融合功能语义关联计算与密度峰值检测的 Mashup 服务聚类方法

陆佳炜[1)] 吴 涵[1)] 张元鸣[1)] 梁倩卉[2)] 肖 刚[1)]

[1)](浙江工业大学计算机科学与技术学院 杭州 310023)

[2)](南洋理工大学计算机科学与工程学院 新加坡 637457 新加坡)

摘 要 随着互联网上 Mashup 服务数量及种类的急剧增长,如何从这些海量的服务集合中快速、精准地发现满足用户需求的 Mashup 服务,成为一个具有挑战性的问题.针对这一问题,本文提出一种融合功能语义关联计算与密度峰值检测的 Mashup 服务聚类方法,用于缩小服务的搜索空间,提升服务发现的精度与效率.首先,该方法对 Mashup 服务进行元信息提取和描述文本内容整理,并根据 Web API 组合的标签对相应 Mashup 服务标签进行扩充.然后,基于功能语义关联计算方法(Functional Semantic Association Calculation Method,FSAC)提取出各服务描述的功能名词集合,并通过功能名词的语义权重来构造 Mashup 语义特征向量.最后,通过基于密度信息的聚类中心检测方法(Clustering Center Detection Method based on Density Information,CCD-DI)检测出最为合适的 K 个 Mashup 语义特征向量作为 K-means 算法的初始中心,进行聚类划分.基于 ProgrammableWeb 的真实数据实验表明,本文所提聚类方法在纯度、精准率、召回率、熵等指标上均有良好表现.

关键词 功能语义;Mashup 服务;密度峰值;聚类;Web API

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2021.01501

Mashup Service Clustering Method Via Integrating Functional Semantic Association Calculation and Density Peak Detection

LU Jia-Wei[1)] WU Han[1)] ZHANG Yuan-Ming[1)] LIANG Qian-Hui[2)] XIAO Gang[1)]

[1)](*School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou* 310023)

[2)](*School of Computer Science and Engineering, Nanyang Technological University, Singapore*, 637457 *Singapore*)

Abstract With the rapid growth in the number and type of Mashup service on the Internet, how to quickly and accurately find Mashup services that meet user needs from these massive services has become a challenging problem. Service clustering technology can simplify the Web API recommendation process, and a lot of different approaches have been proposed. Many of them mainly focus on the semantic similarities research from the Web service document to guide clustering operations. But it usually uses the K-means or its improved algorithms to cluster the services, and did not propose an effective solution to the initial clustering centers selection problem for K-Means. Moreover, most service description documents are short texts, often have a limited contextual information, and they are sparse, noisy and ambiguous, and hence, automatically mining the hidden functional information from them remains an important challenge. Traditional mining algorithms such as LDA are difficult to represent short texts and find satisfactory clustering effects from them. Aiming at these problems, we investigate services and their compositions in

收稿日期:2020-03-26;在线发布日期:2021-03-31.本课题得到国家自然科学基金(61976193)、浙江省自然科学基金(LY19F020034)、浙江省重点研发计划项目(2021C03136)资助.**陆佳炜**,硕士,讲师,中国计算机学会(CCF)会员,主要研究领域为服务计算、软件架构、大数据可视化.E-mail:viivan@zjut.edu.cn.**吴 涵**,硕士研究生,主要研究方向为服务计算.**张元鸣**,博士,副教授,主要研究方向为服务计算、大数据.**梁倩卉**,博士,研究员,主要研究领域为数据科学.**肖 刚**(通信作者),博士,教授,主要研究领域为智能制造、云制造.E-mail:xg@zjut.edu.cn.

ProgrammableWeb which characterize services as APIs and their compositions as Mashups. A Mashup service clustering method via integrating functional semantic association calculation and density peak detection is proposed in this paper, which is used to reduce the search space of services and improve the accuracy and efficiency of service discovery. In the initial stage of the method, each Mashup service description text is normalized, and the Mashup service tag is extended from the Mashup and Web APIs. Then, according to the functional semantic association calculation method (FSAC), the functional noun set of each service description is extracted, and the functional semantic weights of these functional nouns are calculated. Clustering Mashup services based on function similarities would greatly boost the ability of services search engines to retrieve the most relevant Web services. Further, the nouns with higher functional semantic weights are represented as Mashup semantic feature vectors. Finally, the most suitable K Mashup semantic feature vectors are detected by the clustering center detection method based on density information (CCD-DI), which are used as the initial center of the K-means algorithm for clustering. A series of experiments are carried out with real data from ProgrammableWeb. We first mine the contents of the Mashup service documents to extract the functional words describing the meaning of the services. The extracted data is compared with manual to verify the effectiveness of FSAC. Then we use two criteria to evaluate the performance of our approach, namely Precision and Recall. Precision can be seen as a measure of exactness or fidelity, whereas Recall is a measure of completeness. Moreover, we also employ purity and entropy to evaluate the clustering accuracy. Compared with existing methods, the results show that the proposed clustering method has good performance in terms of precision, recall, purity, and entropy. In addition, we use t-SNE tools to visualize the vectors in Mashup based on the TF-IDF, LDA and MFSF methods, respectively. The vector visualization results demonstrate the interpretability of our method by discovering related and consistent clusters.

Keywords functional semantics; Mashup service; density peak; clustering; Web API

1 引 言

近年来,随着云计算、大数据等技术与互联网的深入融合,大量 Web API 服务也不断在互联网上涌现.但是传统的 Web 服务功能单一,无法有效满足用户灵活多变的需求.Mashup 技术,是一种混搭不同功能和风格的 Web API 服务构成的组合应用,被软件开发者用于构建满足复杂需求的程序[1-2].这一便捷高效的开发技术能有效提升软件开发过程中的复用粒度,软件开发者可以从 ProgrammableWeb① 等公开的服务仓库中挑选出满足不同场景需求的 Web API 来搭建 Mashup 服务.然而,现有主流的公共服务仓库都注册了海量的 Mashup 和 API 服务.因此,如何挑选出合适的 Web API 服务和具有价值的 Mashup 服务已经成为软件开发人员需要解决的重点问题.另一方面,Mashup 服务还存在着服务描述文本过于简短、人工分类不合理等问题,这也在一定程度上增加了服务搜索工作的难度.

现有的研究表明[3-11],通过 Mashup 服务聚类技术能有效提升服务的搜索效率,并进一步优化 Web API 推荐流程.在这些服务聚类的研究工作中,有不少学者利用 Web 服务描述语言(Web Service Description Language,WSDL)从服务文档中提取功能特征,并结合服务标签进行聚类[3-4].可是由于目前大多数的 Mashup 服务使用自然语言而非 WSDL 文档的方式描述服务,这对服务功能特征的提取造成了一定的难度.所以许多的研究[5-10]开始基于 LDA[12]模型对 Mashup 服务进行面向功能主题的聚类,或者利用 TF-IDF[13]、Word2Vec[14]等模型对服务描述文本构造特征向量完成聚类.另外还有学者[11]从 Mashup 服务结构相似性的角度来优化聚类效果.从整体来看,目前 Mashup 服务聚类的研究重点还是在服务功能语义匹配计算的层面,并

① https://www.programmableweb.com/

且已经做了大量的改进.但是,它们普遍还存在以下问题:

(1)现有 Mashup 服务描述文本大都存在功能描述含糊不清、特征稀疏等问题.受限于缺乏优质的训练语料,采用 LDA 或其改进模型来提升服务聚类精度的效果并不明显[15].另外,如果单从 TF-IDF 技术角度出发来进行服务功能特征表示,则没有考量好描述文本与服务标签间的语义联系及文本内部的关联结构,又会造成描述文本的语义丢失,进而错失关键的功能语义信息;

(2)目前多数的 Mashup 服务聚类研究[3-6,8-10]都将研究的重心聚焦于对文本向量模型地改善,而对于聚类的具体实现,基本还是采用 *K*-means[16]算法或其改进型算法,但 *K*-means 初始中心的选择又对最终的聚类效果有较大的影响,因此解决这一问题也可以作为提升 Mashup 服务聚类精度的重要方向.

