DOI:10.13718/j.cnki.xsxb.2015.01.021

多属性融合评级的 Mashup 服务推荐算法①

赵国栋[1], 宋丽亚[2], 焦小刚[1]

1. 宁夏大学 数学计算机学院,银川 750021;2. 宁夏大学 物理电气工程学院,银川 750021

摘要:传统的 Mashup 服务推荐是基于关键字的检索方法,对于所推荐 API 服务的社会和功能属性利用较少,不利于全面评价所推荐 API 的适用度,对此提出一种多属性融合评级的 Mashup 服务推荐算法. 首先,利用网爬工具收集 ProgrammableWeb 上的 Mashup 服务信息,并采用后缀剥离算法把 Mashup 服务的标签信息修改为名词形式,以此作为研究分析的数据集. 其次,融入 API 服务的社会和功能等多属性对 API 模型进行扩充,并采用多属性相似度加权融合的方式对候选 API 的适用度进行评价,以此作为 API 服务推荐的依据. 实验结果表明,多属性融合评级 Mashup 服务推荐算法具有更高的正确率和更快的运算时间,是可行有效的.

关 键 词:多属性;Mashup;标签;融合

中图分类号:TP391 **文献标志码**:A **文章编号**:1000-5471(2015)1-0121-08

Mashup 最早来源于乐曲的编写,在音乐中常把用两种以上歌曲段落组合成一首新歌的编曲方式称作是 Mashup. 从某种意义上,把众多的网络服务资源通过一定的组合方式呈现给使用者,也类似于这种编曲方式,因此这种交互式的 Web 服务提供方式也被称为 Mashup. Mashup 概念的提出和推广迎合了 Web2.0 信息资源共享、聚合和重复利用的思想,注重客户的共同参与,关注客户的良好体验,使得网页逐渐由静态的文本链接方式转化为使用诸如 RSS 和 REST 等的信息动态共享和资源的聚合,改善了客户 Web 访问端的操作感受[1].

近年来,WebAPI 数量的爆炸式增长对 Mashup 网络服务应用程序的开发提出了挑战,吸引着众多研究人员对 Mashup 服务推荐算法进行研究. 在应用层面:文献[2]提出一种 Mashup 框架,是基于 SOA 框架的扩展,具有 Mashup 服务聚合的快速性,并且允许终端使用者根据需要随意组合 Mashup 服务;文献[3]提出可独立组合创建 Mashup 情景应用程序的系统,该系统具有层级结构,下层可为上层提供所需的 Mashup 服务等. 在理论方面:文献[4]提出一种基于关键字并整合 API 社会属性的搜索方法,但是该方法没有考虑 API 的功能属性,例如 API 的接口参数,导致所推荐 API 的可用性不能得到有效保证;文献[5]则刚好相反充分考虑 API 的接口参数,并设计一种机制对于接口参数进行语义指定,但忽略了 API 社会属性的利用;文献[6]提出一种基于相似度的 API 评价模型,以实现评级更高的 API 服务替代给定 API 服务的效果.

本文借鉴文献[6]的 API 评价模型并融合 API 的社会和功能属性对 API 评价模型进行扩充,来改善 Mashup 服务的推荐效果,并采用在 ProgrammableWeb 上获取的 Mashup 服务信息组建实验数据库,对所提算法的性能进行实际数据背景下的仿真验证.

① 收稿日期:2014-06-15
基金项目:宁夏回族自治区自然科学基金项目(NZ13048).
作者简介:赵国栋(1972-),男,副教授,博士研究生,主要研究领域为软件工程,计算机教育和服务推荐.

1 Mashup服务测试数据建立

1.1 Mashup网络构建

在Mashup网络中Mashup服务、网络API服务以及服务的标签信息等存在一定联系，这种联系是构建Mashup网络的基本前提. Mashup网络由4部分组成(图1(a)[5])：服务信息的采集与处理、基于标签定义的Mashup网络、多属性融合评级的Mashup服务推荐、用户查询功能.

