　结论与展望

本文利用服务组合历史信息，用软件网络抽象面向服务软件，提出了从拓扑结构角度实现服务分类和推荐的 LFH 方法：利用软件网络上的复杂网络社区发现实现服务的自动分类，这对于服务组织管理具有重要意义；利用软件网络刻画服务之间的可组合关系，并基于该网络实现服务的推荐，为服务组合提供了新思路. 最后，本文以 ProgrammableWeb 上 mashup 应用和 API 服务的真实数据为载体进行实例研究，在说明 LFH 方法具体实施过程的同时验证了其可行性和有效性. 本文方法在服务的管理、查询、组合等方面都具有较好的应用前景.

下一步我们将考虑如下研究工作：(1) 本文构建的 S^2 中有很多边其实是无效的，我们将利用服务的其它信息(如输入、输出等)对服务进行二次匹配，尽量减少无效边的数量，提高服务推荐的准确度；(2) 本文构建的 SOS^2 尽管考虑了标签的作用，但是语义信息的利用仍不足，我们拟利用服务的描述文档，将文档之间的相似性也加入 mashup 应用之间相似度的计算中；(3) 本文最后以 ProgrammableWeb 上 mashup 应用和 API 服务的真实数据验证 LFH 方法的有效性，我们拟用其它类型的服务和基于服务的应用来验证本文方法的有效性.

参　考　文　献

[1] Papazoglou M P. Service-oriented computing: Concepts, characteristics and directions. Information System Journal, 2003, (2): 3-12

[2] Cardoso J. Quality of service and semantic composition workflows [Ph. D. dissertation]. University of Georgia, GA, 2002

[3] Brook Jr F P. Three great challenges for half-century-old computer science. Journal of the ACM, 2003, 50(1): 25-26

[4] Valverde S, Cancho R F, Sole R. Scale free networks from optimal design. Europhysics Letter, 2002, 60: 512-517

[5] Myers C R. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. Physical review E, 2003, 68(4): 046116

[6] Valverde S, Sole R V. Hierarchical small worlds in software

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

architecture. SanteFe Insitute: Working Paper, SFI/03-07-044, 2003

[7] Moura A P S, Lai Y C, Motter A E. Signatures of small-world and scale-free properties in large computer programs. Physical Review E, 2003, 68: 017102

[8] LaBelle N, Wallingford E. Inter-package dependency networks in open-source software. ArXiv: Cs. Se/0411096, 2004

[9] Valverde S, SoléR V. Universal properties of bipartite software graphs//Proceedings of the 9th IEEE International Conference on Engineering of Complex Computer Systems. Florence, Italy. 2004

[10] Han Ming-Chang, Li De-Yi, Liu Chang-Yu, Li Hua. Networked characteristics in software and its contribution to software quality. Computer Engineering and Applications, 2006, 42(20): 29-31(in Chinese)
(**韩明畅, 李德毅, 刘常昱, 李华. 软件中的网络化特征及其对软件质量的贡献. 计算机工程与应用**, 2006, 42(20): 29-31)

[11] Liu J, He K Q, Ma Y T, Peng R. Scale free in software metrics//Proceedings of the 28th Annual International Computer Software and Application Conference. Chicago, USA, 2004: 229-235

[12] Pan W F, Li B, Ma Y T, Qin Y Y, Zhou X Y. Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks. Journal of Computer Science and Technology, 2010, 25(6): 1202-1213

[13] Papazoglou M, Georgakopoulos D. Service oriented computing. Communications of the ACM, 2003, 46(10): 25-28

[14] Paolucci M, Kawamura T, Payne T R, Sycara K. Semantic matching of Web services capabilities//Proceedings of the 1st International Semantic Web Conference. Sardinia, Italy, 2002: 333-347

[15] DAML service home. http://www. daml. org/ser vices/owl-s/, May 5, 2011

[16] Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal, 1970, 49(1): 291-307

[17] Newman M E J. Finding community structure in networks using the eigenvectors of Matrices. Physical Review E, 2006, 74(3): 036104

[18] Angelini L, Boccaletti S, Marinazzo D, Pellicoro M, Stramaglia S. Fast identification of network modules by optimization of ratio association. ArXiv: cond-mat/0610182v2, 2006

[19] Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D. Defining and identifying communities in networks//Proceedings of the National Academy of Science of the United States of America, 2004, 101(9): 2658-2663

[20] Newman M E J, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004, 69(2): 026113

[21] Alves N A. Unveiling community structures in weighted networks. Physical Review E, 2007, 76(3): 036101

[22] Farkas I, Abel D, Palla G et al. Weighted network modules. New Journal of Physics, 2007, 9(6): 180-199

[23] Zhang S H, Wang R S, Zhang X S. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Physica A, 2007, 374(1): 483-490

[24] Zhang S H, Wang R S, Zhang X S. Uncovering fuzzy community structure in complex networks. Physical Review E, 2007, 76(4): 046103

[25] Clauset A, Newman M E J, Moore C. Finding community structure in very large networks. Physical Review E, 2004, 70(6): 066111

[26] Duch J, Arenas A. Community detection in complex networks using extremal optimization. Physical Review E, 2005, 72(2): 027104

[27] Clauset A. Finding local community structure in networks. Physical Review E, 2005, 72(2): 026132

[28] Bagrow J P. Evaluating local community methods in networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(5): P05001

[29] Newman M E J. Fast algorithm for detecting community structure in networks. Physical Review, 2004, 69(6): 066133