为了能够合理有效地解决上述问题,本文主要做了以下的工作:

(1)针对如何有效表示 Mashup 服务功能语义信息的问题,本文首先从 ProgrammableWeb 爬取了大量真实的 Mashup 服务及 Web API 服务.进一步给出了 Mashup 服务数据的预处理方案,用于对服务描述信息进行规范性整理.然后,结合服务标签和服务描述文本间的信息,给出 FSAC 方法来计算服务的功能语义关联信息.最后,本文将 FSAC 方法计算结果作为语义制约因子嵌入 TF-IDF 权重公式,使权重计算敏感于文档的语义关联变化,并结合 Word2Vec 模型构造出能够有效表示服务核心功能特征的 Mashup 语义特征向量;

(2)针对服务聚类算法的优化问题,本文以传统密度峰值聚类(Density Peak Clustering,DPC)算法[17]为基础,提出了一种基于密度信息的聚类中心检测方法 CCD-DI.该方法改进了 DPC 算法中局部密度的定义方式,并能依据 Mashup 描述文本的语义特征,自适应确定最优的聚类中心,以此提升 Mashup 服务聚类的最终效果;

(3)使用爬取的真实数据与其它聚类算法进行了实验评估,实验结果表明我们提出的方法具有良好的聚类效果.

本文第 2 节介绍 Mashup 服务聚类与 DPC 算法的相关工作;第 3 节详细阐述所提聚类方法;第 4 节为实验证明与分析;第 5 节进行论文总结与未来展望.

2　相关工作

由于服务聚类可以有效缩小服务搜索空间,提升服务发现效率[18],所以长期以来都广受国内外学者的关注,目前针对这一领域已有大量的研究.在较早的研究工作中,不少学者倾向于利用 WSDL 文档挖掘服务功能属性进行服务功能相似性聚类[3-4,19].例如文献[3-4]就基于服务标签和 WSDL 文档相似性来获取服务间的功能相似度.文献[19]从 WSDL 文档的类型、内容等信息中提取服务特征,进行功能匹配,提升服务聚类的精度.但是由于现有的 Mashup 服务缺乏规范性的 WSDL 文档以及相关服务属性说明,这就促使研究学者从其他角度展开研究来对 Mashup 服务进行功能特征提取.

现有大多数研究方法[5-10,15]都是从 Mashup 服务描述文本中抽取主题特征,进而进行聚类建模来展开研究.文献[5-6,8]提出了一类 Mashup 网络模型,综合考虑了 Mashup 服务的标签、类别、描述、Web API 等信息,并融合 LDA 模型来进行 Mashup 服务主题模型的计算,确保了 Mashup 服务聚类的准确性.文献[7]通过结合 Word2Vec 和 LDA,设计了一种高质量的词向量模型,用于提升服务的聚类精度.文献[20]则借助 Doc2Vec 模型来学习服务描述中的语义信息,进而训练高质量的文本向量进行服务聚类.文献[10]在 Mashup 的聚类过程运用了一种基于监督 LDA 的主题模型,改进了 Mashup 服务的聚类效果.文献[21]在其研究中集成了递归卷积神经网络、增强型概率主题模型,用以准确提取服务功能特征,优化聚类效果.虽然已有许多研究对 LDA 模型做了改进,但是大多数改进模型还需要预先设定主题的数量.基于此,文献[15]采用了 HDP 模型和 SOM 神经网络进行主题挖掘,在聚类过程中自动确定最优主题数.此外,相较于 LDA 这类传统的主题模型,一些研究人员认为基于词频信息的统计方法或许更简便高效[9,22].例如文献[9]提出 TF-IDF 向量表示 Mashup 服务描述文本,并对向量进行 *K*-means 算法聚类.但是此类方法对描述文本内部及服务标签间的语义联系缺乏考虑,因此如何有效地对服务的功能信息进行表征,也是该类方法改进的一个重要方向.

纵观 Mashup 服务聚类的研究现状,许多的研究工作还只是基于 *K*-means 算法或其改进型算法进行 Mashup 服务聚类的实现,未能针对性地

解决 K-means 初始中心选择问题. 因此本文决定引入 DPC 算法并加以改良来解决这一问题. DPC 算法[17]的核心思想是基于数据点的局部密度 ρ 和最近距离 δ 来绘制决策图，进而快速确定聚类中心. DPC 算法能够用于任意形状数据的聚类分析，但是在实际运用中它还存在以下不足[23-27]：(1) 截断距离 d_c 的选取会影响聚类的结果；(2) 在数据量较大的情况下，从决策图中挑选出合适的聚类中心点将会变得困难.

在截断距离选取的优化上，文献[23]利用信息熵来进行截断距离的选取，通过控制密度估计熵来获得最优的截断距离. 文献[24] 通过邻域信息来计算数据点的局部密度. 文献[25]基于欧几里得距离计算局部密度，代替了直接使用截断距离计算局部密度的方案. 在聚类中心选取的优化上，目前主流的研究方案仍然是将局部密度 ρ 和最近距离 δ 作为聚类中心的计算依据. 例如，文献[26]提出了一种自适应筛选聚类中心的方法，在判断聚类中心时得到了较好的效果. 文献[25]将数据点基于 ρ 与 δ 进行了分类，并在聚类中心提取过程中融入了置信度检测、逻辑回归等方法. 文献[27]在计算局部密度时结合了一种新颖的残差计算方法，在绘制的决策图中简化了聚类中心的选取过程. 此外，比较典型的研究还有文献[23]，通过计算数据点的 δ 值标准差和判断聚类中心候选点的局部密度 ρ 来挑选出最优的聚类中心.

通过对 Mashup 服务聚类、密度峰值聚类的相关工作分析，本文提出了一种融合功能语义关联计算与密度峰值检测的 Mashup 服务聚类方法，用于精准地提取 Mashup 描述文本的语义特征，自适应确定最优的聚类中心，从而改善 Mashup 服务聚类的最终效果.

3 方 法

为了能够有效提升 Mashup 服务聚类的精度，本文将所提 Mashup 聚类方法分为两步进行：(1) 将爬取到的真实 Mashup 数据进行预处理，接着利用 FSAC 方法对服务描述中的功能名词进行精准定位以及语义关联计算. 在此基础上，结合 Word2Vec 模型构造出能够体现 Mashup 服务功能特征的语义特征向量；(2) 在得到每个 Mashup 服务的语义特征向量后，利用 CCD-DI 方法计算出 Mashup 数据中最为合适的初始聚类中心，最后利用 K-means 算法实现聚类. 图 1 展示了整体的方法框架，下面各小节也将围绕框架内容进行详细介绍.

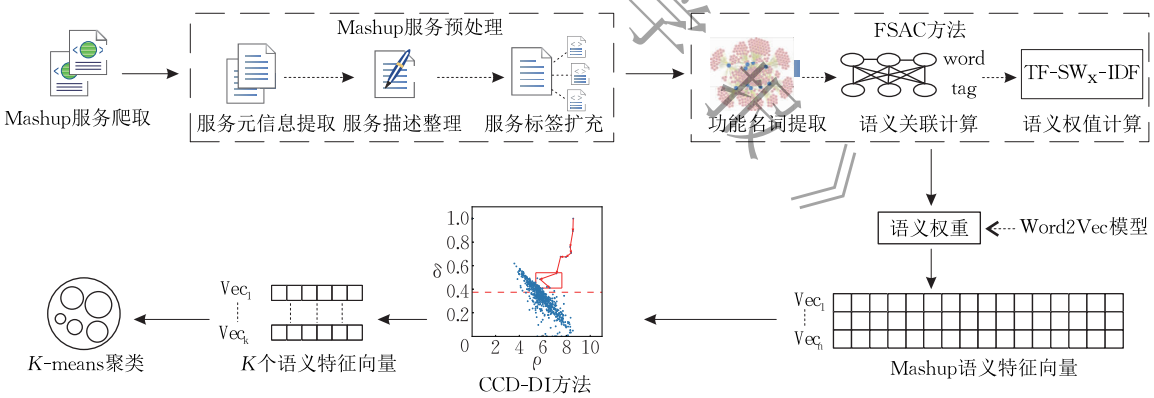


图 1 方法总体框架

3.1 Mashup 服务预处理

本文对源自 ProgrammableWeb 等平台上爬取到的 Mashup 服务进行分析，发现 Mashup 服务大都存在着分类标签不全、描述文本内容简短等问题，所以在方法的初始阶段，需要先对爬取到的 Mashup 服务进行预处理操作. 操作过程具体如下：

(1) 提取服务元信息. 整合 Mashup 服务中有实际效益的信息，例如服务描述文本、标签信息、Web API 信息等. 并且从爬取的数据集上过滤过缺少服务描述文本的 Mashup 服务，以避免无效服务对整体结果的影响；统一所有缺失名称的服务，将其名称通过 ID 的形式进行表示，便于后续建模.

(2) 整理服务描述. 首先将有意义的符号进行转义，将其表达为英文单词的形式，以提高服务描述文本的质量. 然后，为提升下一阶段功能词汇的检索速度，对不含语义信息的字符进行剔除，例如“#”、“▲”等符号. 最后，对缺失字母的单词尽量进行补全，如果出现实在无法补全的情况，则需要将其删除.