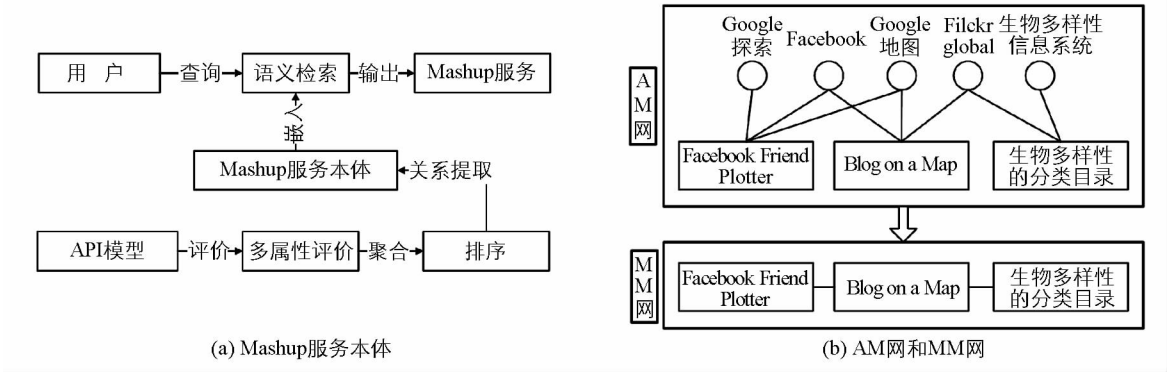

图1 Mashup服务网络

定义1(A-M网) API服务与Mashup服务相关联的二模网络(如图1(a)).

图1(a)中方框或圆圈代表的是Mashup或者API服务，直线表示的是Mashup服务调用API服务的关系. 图1(a)中可以看出一个Mashup服务是由多个API服务组合而成.

定义2(M-M网) Mashup服务与Mashup服务之间的关系网络(如图1(b)).

方框代表的是Mashup服务，直线表示的是Mashup服务之间的相互关系. M－M网可表示为：

$$\mathrm{M}-\mathrm{M}=(N,E) \tag{1}$$

其中：N为Mashup服务集，E为Mashup服务关系集.

根据上述定义，多属性融合评级的Mashup服务推荐所要解决的问题是：通过API模型构建、相似度评价及等级排序，将合适的API服务推荐给网络开发者或终端用户，并希望这种推荐方式能够足够精确，满足实际需要.

1.2 Mashup服务数据提取整理

服务信息的标签是服务提供者对服务软件从个人认识的角度进行的文字定义，由于定义者的不同导致个体间的文字定义存在不同，例如wps,WPS两个标签都是用来标记服务或软件的信息，虽然字母的大小写不同，但具有相同的含义. 对此使用文献[7]提出的后缀剥离算法对网爬下来的服务标签信息进行处理，采用名词结构进行重新标注. 后缀剥离算法如下：

英文字母中有元音和辅音之分，元音有A,E,I,O和U，再就是Y前面的字母也称之为元音. 在一个单词中一般把辅音用C表示，元音用V表示，如果单词中连续元音或辅音字母数量大于1，那么统一用V或C表示，例如CCCVVVCCVV可表示为CVCV. 下式为元音辅音连续出现的情形

$$\begin{cases}\mathrm{CVCV}\cdots\mathrm{C}\\ \mathrm{CVCV}\cdots\mathrm{V}\\ \mathrm{VCVC}\cdots\mathrm{C}\\ \mathrm{VCVC}\cdots\mathrm{V}\end{cases} \tag{2}$$

(2)式所代表的4种单词元辅音组合形式可统一表示为：

$$[C](\mathrm{VC})^{m}[V] \tag{3}$$

m可作为后缀剥离的判别条件. 当$m=0$时将转成空字. 后缀剥离算法规则如下：

$$(\text{判别条件})\mathrm{S1}\rightarrow\mathrm{S2} \tag{4}$$

(4)式的含义是对后缀为 S1 的单词，如果后缀前面的单词满足判别条件的要求，那么后缀 S1 变为 S2 的形式. 判别条件一般是根据 m 取值确定，例如：

$$(m>1)\text{EMENT}\rightarrow \tag{5}$$

式中后缀 S1 是“EMENT”，如果满足判别条件 $m>1$ 那么后缀 S1 将被 S2 取代. 由于 S2 为空字，那么经过上述步骤就起到了对单词后缀进行剥离的效果. 例如单词“REPLACEMENT”，前缀“REPLAC”按照元辅音组合形式可转化为“CVCVC”形式，那么 $m=2>1$ 满足判别条件，对后缀“EMENT”进行剥离，单词“REPLACEMENT”变为“REPLAC”，这就起到了简化作用，并且便于标签信息的分类.