[30] Pan W F, Li B, Ma Y T, Liu J, Qin Y Y. Class structure refactoring of object-oriented software using community detection in dependency networks. Frontiers of Computer Science in China, 2009, 3(3): 396-404

[31] Wikipedia. http://en. wikipedia. org/wiki/Centrali ty, May 5, 2011

[32] Porter M F. An algorithm for suffix stripping. Program, 1980, 14(3): 130-137

[33] Data. http://blog. sina. com. cn/breezepan, May 5, 2011

[34] Christopher D, Prabhakar R, Hinrich S. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008

[35] Zhao Y, Karypis G. Criterion functions for document clustering: experiments and analysis. Department of Computer Science, University of Minnesota, Technical Report CS0140, 2001

[36] Zeng C, He K Q, Li B, Wang C, He Y F. Toward multi-ontology based interoperability in web service registry. Journal of Computational Information Systems, 2009, 5(6): 1669-1677

PAN Wei-Feng, born in 1982, Ph.D., lecturer. His current research interests include software engineering, complex networks, and intelligent computation.

LI Bing, born in 1969, professor, Ph. D. supervisor. His current research interests include software engineering, service computing, and cloud computing.

SHAO Bo, born in 1988, M. S. candidate. His current research interest is service computing.

HE Peng, born in 1988, M. S. candidate. His current research interests include software engineering and complex networks.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

Background

The internal structures of a software system will have great importance on its functionality, performance, reliability, and other quality attributes. So, to conduct deep investigation into software systems, the information embedded in their structures should be effectively explored. And it is deemed as one of the three challenging issues for the half-century-old computer science.

Recent research achievements in complex networks introduce new mathematical foundations and software engineering methods for software complex systems. A few researchers introduced the complex network theory to software engineering domain, and used software networks to represent software structures. Based on software networks, much work has been carried out to study the software structure and its dynamics, the results of which provide useful insights into software development and maintenance.

Service oriented computing is an emerging cross-disciplinary paradigm for distributed computing. It is changing the way software applications are designed, architected, delivered and consumed. Service classification and recommendation are two important problems in service management and composition. This paper, utilizing service composition histories, investigates the method for service classification and recommendation from a topological perspective of service-oriented software (SOS). It proposes a software network model for SOS to abstract its topological structures. It uses a community detection algorithm to cluster software networks and realizes service classification. It uses software networks to describe compositional strength between services and proposes an algorithm for service recommendation. This work has been supported by the National Basic Research Program (973 project) of China under grant No. 2007CB310801, the National Natural Science Foundation of China under grant Nos. 60873083 and 61003073, the China Postdoctoral Science Foundation under grant Nos. 20090460107 and 201003794, and the Fundamental Research Funds for the Central Universities under grant Nos. 114013 and 3101032.

首届全国信息安全等级保护技术研讨会(ICSP'2012)

征文通知

首届全国信息安全等级保护技术研讨会拟于 2012 年 4 月在广西桂林召开. 本次会议由公安部网络安全保卫局等部门指导,中科院信息安全共性技术国家工程研究中心主办,公安部第三研究所、中国信息安全测评中心和国家信息技术安全研究中心协办. 会议将出版大会论文集,评审优秀论文并推荐至《中国科技大学学报》、《计算机工程》、《计算机应用》、《计算机工程与应用》、《计算机应用与软件》、《中国科学院研究生院学报》、《计算机系统应用》等期刊发表.

征文范围:

等级保护标准支撑技术:包括基于《信息安全等级保护基本要求》、《信息安全等级保护安全设计技术要求》等标准,研究计算环境安全、网络边界安全、通信安全和安全管理平台等技术、标准验证环境构建技术等. 等级保护建设整改技术:包括信任体系模型与构建技术、可信计算技术、针对重要行业关键信息基础设施的专用等级保护支撑性技术和方法. 等级保护测评技术:包括标准符合性检验技术、安全基准验证技术、无损检测技术、渗透测试技术、逆向工程剖析技术、源代码安全分析技术等. 等级保护的监管技术:包括用于支撑安全监管需求的敏感数据保护技术、安全态势评估技术、安全事件关联分析技术、安全绩效评估技术等. 新技术应用环境的等级保护技术:包括云计算、物联网、三网融合、新媒体等新技术新环境下的等级保护支撑性技术,等级保护技术体系在新环境下的应用方法等. 其他等级保护相关的安全技术和方法.

投稿要求:

来稿内容应属于作者的科研成果,数据真实、可靠,未公开发表过;引用他人成果应注明出处;署名无争议;论文摘要及全文不涉及保密内容.

会议只接受以 Word 排版的电子稿件,稿件一般不超过 10 页,格式符合《软件学报》的要求.

稿件以 Email 的方式发送到会议征稿邮箱 icsp2011@nercis.ac.cn.

凡投稿文章被录用且未做特殊声明者,视为已同意授权出版.

重要日期:

论文提交截止日期:2012 年 2 月 10 日

录用通知发出日期:2012 年 3 月 10 日

会议召开日期:2012 年 4 月 7 日～8 日

联系方式:

联系人:陈恺,张玲,石雅辉　　联系电话:010-82486444,010-82486299

E-mail:icsp2011@nercis.ac.cn　　官方微博:http://weibo.com/icsp

通讯地址:北京市海淀区中关村南四街 4 号　　邮编:100190

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net