(3) 扩充服务标签. 文献[15]采用同义词的方法扩充单词，文献[28]按 TF-IDF 权重的方法扩

充单词.不同于以上研究,本文将组成每条 Mashup 的 Web API 信息进行整理,再逐一爬取,通过引入 Web API 标签来缓解 Mashup 服务特征稀疏性的问题.

为增强 Mashup 服务的功能语义信息,我们之前曾尝试引入 Web API 服务描述文本来扩充 Mashup 服务描述,但是在后续的实验过程中效果并不理想.我们分析认为,Mashup 是一种混搭 Web API 的应用,不同 Web API 中的功能描述差异较大,导致从中提取的功能特征较为分散,进而弱化了 Mashup 服务本身的功能特征.另外,为了保障后续处理阶段的顺利进行,我们在预处理阶段进行了较为严格的服务过滤机制,剔除了不符要求的部分服务.但从数据集上来看,剔除的服务仅占爬取数据的 1%左右,因此不会对后续处理造成太大的影响.

3.2　基于 FSAC 的 Mashup 语义特征向量表示

在对 Mashup 服务进行预处理后,需要根据 Mashup 服务的元信息构建可以表示 Mashup 服务功能特征的语义特征向量,将其作为下一步 Mashup 服务聚类的处理单元.因此,构造 Mashup 语义特征向量也就转化成了提取 Mashup 服务功能语义信息以及基于功能语义信息进行向量化这两个子问题.经过对大量 Mashup 服务元信息的比较分析,我们发现 Mashup 服务描述文本中的名词信息也有助于体现功能语义,而形容词、副词等其它词性的单词,多数只是起修饰性作用,对整体的语义表达能力较弱.本文提出的 FSAC 方法将名词信息、标签信息和单词上下文信息综合考虑引入到语义计算过程中,以提高建模的准确性.它主要包括:(1)提取功能名词;(2)结合标签信息和单词上下文信息计算单词语义关联度;(3)计算语义权重值.

在步骤 1 中,本文借助 NLP 工具(Natural Language Toolkit,NLTK①)对每条 Mashup 服务描述文本进行了词性标注与分析处理,还通过收集各类停用词,以及 Mashup 服务名称、代词等无功能语义的名词成分来建立对比词库,以便于快速过滤出所需的功能名词.其核心步骤如下:

(1)使用 NLTK 工具对 Mashup 服务描述文本进行词性标注;

(2)根据词性标注和收集的停用词库,过滤掉停用词以及形容词、副词、量词等实际语义较弱的单词,减少无意义词对建模的影响;

(3)利用 NLTK 中的词形还原工具对名词单词进行词形还原,并将去重结果放入功能名词集合 FNS 中;

(4)去除 FNS 集合中无用单词或者功能语义较弱的单词,例如 Web API 服务名称或者 Mashup 服务名称.

图 2 为上述功能名词提取步骤的具体展示.随后,本文使用了 WordNet② 工具计算功能名词间的语义关联,以获得 Mashup 服务在 FNS 集合中的语义权重分布,具体步骤如算法 1 所示.

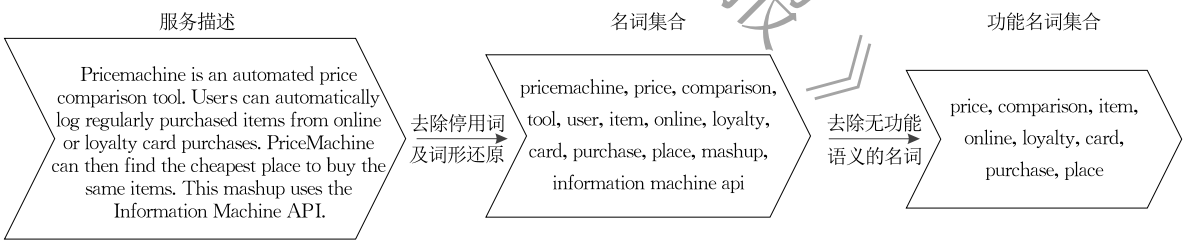


图 2　功能名词提取过程

算法 1.　Mashup 服务语义关联权重计算.

输入:Mashup 服务的服务标签集合 $Set(STag)$ 和功能名词集合 FNS

输出:Mashup 服务的功能语义权重字典 $SimDic$

1. $LenAvg=getLenAvg()$;
2. If(len(FNS)!=$LenAvg$)
3. $w=1/(|\text{len}(FNS)-LenAvg|)\times 2$;
4. End If
5. For n_i in FNS
6. $FSim_i=getFsim(n_i)$;
7. $CRSim_i=0$;
//计算第 i 个功能名词的语义权重校正量 $CRSim_i$
8. For j in $Set(STag)$
9. $CRSim_i=\max(CRSim_i,\text{WordNet}(ni,nj))$;
10. End For
//计算出功能名词 n_i 的最终语义权重 SW_i
11. $SimDic[n_i]=w\times FSim_i+(1-w)\times CRSim_i$;
12. End For
//只将语义权重较高的前 $LenAvg$ 个单词存放于字典 $SimDic$ 中
13. Sort($SimDic$,$LenAvg$);
14. Return $SimDic$;

① https://pypi.org/project/nltk/
② https://wordnet.princeton.edu/

$LenAvg$ 是所有功能名词集合的平均长度，通过 $getLenAvg()$ 函数（算法第 1 行）求出，具体计算公式如下：

$$LenAvg=\frac{\sum_{Set(FNS)}|FNS|}{|Set(FNS)|} \tag{1}$$

上述公式中 $Set(FNS)$ 为 FNS 的集合. $FSim$ 表示名词间语义关联权重，求出 $LenAvg$ 可用于评估功能名词数量对 $FSim$ 的影响，进而在算法 2～4 行动态调整参数 w. w 是 $FSim$ 对最终语义权重 SW 的影响因子，取值在 0 到 1 之间. w 初始默认值为 0.5，以便于在语义权重的计算过程中可以综合评估 FNS 中单词之间的语义关联度以及单词与服务标签之间语义关联度的影响.

通过函数 $getFsim(n_i)$ 可以得到功能名词 n_i 与其他名词之间的语义关联权重 $FSim_i$（算法第 6 行），其计算公式如下：

$$FSim_i=\sum_{j=0}^{|FNS|}\frac{\mathrm{WordNet}(n_i,n_j)}{|FNS|-1},\ i\neq j \tag{2}$$

$|FNS|$ 为该 Mashup 服务中的功能名词数量. $\mathrm{WordNet}(n_i,n_j)$ 为使用 WordNet 词典工具进行计算得到的单词语义相似度.

而在部分情况下，FNS 中可能会存在功能名词数量过多或过少的情况，这样 $FSim$ 的计算结果就可能因为过于偏离服务的核心功能特征而失去实际的参考意义. 因此，通过计算功能名词与服务标签之间的语义相似度信息，来进行每个功能名词的语义权重校正，进而得出语义权重校正量 $CRSim_i$（算法 8～10 行）. 在此基础上，算法可以通过调整影响因子 w 的计算结果，动态调整各功能名词的最优语义权重（第 11 行）.

此外，通过计算 $LenAvg$ 还可以让参与 Mashup 语义特征向量计算的功能名词保持相对合理的数量，即该 Mashup 服务的功能语义权重字典 $SimDic$ 中只存放语义权重较高的前 $LenAvg$ 个单词（算法第 13 行）. 从算法时间复杂度的角度来看，算法 1 主要由功能语义权重计算（5～12 行）与权重排序（第 13 行）两个核心步骤组成，时间复杂度分别为 $O(kn)$（k 为服务标签数量）与 $O(n\log n)$.

TF-IDF 是一种词频信息统计方法，可以用于衡量单词在文档中的重要性[29]. TF-IDF 算法统计单词的词频信息以区分单词对于文档的作用，若单词 w 在文档 d 中出现得次数越多，但是在其他文档中出现的频率很少，那么单词 w 在文档 d 中有较高的 TF-IDF 权重值，更有可能是关键性单词. 虽然通过 TF-IDF 方法计算单词的权重值比较便捷，但是仅以单词的 TF-IDF 权重来区分单词是否为关键性单词，仍然是不够全面的. 一些词频较高的专属形容词和特殊名词缺乏功能语义，无法合理反映出服务的实际功能特征. 但是它们在 Mashup 服务描述文本中也往往可能获得较高的 TF-IDF 权重值. 因此，对于每一条 Mashup 服务，本文取出字典 $SimDic$ 中每个单词的语义权重 SW_x，并针对性地将 SW_x 嵌入到 TF-IDF 算法中，以此求出 TW_x 权值，将其作为每个单词的最终语义权重. TW 权值的计算公式如下所示：

$$TW_x=\frac{TF\text{-}IDF_x}{1-SW_x} \tag{3}$$

为了能够有效地表示 Mashup 服务的功能特征，本文结合式(3)，并借助由谷歌新闻作为语料训练好的 Word2Vec 模型，将字典 $SimDic$ 中的单词向量化. 算法 2 给出了 Mashup 语义特征向量的具体计算过程.