1.3　标签推荐过程

实际上，ProgrammableWeb 上的 Mashup 服务很多并未使用 API 服务，所以利用 Tag 进行服务推荐要比直接采用 API 方式高效得多. ProgrammableWeb 平台上的 Mashup 服务标签数量 n 不尽相同，但一般在[1,6]上取值，最常采用的是 3 标签形式. 由于这里采用的是基于标签的推荐算法，当服务或 API 的标签信息过少时会影响推荐的质量，因此针对这部分标签信息需要采取扩充手段：首先计算查询者给出的标签概念和相关标签之间的共现参数，根据该参数值进行排序，选取前 k 个作为后备标签. 共现参数的计算方法有很多，如 Jaccard 算法[8]：

$$\text{Co}(t_i,\ t_j)=\frac{|t_i\ \bigcap\ t_j|}{|t_i\ \bigcup\ t_j|} \tag{6}$$

其中：$|t_i\bigcap t_j|$的含义是在 Mashup 网络的所有服务中含有 t_i 和 t_j 的数量，$|t_i\bigcup t_j|$的含义是在 Mashup 网络的所有服务中含有 t_i 或者 t_j 的数量. 经过上述操作，使得用户给出的每一个服务标签 $u\in U$ 都具备了一定数量的后备标签.

需要用到 3 种不同的标签：用户给定标签集 U、后备标签集 C 和推荐标签集 R. 利用 Jaccard 公式计算后备标签的共现参数，依据该参数值对后备标签进行排序，从而得到推荐标签集 R，计算公式如下[9]：

$$\begin{cases}\text{vote}(t,\ u)=\begin{cases}1,\ \text{if } t\in T_u\\0,\ \text{otherwise}\end{cases}\\ \text{score}(t)=\sum\limits_{u\in U}\text{vote}(t,\ u)\\ \text{score}(c)=\sum\limits_{u\in U}\text{Co}(u,\ c)\end{cases} \tag{7}$$

2　多属性评价融合评级的 Mashup 服务推荐算法

2.1　融合多属性的 API 模型

在该模型中，对 API 属性采取语义标签的描述方式，这种描述一般是 API 开发者或使用者给定的，从某种意义上是对 API 的评价，文字评价通过人脑的运算可以辨别，但对于电脑则需要通过精确合理的数值计算确定[10].

定义 3(API 库)　融合社会和功能属性的 API 模型库定义如下：

$$\mathscr{R}=\langle\omega,\ \psi,\ \chi,\ \eta,\ \delta,\ \iota,\ \vartheta,\ \mu\rangle \tag{8}$$

其中：ω,ψ,χ,η 是 4 个有限集合，分别为 WebAPI 集合、Mashup 集合、网络开发者集合以及语义标签；$\delta\subseteq\psi\times\chi$ 代表的是 Mashup 集合中的服务程序与网络开发者集合中开发者的隶属关系；$\iota\subseteq\omega\times\psi$ 代表的是有限集中的 Mashup 服务和 API 服务的包含关系；$\vartheta\subseteq\omega\times\psi\times\eta\times\chi$ 是有限集的四元关系，表示设计者对 API 的语义标签定义对给定 mashup 的影响；μ：$\omega\times\psi\times\chi\mapsto[0,\ 1]$，是 API 开发者给定的量化等级，用来评价该 API 是否适合于给定的 Mashup 服务.

该模型由于考虑到了服务程序与网络开发者集合中开发者的隶属关系，允许开发者对别的开发者拥有的 Mashup 服务进行语义标签的赋值及评价，因此该模型充分利用了 API 服务的社会关系，有利于更加全

面地评价 API 服务的等级.