算法 2. Mashup 服务语义特征向量计算.

输入：Mashup服务的词向量字典 $WDic[]$，单词的 TF-IDF 权重字典 $TDic[]$，功能语义权重字典 $SimDic$

输出：特征向量 $\boldsymbol{DVec}$

1. $start(\boldsymbol{DVec})$;
2. For x in $SimDic$
3. $\quad\boldsymbol{WVec}_x=getDic(WDic,x)$;
4. $\quad SW_x=SimDic.\mathrm{fromDic}(x)$;
5. $\quad TF\text{-}IDF_x=TDic[x]$;
6. $\quad TW_x=TF\text{-}IDF_x/(1-SW_x)$;
7. $\quad$If($getLen(SimDic)>0$ and $SW_x==0$)
8. $\quad\quad TW_x=0$;
9. $\quad$End If
10. $\quad\boldsymbol{DVec}+=\boldsymbol{WVec}_x\times TW_x$;
11. End For
12. Return $\boldsymbol{DVec}$;

首先，对服务向量 $\boldsymbol{DVec}$ 进行初始化（算法第 1 行）. 随后，在算法 3～6 行中，针对 $SimDic$ 中的每个单词，依次提取出和它相关的词向量 $\boldsymbol{WVec}_x$、语义权重 SW_x 和 TF-IDF 权重值 $TF\text{-}IDF_x$，进而计算出最终权重 TW_x. 在算法第 7 行，考虑了 $SimDic$ 存在单词但是语义权重 SW_x 为 0 的情况，然后在算法第 8 行中，将该单词的 TW 值置为 0. 第 10 行得出的最终 Mashup 语义特征向量是由 $SimDic$ 中每个单词的词向量 $\boldsymbol{WVec}$ 与最终权重 TW 的乘积累加. 通过向量的信息表示 Mashup 服务可以更好地计算 Mashup 服务之间的关系.

3.3　基于密度峰值检测的 Mashup 服务聚类

在 DPC 算法[17]中，对于每个样本点 s 都有两个重要属性，分别是局部密度 ρ_s 和较高密度点的最近距离 δ_s，式(4)给出了 ρ_s 的计算公式：

$$\rho_s=\sum_{s\neq o}\left|\chi(d_{so}-d_c)\right. \tag{4}$$

其中，d_c 为截断距离，一般情况下由人工设定，d_{so} 表示其他样本点 o 到样本点 s 的距离. $\chi(d_{so}-d_c)$ 由 d_{so} 和 d_c 决定，若 $d_{so}-d_c>0$，则 $\chi(d_{so}-d_c)$ 的值为 0，反之 $\chi(d_{so}-d_c)$ 的值为 1. 在得到各个样本点的局部密度后，便可以得到属性 δ_s 的具体值：

$$\delta_s=\begin{cases}\min\limits_{o:\rho_o>\rho_s}(d_{so}), & \rho_s \text{ is not maximum}\\ \max\limits_{o}(d_{so}), & \rho_s \text{ is maximum}\end{cases} \tag{5}$$

DPC 算法的核心是筛选出 δ_s 与 ρ_s 都较高的样本点，以此作为聚类中心并进一步采取特定的聚类算法完成聚类. 数据规模较小时，DPC 算法可以通过 δ_s 与 ρ_s 绘制出决策图，直观地筛选出聚类中心. 以图 3(a)上分布的 28 个样本点为例，在通过式(4)和式(5)计算出所有样本点的 δ_s 与 ρ_s 后，进一步绘制出决策图(图 3(b))，可以直观地发现点 10 与点 1 适合作为聚类中心.

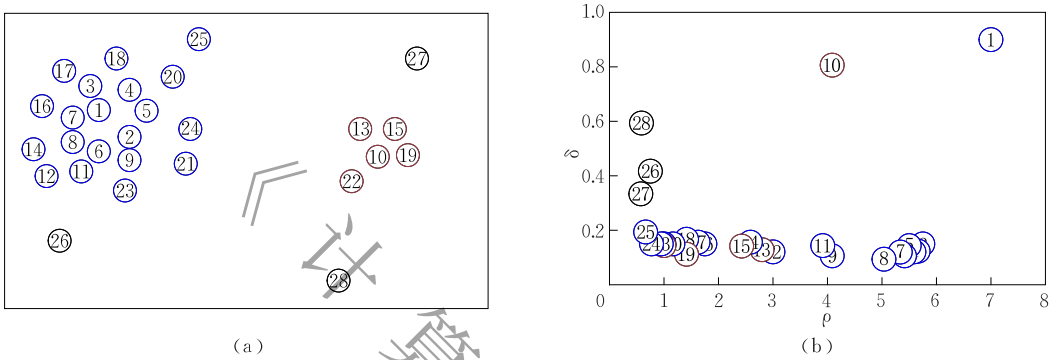


图 3　决策图样例[17]

DPC 算法的思想虽然便捷高效，但是仍然存一定的缺陷需要改善：(1) 截断距离 d_c 的设定极大程度影响着 DPC 算法聚类的结果；(2) 在样本数据达到一定量级后，直接挑选合适的聚类中心变得较为困难. 因此，本文受到研究[24-25]中启发，将其中的优化思路进行提炼后，结合 Mashup 服务聚类的研究场景，将局部密度的计算方式重新进行了定义，具体计算公式如下：

$$\rho_y=\sum_{z=1}^{k}\cos(DVec_y,DVec_z) \tag{6}$$

其中，***DVec*** 是 Mashup 服务在聚类场景中的一种表示模型，即为 3.2 节计算所得的 Mashup 语义特征向量，服务 y 的局部密度 ρ_y 的值为离其最近的 k 个 Mashup 服务特征向量的余弦相似度累加和. 这种改进方法解决了 DPC 算法难以确定截断距离进而导致局部密度取值不合理的问题. 同时，借助向量相似度信息，本文给出了一种新的向量距离计算方法，具体计算公式如下所示：

$$d_{yz}=1-\cos(DVec_y,DVec_z) \tag{7}$$

将式(5)与式(7)相结合，便可以得到各向量的 δ_y 属性，为方便表示，定义 ρ 与 δ 的乘积为 γ. 为进一步了解 DPC 算法中 δ 属性与 ρ 属性对决策的影响，本文选取数量不同的 Mashup 服务进行实验，计算最近距离 δ 和局部密度 ρ，并绘制决策图. 如图 4 所示，

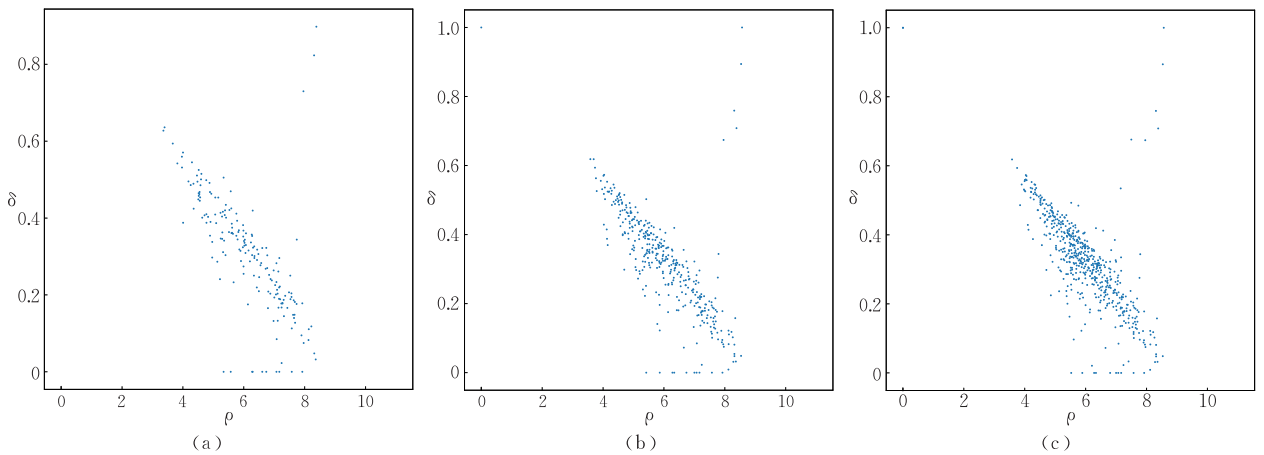


图 4　三种 Mashup 服务数量规模下的决策图对比

a、b、c 三图的服务数量规模分别是 200、400 与 600.

分析图 4 可知,三张决策图的右下方汇聚了大部分的 Mashup 服务点,其 δ 值较低,根据式(5)可知,这些点代表的向量基本不可能为聚类中心候选点. 当 δ 值增加时,会出现两类点:一类点 δ 值较高但 ρ 值较低,这些点多数为异常点,对聚类可能起到干扰作用,一般位于决策图中偏离红线中心的左上方处;另一类点 δ 值与 ρ 值均偏高,这类点是 DPC 算法所需要的点,基本都为质量较高的聚类中心候选点,位于决策图中偏离红线中心的右上方处. 虽然决策图能显示有效的聚类中心候选点,但是区分信息有限,无法在聚类中心候选点中给出进一步的判断.

因此,本文将研究焦点转向了 Mashup 语义特征向量的分布结构. 在对 Mashup 语义特征向量降维可视化处理后发现(图 5),将点分布均匀并密集的区域使用方框标注. 经计算分析可知,这些区域中向量的 δ 值与 ρ 值变化较小,且存在 γ 值较高的点,这表明该区域是一个密度峰值区域. 因此,将特征向量的分布结构与决策图分析相结合,可以进一步对聚类中心候选点进行筛选.

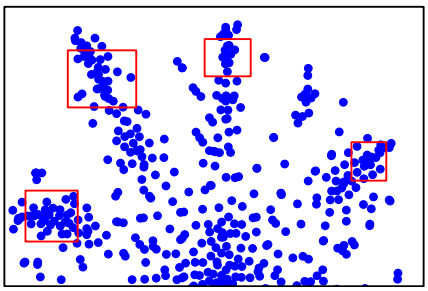


图 5 Mashup 语义特征向量局部分布图

基于上述分析结果,为自适应确定最优的 Mashup 服务初始聚类中心,本文提出了一种借助向量相似度来计算密度信息和检测聚类中心的算法 CCD-DI,并将其输出作为 K-means 的输入. CCD-DI 的核心流程如下所示:

(1)在所有 Mashup 语义特征向量中,识别并过滤出聚类中心候选点;

(2)遍历聚类中心候选点,提取出它们的密度信息,进行综合评估计算;

(3)基于综合评估计算的结果,从聚类中心候选点中筛选出最优的 K 个向量.

根据上述核心流程,算法 3 给出了基于 CCD-DI 的 Mashup 服务聚类算法.

算法 3. 基于CCD-DI的Mashup服务聚类算法.

输入:聚类划分个数 K,Mashup 服务的语义特征向量数组 $EVArray$,存放各向量的 δ 属性数组 $Array_\delta$ 和 ρ 属性数组 $Array_\rho$

输出:K 个 Mashup 服务簇

1. $\theta=(\max(Array_\delta)+\min(Array_\delta))/2$;
2. $Array_{tmp}=EVArray$ //将 $EVArray$ 备份至 $Array_{tmp}$
3. $Sarr_1=pickUnder(\theta,Array_\delta)$;
4. $fn=0$;
5. $cnt=$(number of unique item in $Sarr1$)$\times$(number of item in $Sarr1$) //cnt 为密度差值数量
6. For δ_x in $Sarr_1$
7. $fn+=getFn(\delta_x)$;
8. End For
9. $fn=fn/cnt$;
10. $r=\theta$, $lr_1=0$, $lr_2=fn$;
11. Loop If $\theta>lr_1$
12. $modifyData(r,lr_1,lr_2)$;
13. End Loop
14. $EVArray$.delete($\boldsymbol{DVec}$,r) //将普通点 st 与异常点 ab 从 $EVArray$ 中删除
//筛选 $EVArray$ 中 γ 最高的向量,并统计其半径 r 内包含的向量个数 m
15. $m=caculateVec(EVArray,Array_\rho,Array_\delta,r)$;
16. $Sarr_2=[]$;
17. For $\boldsymbol{DVec}$ in $EVArray$
18. $Sarr_2$.add($caltSD(\boldsymbol{DVec},Array_\rho,Array_\delta)$);
19. End For
20. $Sarr_3=$normalize($\boldsymbol{DVec}$,$Array_\rho$,$Array_\delta$,$Sarr_3$);
21. $center=sortData(\boldsymbol{DVec},K,Sarr_4)$;
22. $dataRes=$KMEANS($center$,$Array_{tmp}$);
23. Return $dataRes$;

算法 3 首先在第 1 行中设定属性 δ 的限定值 θ,以便快速确定普通点 st 的所在范围,并在第 3 行将 δ 值低于 θ 的普通点 st 存储到数组 $Sarr_1$ 中. 在 $Sarr_1$ 中,根据 δ 值在区间 $(0,\theta)$ 上的分布,将区间划分为 fn 个等长的区域. 其中算法第 4~9 行用于计算 fn. 随后,在算法 12 行中,$modifyData$ 函数处理划分出的等长区域,并寻找 δ 数量递增的区域. 循环最初设置 lr_1 为 0,lr_2 为 fn,即范围为群体区间集合,r 的默认值为 θ. 当 δ 数量增长时更新 lr_1 和 lr_2 的值,否则将 st 的判定半径 r 改为 lr_1. 在获取判定半径 r 后,从 $EVArray$ 中删除所有普通点 st 和异常点 ab,保留聚类中心候选点(算法第 14 行). 其中,普通点 st 的定义是 $\delta\leq r$ 的向量,而判定异常点 ab 的依据则是该点半径 r 内不包含其它的向量.

经过上述步骤后,$EVArray$ 只剩下聚类中心候选点集合 $center$. 为了分析 $center$ 的邻域信息,首先

在 $EVArray$ 中筛选出 ρ 与 δ 乘积 γ 最高的向量，并统计其半径 r 内包含的向量个数 m. 然后，遍历 $EVArray$ 中的每个聚类中心候选点 $center$，利用 $caltSD$ 函数计算 $center$ 的波动值 SD 并进行存储. 进一步，波动值 SD 的计算公式为

$$SD_y=\sqrt{\frac{1}{m}\sum_{z\in U(y)}(\boldsymbol{\gamma}_z-avg_z)^2} \tag{8}$$

其中，avg_z 为这 m 个向量 $\boldsymbol{\gamma}_z$ 的均值，$\boldsymbol{U}(\boldsymbol{y})$ 表示距离向量 $\boldsymbol{y}$ 最近的 m 个向量. 算法第 20 行采用 normalize 函数对 $EVArray$ 中每个向量的 SD 和 γ 进行加权评估计算($0\leq a\leq 1$，默认值为 0.5)：

$$score=a\times\left(1-\frac{(SD-\min(SD))}{\max(SD)-\min(SD)}\right)+(1-a)\times\frac{(\gamma-\min(\gamma))}{\max(\gamma)-\min(\gamma)} \tag{9}$$

上式中，$\min(SD)$ 和 $\max(SD)$ 表示统计 $EVArray$ 中向量所含 SD 的最小值与最大值，$\min(\gamma)$ 和 $\max(\gamma)$ 表示统计 $EVArray$ 中向量所含 γ 的最小值与最大值. 算法 3 第 21 行中，$sortData$ 函数为排序函数. 其输出的结果中前 K 个点即为候选聚类中心点，其中 K 值表示设定的 Mashup 服务类别数. 算法 3 的整体时间复杂度为 $O(n^2)$.

基于图 4(c)的决策图，当 $K=9$ 时，图 6 展示了算法 3 的聚类中心筛选过程. 图中含方向的箭头代表了算法在选择聚类中心时的顺序，特别是图中方框区域有多个合适的候选点时，算法借助 γ 值与 SD 值，能对最优聚类中心进行自适应判断.

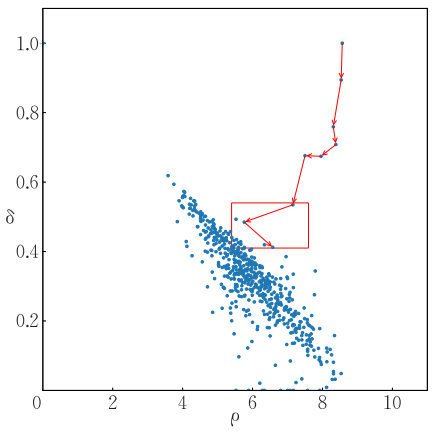


图 6　$K=9$ 时聚类中心筛选过程

4　实验与分析

为了对所提方法进行客观公正地评估与比较，本文首先从 ProgrammableWeb 平台上爬取了大量真实的 Mashup 服务数据与 Web API 数据，作为实验测试数据. 接着，通过对大量实验工作[5-10,23-31]的分析总结，设计了实验的具体实施方案，进行方法的验证与分析. 最后，我们将文中所涉及的数据、所比较的各类算法与实现以及本文方法的具体代码都进行了归纳整理，并上传至网络，以方便本领域相关研究人员进行参考比较，详情见附录表 1.