2.2 网络 API 查询和检索目标

在模型库中对任意一个 API 服务 $w \in \omega$，属性描述主要有：名称 n_w，唯一资源标识符 URI_w，语义标签集 $\{t_w\}$ 和技术特征集 F_w. 模型库中的 API 服务可定义如下：

$$w = \langle C_w, \{t_w\}, F_w \rangle \tag{9}$$

类似地，网络 API 请求是网络设计者根据查询目标提出的，网络 API 请求可定义如下：

$$\eta = \langle C_\eta, \{t_\eta\}, \{\upsilon_\eta\}, F_\eta \rangle \tag{10}$$

式中：C_η 是网络 API 请求的类别信息，用于与 API 库中的类别 C_w 进行匹配；$\{t_\eta\}$ 是查询用户定义的查询关键词，用于与 API 库中的语义标签 $\{t_w\}$ 进行匹配；$\{\upsilon_w\}$ 是 Mashup 默认的 API 服务，是被替换的对象；F_η 是可选择属性组合的集合，例如技术特征、数据格式等.

API 检索是通过计算用户描述和库中 API 间的相似性来进行的，该相似性可定义如下：

$$\begin{aligned} \mathrm{Sim}(\eta, w) = {} & \zeta_1 \cdot \mathrm{Sim}_c(\eta, w) + \zeta_2 \cdot \mathrm{Sim}_t(\eta, w) + \\ & \zeta_3 \cdot \mathrm{Sim}_{comp}(\eta, w) + \sum_{i=4}^{N} \zeta_i \cdot \mathrm{Sim}'_i(\eta, w) \in [0, 1] \end{aligned} \tag{11}$$

式中：$\mathrm{Sim}_c(\eta, w) \in [0, 1]$ 是类别的相似性；$\mathrm{Sim}_t(\eta, w) \in [0, 1]$ 为语义标签的相似性；$\mathrm{Sim}_{comp}(\eta, w)$ 是查询者定义的 η 与 API 模型库 $\mathscr{R}$ 中可选择的 API$w \in \omega$ 之间的组合相似性；$\mathrm{Sim}'_i(\eta, w)$ 是 API 技术特征之间的相似性. $\mathrm{Sim}_c(\eta, w)$ 和 $\mathrm{Sim}_t(\eta, w)$ 计算方法参照文献[11]，$\mathrm{Sim}_{comp}(\eta, w)$ 与 $\mathrm{Sim}'_i(\eta, w)$ 的计算公式如公式(12),(13) 所示.

$$\begin{cases} Z = Mash\,\mathrm{Sim}(\psi_1, \psi_2) = \dfrac{2 \cdot |\psi_1 \cap \psi_2|}{|\psi_1| + |\psi_2|} \\ \mathrm{Sim}_{comp}(\eta, w) = \dfrac{\sum\limits_{i=1}^{|\chi_w|} \dfrac{\sum\limits_{k=1}^{|L_i|} \sigma_i \cdot Z(\{\upsilon_?\}, \psi_i^k)}{|L_i|}}{|\chi_w|} \end{cases} \tag{12}$$

式中：$\chi_w \in \chi$ 是使用 API 服务 w 的网络开发者集合，$|\cdot|$ 是开发者数量，L_i 代表开发者 $\chi_i \in \chi$ 拥有的 Mashup 集，$\psi_i^k \in \psi_i$，σ_i 是开发者 $\chi_i \in \chi$ 的技能系数，该系数与开发者拥有的 Mashup 数量相关.

$$\mathrm{Sim}'_i(\eta, w) = \begin{cases} \dfrac{2 \cdot |F_\eta^i \cap F_w^i|}{|F_\eta^i| + |F_w^i|}, & \text{for single} \\ \dfrac{|F_\eta^i \cap F_w^i|}{|F_\eta^i|}, & \text{otherwise} \end{cases} \tag{13}$$

式中针对单一 API 服务选择和多个 API 服务选择情况进行区分，采用不同的相似度计算公式. 式(11) 中的权重参数 $0 \leqslant \zeta_i \leqslant 1$，$\sum\limits_{i=1}^{N} \zeta_i = 1$，选取方式如下：

$$\begin{cases} \zeta_1 = \zeta_2 = \zeta_i, & \text{for single select} \\ \zeta_1 = 0.1,\ \zeta_i = c, & \text{Mashup completion} \\ \zeta_3 = \zeta_i, & \text{Proactive mashup completion} \\ \zeta_1 = 0.1,\ \zeta_i = c, & \text{Web API substitution} \end{cases} \tag{14}$$

通过计算上述相似度，查询返回的将是相似度大于 Mashup 服务开发者给定阈值 $\gamma \in [0, 1]$ 的 API 服务. 同时可以给出 API 服务的评价标准：

$$\rho(w) = \frac{\mathrm{Sim}(\eta, w)}{|\chi_w|} \cdot \sum_{i=1}^{|\chi_w|} \frac{\sum\limits_{k=1}^{|\psi_i|} \sigma_i \cdot \mu(w, \psi_i^k, \chi_i)}{|\psi_i|} \tag{15}$$

式中：$\chi_{w'}$ 是在 Mashup 服务中使用 API 服务 w 的开发者集合；其它参数定义类似于公式(12).