4.1　实验设置与数据集

实验所用数据均爬取于 ProgrammableWeb 平台，其中 Mashup 服务爬取 6217 条，Web API 爬取 11 930条(数据详见附录表 1-①). 实验在 Win10 系统环境下进行，内存为 16 GB，代码由 Python 实现.

利用 3.1 节描述的 Mashup 服务预处理规范，本节首先对服务数据进行预处理，以解决 Mashup 服务描述内容短、分类不规范和文本描述随意等问题，之后从处理过的 Mashup 服务中随机选择 1500 条进行人工分类. 考虑语句理解能力上的差异，为了提高人工分类结果的可靠性，人工分类执行两次，并分别安排不同的 10 名研究人员进行. 最终，在剔除功能描述过于模糊的服务后，使用 1346 条经过人工分类处理的 Mashup 服务进行聚类实验. 表 1 为用于实验的 Mashup 服务具体类别(实验数据详见附录表 1-②).

表 1　Mashup 服务类别

类别	数量	类别	数量
Search	253	Mapping	159
Travel	152	Video	106
Photos	103	Telephony	93
Weather	88	Music	87
Real Estate	85	eCommerce	82
Games	71	Messaging	67

为了获得效果良好的 Word2vec 词向量模型，本文将词向量设置为 300 维，并采用谷歌新闻数据语料进行训练. 经过预处理的服务数据已剔除大量的代词与专有名词，极大地减轻了未登录词带来的影响.

4.2　实验过程与结果分析

4.2.1　FSAC 方法验证与分析

本文在人工分类阶段人工提取了服务描述中的功能名词. 通过对功能名词的提取工作进行多次复检，尽可能提高功能名词质量. 但是由于多数情况下，功能名词间还存在近义词或同义词，所以 Mashup 服务的核心功能点通常会低于人工提取的功能名词数量.

图 7 绘制了人工提取的功能名词和服务类别间的关系图，从图中可以看出，尽管不同服务类别的功能名词在图中心存在交集，但仍有较多可反映自身功能特征的功能名词存在于图的边缘部分. 针对每

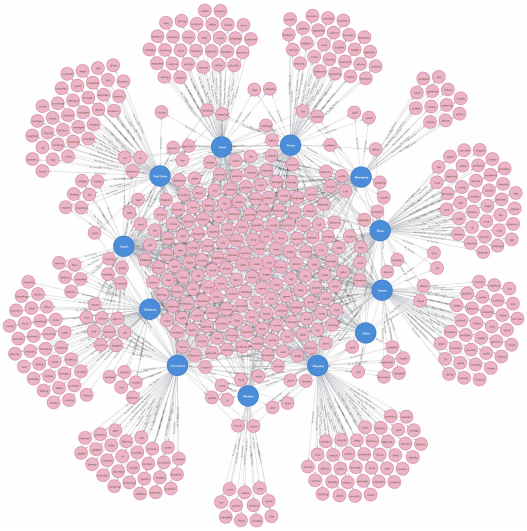


图 7 Mashup 服务类别与功能名词关系图

条 Mashup 服务，我们将基于人工提取的功能名词集 RN_m 与 FSAC 方法最终提取至 *SimDic* 的功能名词集 FN_m 进行比对（数据详见附录表 1-⑧）. 为了验证 FSAC 方法的有效性，本文利用匹配率 *Match* 来对上述两者进行比较. 匹配率 *Match* 定义为

$$Match(m)=\frac{|RN_m \cap FN_m|}{|RN_m|} \tag{10}$$

其中，$|FN_m|$ 为利用 FSAC 方法提取的功能名词数量，$|RN_m|$ 为人工提取的功能名词数量，$|RN_m \cap FN_m|$ 则表示两者的交集数量. 当 FN_m 中的名词多于 RN_m 中名词数量时，取 FN_m 中前 $|RN_m|$ 个 *TW* 权重最高的名词进行比对. FSAC 方法分别在 12 类服务数据上进行实验，方法获取的功能名词与人工分类的匹配结果如图 8 所示.

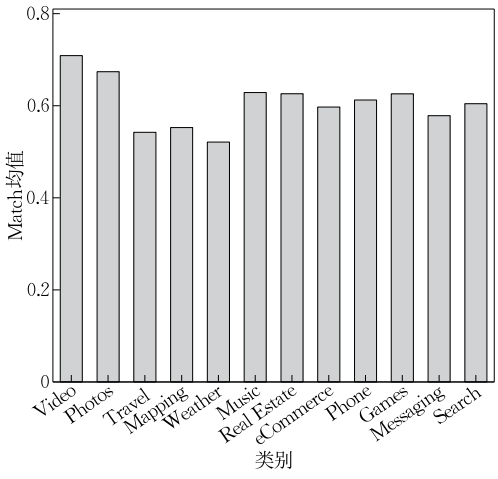


图 8 功能名词匹配结果

分析图 8 可得，每个类别的功能名词匹配率基本维持在 50%～70%. 这在一定程度上表明 FSAC 方法能通过语义关联计算有效提取名词集合.

4.2.2 聚类实验实施及结果

在 Mashup 服务聚类实验中，本文选用精准率、召回率、纯度与熵这四个指标对聚类方法的效果进行评估. 其中，人工分类的结果由 $RM=\{RM_1, RM_2, \cdots, RM_P\}$ 表示，实验聚类结果由 $EM=\{EM_1, EM_2, \cdots, EM_Q\}$ 表示. 精准率 *Precision* 与召回率 *Recall* 的定义如下所示：

$$Precision(EM_i)=\frac{|EM_i \cap RM_i|}{|EM_i|} \tag{11}$$

$$Recall(EM_i)=\frac{|EM_i \cap RM_i|}{|RM_i|} \tag{12}$$

人工分类中的 Mashup 服务数量用 $|RM_i|$ 表示，实验聚类结果的 Mashup 服务数量用 $|EM_i|$ 表示. *Precision* 与 *Recall* 由所有类别的精准率均值和召回率均值来表示，其值越高，表明聚类在精准性方面的表现越好. 此外，采用纯度 *Purity* 与熵 *Entropy* 对聚类的整体效果进行评估，式(13)到式(16)给出了相关定义.

$$Purity(EM_i)=\frac{\max(num_{ij})}{|EM_i|},\ 1\leq i\leq Q, 1\leq j\leq P \tag{13}$$

$$Purity(EM)=\sum_{i=1}^{Q}\frac{|EM_i|}{|SM|}Purity(EM_i) \tag{14}$$

$$Entropy(EM_i)=-\frac{1}{\log P}\sum_{j=1}^{P}\frac{num_{ij}}{|EM_i|}\log\left(\frac{num_{ij}}{|EM_i|}\right) \tag{15}$$

$$Entropy(EM)=\sum_{i=1}^{Q}\frac{|EM_i|}{|SM|}Entropy(EM_i) \tag{16}$$

SM 为所有服务的集合，num_{ij} 为人工分类与算法聚类结果交集中的服务数量. 好的聚类效果意味着更高的纯度与更低的熵. 之后，本文分别选取 7 种聚类方法以及本文方法（F+C+K）进行实验，并根据实验结果进行对比分析（实现代码详见附录表 1-④）.

(1) T+Q. 通过计算每个词的 TF-IDF 值，将 Mashup 服务表示为向量形式，之后使用 QT 算法聚类[32].

(2) T+K. 获取 Mashup 服务的 TF-IDF 特征向量，利用 *K*-means 算法对文本向量进行聚类[9].

(3) LDA. 通过 LDA 主题模型将 Mashup 服务集合转换为文档主题概率矩阵，聚类依据为矩阵中每个概率的最大分量[31].

(4) L+K. 通过 LDA 模型处理 Mashup 服务描述，将其表示为概率向量形式，并利用 *K*-means 算法进行聚类[8].

(5) F+K. 结合 3.2 节提出的 FSAC 方法提取出 Mashup 服务描述中的语义特征向量（MFSF 向

量)，然后使用 K-means 聚类.

(6) F+C+D. 该方法首先将 Mashup 服务描述转化为 MFSF 向量，之后利用 CCD-DI 方法获取最合适的 K 个文本向量作为聚类初始中心，以传统 DPC 算法实现聚类.

(7) R+C+K. 利用 Word2Vec 模型将人工提取的功能名词转化为 Mashup 服务的文本向量，并结合 K-means 聚类以 CCD-DI 方法检测出的最合适的 K 个 MFSF 向量作为初始聚类中心.

(8) F+C+K：融合了 FSAC 和 KCCD-DI 两种方法，并把检测出的 K 个文本向量作为初始中心进行 K-means 聚类.

4.2.3　聚类实验结果分析

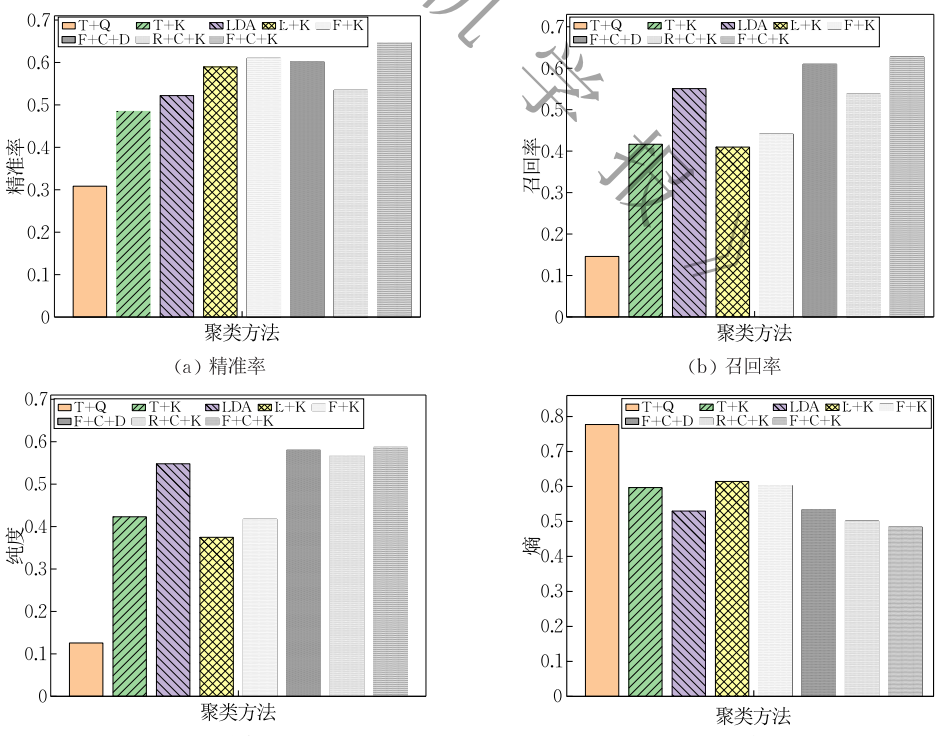
图 9 展示了上述 8 种聚类方法在 12 类(表 1)服务数据规模下，四种聚类评价指标的表现情况. 分析图中结果，可得出以下结论：

(1) 相比于其他方法，T+Q 方法与 T+K 方法的四种评估指标结果都较低. 这是由于服务描述文本内容一般都较短，且 TF-IDF 仅基于文档词频信息进行统计，并不能很好体现原服务描述的语义信息，进而造成较差的聚类效果.

(2) 由于 LDA 模型生成的主题概率分布能计算服务间的语义相似度[29]，因此，与 T+Q 与 T+K 方法相比，LDA 方法在四种评估指标上的表现都有所提升；此外，通过单独对比 L+K、F+K 两种方法可以发现，F+K 方法在四项评估指标上均略优于 L+K 方法，这主要是因为 F+K 利用了 MFSF 向量，而该向量来源于 FSAC 方法的计算结果，包含更能反映服务功能的特征信息，因此提升了聚类效果.

(3) 仔细观察 F+K 方法、L+K 方法在各项评估指标上的表现，可以发现，这两种方法的召回率相较于精准率都下降了约 30%左右. 通过对实验数据详细地分析后，我们认为 K-means 初始中心的选择策略对实验结果有较大影响. 传统的 K-means 方法随机进行初始中心的选取，往往只能获取局部最优解，从而使聚类结果与标准分类偏差较大.

(4) 对于 F+C+D、F+C+K、R+C+K 这三种方法，由于已经通过 CCD-DI 方法对聚类的初始中心进行了优化，因此在整体上可以保持较优的聚类效果. 而其中的 R+C+K 方法，由于只是简单将人工提取的功能名词转化为表示 Mashup 服务的特征向量，未能很好地考虑功能语义权重因素，因此，其整体聚类效果又会略低于 F+C+D 方法和 F+C+K 方法.



(a) 精准率　(b) 召回率　(c) 纯度　(d) 熵

图 9　聚类实验结果

此外，表 2 还给出了上述 8 种聚类方法在 12 类服务数据规模下的平均执行时间. 其中，T+Q 方法由于计算规模较为庞大，因此聚类执行时间最长，而 T+K、L+K、F+K 这三种方法是直接以 K-means 算法实现聚类，因此整体聚类时间偏低. 相比而言，F+C+D、R+C+K、F+C+K 三种方法由于都引

入了 CCD-DI 方法进行聚类中心的检测，所以，它们的平均执行时长相较于 T＋K、L＋K、F＋K 三种方法会偏高约 5 s 左右.

表 2 聚类方法平均执行时间

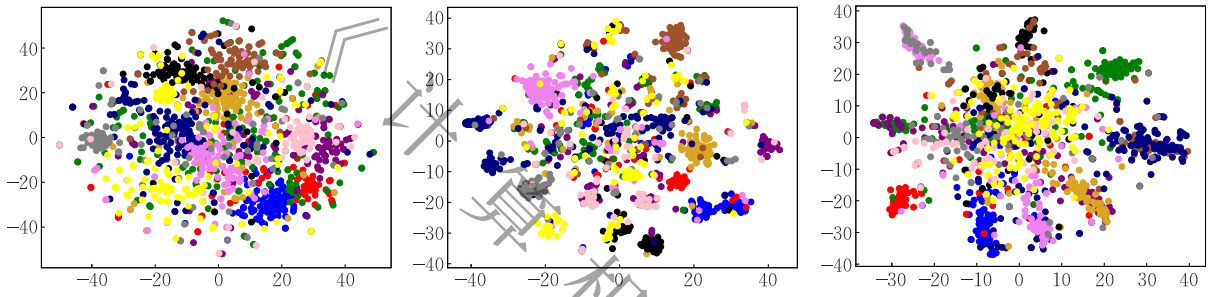
聚类方法	执行时间/s	聚类方法	执行时间/s
T＋Q	190.321	F＋K	0.023
T＋K	0.204	F＋C＋D	6.109
LDA	0.179	R＋C＋K	4.718
L＋K	0.099	F＋C＋K	4.431

在上述分析的基础上，本文利用 t-SNE 工具[33]对 12 类服务的 TF-IDF 特征向量、LDA 特征向量与 MFSF 向量进行了降维可视化展示，展示结果如图 10 所示. 从图 10 可知，TF-IDF 特征向量的分布效果最差，不同类重叠较大；相对而言，LDA 特征向量的分布相比 TF-IDF 更为鲜明，但错误分类的服务依旧较多；尽管 MFSF 向量分布的中心区域依旧有类别的重叠，但类别间的区分效果和前两种特征向量相比有了进一步提升，并且错误分类的服务数量有所减少. 借助图 10 的展示结果，可以从更直观的角度证明结论 1 和结论 2 的正确性(实现代码详见附录表 1-⑤).