本文得到的 API 服务候选方法将综合 API 开发者和使用者对 API 服务的定义和评价，并且给出了量化的评价指标. 这种选择方式将更倾向于选择与其他 Mashup 已有 API 服务存在联系的 API 服务，而对于不满足这个条件的 API 选取的可能性较小.

2.3　社会及功能属性扩充

定义 4(功能语义及接口参数)　在判断 API 社会属性时主要考虑 API 的 3 个属性(如图 2*a*)：动作、对象及 API 的接口参数，这 3 个都是 API 选择等级需要考虑的元素，因此可定义为：

$$C = \langle \text{action}, \text{object}, \text{I/O} \rangle \tag{16}$$

在进行匹配时，可分为精确匹配和模糊匹配. 精确匹配指两个功能语义的动作和对象完全一致. 这种情况的检索无需候选 API 的选取，方式比较简单. 但实际应用中由于查询者和开发者对于 API 的理解或定义不可能相同，大多采用的是模糊匹配方式. 对于模糊匹配相似度的计算需要用到两个元素：父类匹配度(parHeit) 及与父类的距离($d(C_i)$). 相似度计算公式如下：

$$\text{CoSim}(C_1, C_2) = \frac{2 \cdot \text{parHeit}(C_1, C_2)}{d(C_1) + d(C_2)} \tag{17}$$

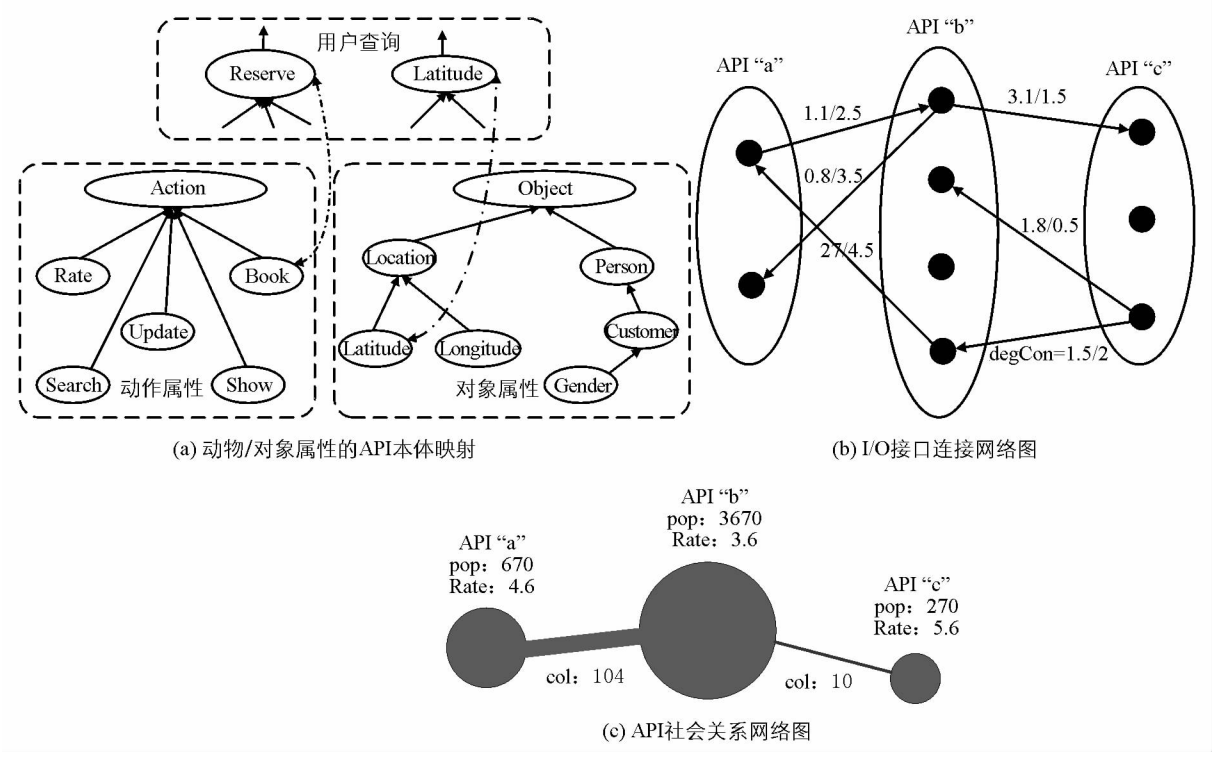

图 2　API 扩充模型示意图

图 2(b) 是一个简化的 API 社会关系网络图，图中的节点代表 API，连线代表 API 之间的相互关系紧密度(col). 知名度(pop) 表示的是该 API 被使用的次数，$rate$ 是用户对于 API 评价的平均值. 则 API 的社会关系评价公式为[12−13]：

$$So\rho(w) = \frac{\sum col(w) \cdot \sum rate(w)}{\sum pop(w) \cdot Maxrate(w)} \tag{18}$$

图 2(c) 是简化的 API 接口连接图，图中显示了每个 API 节点的 I/O 接口数量及相关参数，$|ExaMat|$ 代表语义精确匹配的数量，$|PartMat|$ 是模糊匹配的数量. 则 API 之间的连接强度可定义如下：

$$\underset{(u \to v)}{degCon} = \frac{|ExaMat| + \alpha \cdot |PartMat|}{|v's\ inputs|} \tag{19}$$

$$Con\rho(\omega)=\frac{\sum_{u,v}^{q} degCon_{(u\to v)}}{q} \tag{20}$$

式中：q 代表接口连接网络中所有的连接数量；$Con\rho(\omega)$ 是 API 服务 ω 连接等级评价. 则社会及功能属性扩充等级评价采用加权方式，可定义如下：

$$As(w)=\beta_1 \cdot Soc\rho(w)+\beta_2 \cdot Con\rho(w) \tag{21}$$

式中：β_1 和 β_2 是权重系数，且 $\beta_1+\beta_2=1$，$0\leqslant As(w)\leqslant 1$.

对原评价标准(15)与扩充标准(20)采取加权融合的方式：

$$Rank(w)=\lambda \cdot \rho(w)+(1-\lambda)\cdot As(w) \tag{22}$$

3 仿真结果及分析

采用前述 ProgrammableWeb 上获取的 Mashup 服务信息数据库对算法进行性能测试. 数据库含有 650 个 API 服务，并且按照本文算法获得名称、知名度、用户评价、类别和技术特征等，组成 API 服务模型. 然后根据 API 服务的说明手册得到 821 个操作和 8 735 个输入输出参数，从这些操作和输入输出参数中提取 API 的功能语义、操作对象及 I/O 口参数等信息组成 API 服务的扩充模型[14].

建立 API 数据库后，构建 API 请求的接口连接图和社会关系网络图，利用第 2 节的计算方法对 20 个自然语言的 API 查询进行仿真测试. 建立原始 API 库和扩充库的主要目的是为便于与原有方法进行实验对比，最后选取评价指标大于给定阈值的 API 服务作为推荐对象. 各权重参数设置如下：

$$\begin{cases}\zeta_1=0.1,\ \zeta_i=0.225\\ \beta_1=\beta_2=0.5\\ \lambda=0.4\end{cases} \tag{23}$$

为简化服务推荐算法复杂度，API 模型中的可选属性只选择数据格式，主要是考虑程序的兼容性，所以 $\zeta_i=0.225(i=2,3,4,5)$，$\beta_1=\beta_2=0.5$ 表示 API 的社会关系评价和接口参数匹配程度同等重要，$\lambda=0.4$ 表示在总体评价时稍微侧重于社会关系评价和接口参数匹配程度的评价指标. 对比算法选取 DTV，FRA 及 MAFRA 3 种算法[14]，算法性能的评价指标是查全率和查准率. 对比结果如图 3 所示，图 4 显示的是融合算法权值两种临界值与本文所选权重的 Mashap 服务推荐效果对比.

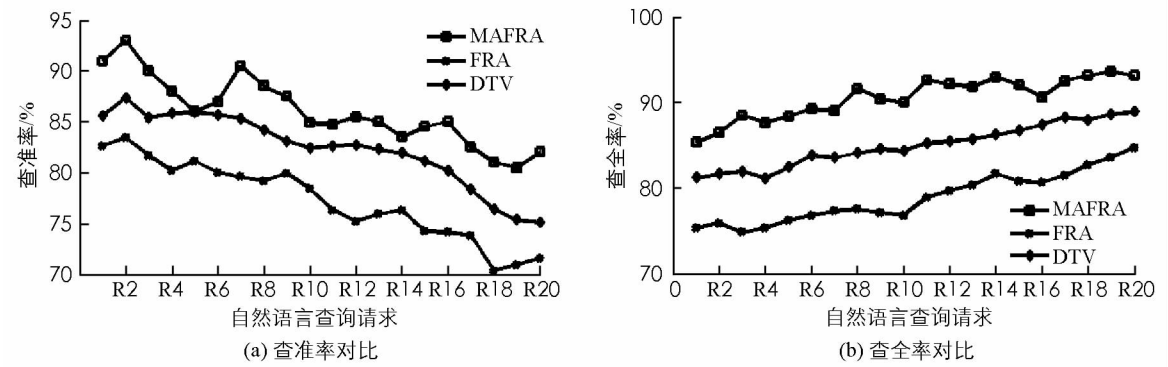

图 3 3 种算法的查准率和查全率对比

图3显示的是3种算法的查全率与查准率两项评价指标的对比结果. 在查准率方面(图3(a))，MAFRA 算法的最佳效果是 93.1%(R2)，最差效果 80.4%(R19)，平均查准率为 86.0%；FRA 算法的最佳效果是 83.4%(R2)，最差效果 70.3%(R18)，平均查准率为 77.2%；DTV 算法的最佳效果是 87.3%(R2)，最差效果 75.3%(R20)，平均查准率为 82.3%. 在查全率方面(图 3(b))，从图中可以看出：MAFRA 算法的平均查全率在 90% 左右；FRA 算法的平均查全率在 80% 左右；DTV 算法的平均查全率在 85% 左右. 由此可知，MAFRA 算法在查准率和查全率方面都要好于 FRA 和 DTV 算法，说明 MAFRA 算法推荐的 API 服务

的准确度最高，并且 MAFRA 算法推荐的 API 服务更加全面，数量更多选择余地更大.

公式(22)的加权融合方式涉及到权重 λ 的选择，本文选取 $\lambda=0.4$. 当 $\lambda=1$ 时，$Rank(w)=\rho(w)$，演变为基于未扩充模型的服务推荐算法(FRA)；$\lambda=0$ 时，$Rank(w)=As(w)$，演变为基于扩充模型的服务推荐算法. 图 4、图 5 给出的是这 3 种情况的查全率和查准率的效果对比. 融合后的服务推荐算法总体效果要明显好于未融合的服务推荐效果.

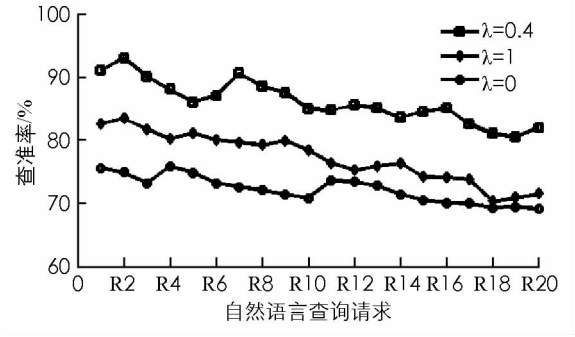

图 4　融合权重不同取值的查准率对比

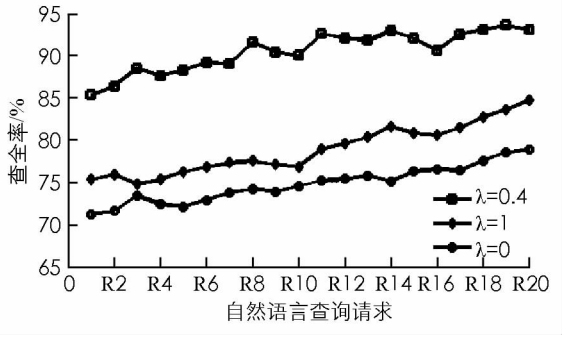

图 5　融合权重不同取值的查全率对比

4　结束语

对 Mashup 服务的聚类推荐算法进行研究，针对传统 Mashup 服务推荐算法中对所推荐 API 服务的社会和功能属性利用较少，不利于全面评价所推荐 API 适用度的问题，对 API 模型进行扩充，提出一种多属性融合评级的 Mashup 服务推荐算法. 仿真实验表明本文算法具有较高的查全率和查准率.

参考文献：

[1] WOOD J, DYKES J, SLINGSBY A. Interactive Visual Exploration of a Large Spatio-Temporal Dataset: Reflections on a Geovisualization Mashup [J]. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(6): 1176－1183.

[2] GUO J X, TOKUDA T. Description-Based Mashup of Web Applications [J]. Current Trends in Web Engineering, 2010 (6385): 545－549.

[3] STOLEE K T, ELBAUM S. Identification, Impact, and Refactoring of Smells in Pipe-Like Web Mashups [J]. IEEE Transactions on Software Engineering, 2013, 39(12): 1654－1679.

[4] FUNG B C M, TROJER T, LI X. Service-Oriented Architecture for High-Dimensional Private Data Mashup [J]. IEEE Transactions on Services Computing, 2012, 5(3): 373－386.

[5] TANG X Q, ZHU P. Hierarchical Clustering Problems and Analysis of Fuzzy Proximity Relation on Granular Space [J]. IEEE Transactions on Fuzzy Systems, 2013, 21(5): 814－824.

[6] STIRBU V, YU Y, ROIMELA K. A Lightweight Platform for Web Mashups in Immersive Mirror Worlds [J]. Pervasive Computing, 2013, 12(1): 34－41.

[7] HAN H, XUE Y X. Mashup Technology: Beyond Open Programming Interfaces [J]. Computer, 2013, 46(12): 96－99.

[8] GEBHARDT H, GAEDKE M, DANIEL F. From Mashups to Telco Mashups: A Survey [J]. Internet Computing, 2012, 16(3): 70－76.

[9] 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.

[10] ZHOU C Y, CHEN H J, PENG Z P. Ontology-Driven Mashup Auto-Completion on a Data API Network [J]. Tsinghua Science and Technology, 2010, 15(6): 657－667.

[11] YU T W, PENG H S. Hierarchical Clustering of High-Throughput Expression Data Based on General Dependences [J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2013, 10(4): 1080－1085.

[12] PORTER M F. An Algorithm for Suffix Stripping [J]. Program: Electronic Library and Information Systems, 2006,

40(3): 211－218.

[13] MAXIMILIEN E M, RANABAHU A, GOMADAM K. An Online Platform for Web APIs and Service Mashups [J]. Internet Computing, 2008, 12(5): 32－43.

[14] 黄 媛, 李 兵, 何 鹏, 等. 基于标签推荐的Mashup服务聚类 [J]. 计算机科学, 2013, 40(2): 167－171.

On Mashup Service Recommendation Based on Multi Attributes Fusion Rating Algorithm

ZHAO Guo-dong[1], SONG Li-ya[2], JIAO Xiao-gang[1]

1. School of Mathematics and Computer, Ningxia University, Yinchuan 750021, China;

2. School of Physics and Electronics Information Engineering, Ningxia University, Yinchuan 750021, China

Abstract: The traditional Mashup service recommendation is a retrieval method based on keywords, the use of social and functional properties for the recommended API service is too less to make a comprehensive evaluation of the recommended API applicability, so the Mashup service recommendation based on multi attributes fusion rating algorithm has been proposed to solve this problem. Firstly, the climbing tools have been used to collect ProgrammableWeb Mashup service information, and the suffix stripping algorithm been used to modify the Mashup service label with noun form as the research and analysis data sets. Secondly, the API model has been extended with the social and the functions attributes, then the multiple attribute similarity weighted fusion has been used to evaluate candidate API fitness, which would be the API service recommendation basis. The experimental results show that, the multiple attribute fusion rating Mashup service recommendation algorithm has higher accuracy and faster computing time, which is feasible and effective.

Key words: multiple attribute; Mashup; tag; fusion

责任编辑 张 枸