为了进一步验证结论 3、结论 4 的观点，本文选取 T＋K、L＋K、F＋K、F＋C＋K 四种聚类方法，对服务数量最多的前 4～10 类服务进行了实验，并记录 5 次实验的运行结果(数据与实现详见附录表 1-⑥). 其中，分析指标采用 *F*-measure，其定义如下：

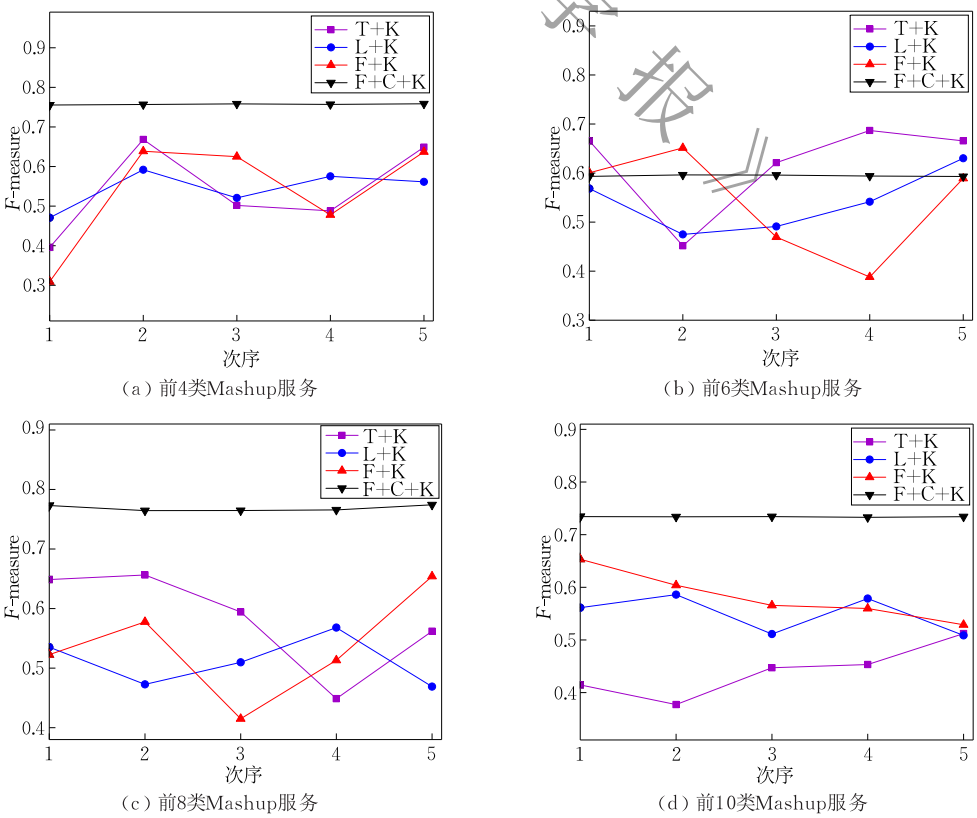
$$F\text{-measure}=\frac{2\times Recall\times Precision}{Recall+Precision} \tag{17}$$

如图 11 所示，T＋K、L＋K、F＋K 这三种方法



(a) TF-IDF特征向量的分布效果　(b) LDA特征向量的分布效果　(c) MFSF特征向量的分布效果

图 10 降维后的特征向量可视化结果



(a) 前4类Mashup服务　(b) 前6类Mashup服务

(c) 前8类Mashup服务　(d) 前10类Mashup服务

图 11 *F*-measure 指标测试

的结果具有不确定性. 这是因为传统的 K-means 算法的结果会随初始中心选取的不同而产生变化，导致聚类效果（F-measure 值）发生改变. 与之相比，本文的 F+C+K 方法，在各类的平均情况下，保持较优的 F-measure 值，获得更稳定的聚类效果.

4.2.4　参数 a 对聚类结果的影响

本文在 CCD-DI 方法阶段，借助式(9)来分析聚类中心候选点，并选取前 K 个 γ 值较高，SD 值较低的候选点. 我们利用对权重参数 a 的变量实验，来探讨 SD 值的影响，为 CCD-DI 方法的参数选择提供参考依据. 表 3 设置了 5 种服务类别分布情况，进行 F-measure 值的变化趋势对比（数据与实现详见附录表 1-⑦），其中权重参数 a 的取值范围为 0 至 1，实验结果如图 12 所示.

表 3　类别分布

名称	类别分布
C1	Video、Photos
C2	Video、Photos、Travel、Mapping
C3	Video、Photos、Travel、Mapping、Weather、Music
C4	Video、Photos、Travel、Mapping、Weather、Music、Real Estate、eCommerce
C5	Video、Photos、Travel、Mapping、Weather、Music、Real Estate、eCommerce、Phone、Games

图 12 中，F-measure 值在类别分布为 C1 时始终没有较大的波动. 分析原因，此时服务数目较少，且 Mashup 服务只有 2 个类别，所以无论采用 γ 值还是 SD 值来进行聚类中心的评估与选取，聚类结果都较好. 在 C2 的情况下，服务类别数目增加，当 $a=0.9$ 时，F-measure 出现大幅下跌，从较为稳定的 0.85 跌至 0.51. 经过分析，本文认为由于类别与服务数目的增加，仅靠 SD 值进行聚类中心评估，类与类之间重叠的部分，即向量密集区域，容易被错选为密度峰值区域，导致聚类中心选取偏差，从而降低了整体聚类效果.

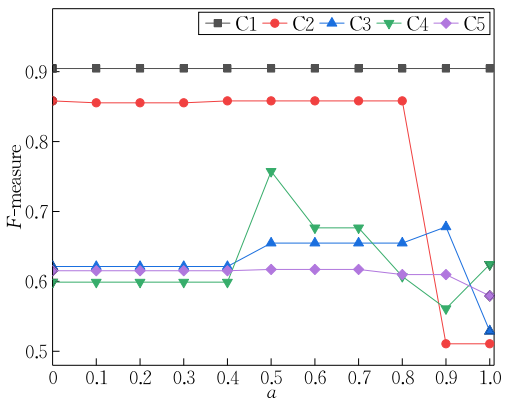


图 12　权重参数 a 对聚类结果的影响

基于此分析，本文又在 C3、C4 与 C5 的分类条件下进行实验，可以发现，a 的值为 1 或接近于 1 时，F-measure 往往较低，聚类效果差. 该结果表明，单单利用 SD 值进行聚类中心评估的方法，难以得到较好的聚类效果. 此外，在图 12 中还可以发现，当服务数量较大时，即 C3、C4、C5 这些实验场景，仅仅依靠降低权值 a，增大 γ 值来影响聚类中心的评估，对聚类效果的提升并不大. 而 a 取值的最佳范围约为 0.5～0.7，在该范围内，聚类效果最佳. 这个结果表示在服务数量较多的情况下，需要综合考量 SD 值与 γ 值来进行聚类中心的筛选，以此提升聚类效果.

5　总结与展望

为了能够有效提升 Mashup 服务聚类的精度，本文提出一种融合功能语义关联计算与密度峰值检测的 Mashup 服务聚类方法. 该方法主要分为两步进行：(1) 将爬取到的真实 Mashup 数据进行预处理，接着利用 FSAC 方法对服务描述中的功能名词进行精准定位以及语义关联计算. 在此基础上，结合 Word2Vec 模型构造出能够体现 Mashup 服务功能特征的语义特征向量；(2) 在得到每个 Mashup 服务的语义特征向量后，借助 CCD-DI 方法对 Mashup 服务描述特征向量进行筛选，选取 K 个向量作为聚类初始中心，最后利用 K-means 算法实现聚类. 基于 ProgrammableWeb 的真实数据实验表明，本文所提聚类方法在精准率、召回率、纯度、熵等指标上均有良好表现.

在下一阶段，本文的研究工作主要包括三个方面：(1) 在 Mashup 语义特征向量模型中融入动词、名词的组合信息，进行优化；(2) 通过结合 Mashup 服务的结构、关系特征，来对服务相似性计算方法进行进一步改良；(3) 将所提 Mashup 服务聚类方法与 Web API 推荐工作相结合，简化 Web API 推荐流程，提升 Web API 的推荐精度.

参　考　文　献

[1] Xia B, Fan Y, Tan W, et al. Category-aware API clustering and distributed recommendation for automatic Mashup creation. IEEE Transactions on Services Computing, 2014,

8(5): 674-687

[2] Benslimane D, Dustdar S, Sheth A. Services Mashups: The new generation of Web applications. IEEE Internet Computing, 2008, 12(5): 13-15

[3] Chen L, Hu L, Zheng Z, et al. WTCluster: Utilizing tags for Web services clustering//Proceedings of the International Conference on Service-Oriented Computing. Paphos, Cyprus, 2011: 204-218

[4] Wu J, Chen L, Zheng Z, et al. Clustering Web services to facilitate service discovery. Knowledge and Information Systems, 2014, 38(1): 207-229

[5] Cao B, Liu X, Rahman M D M, et al. Integrated content and network-based service clustering and Web APIs recommendation for Mashup development. IEEE Transactions on Services Computing, 2017, 13(1): 99-113

[6] Rahman M M, Liu X, Cao B. Web API recommendation for Mashup development using matrix factorization on integrated content and network-based service clustering//Proceedings of the 2017 IEEE International Conference on Services Computing (SCC). Honolulu, USA, 2017: 225-232

[7] Shi M, Liu J, Zhou D, et al. WE-LDA: A word embeddings augmented LDA model for Web services clustering//Proceedings of the 2017 IEEE International Conference on Web Services (ICWS). Honolulu, USA, 2017: 9-16

[8] Cao B, Liu X, Li B, et al. Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model//Proceedings of the 2016 IEEE International Conference on Web Services (ICWS). San Francisco, USA, 2016: 212-219

[9] Gao W, Chen L, Wu J, et al. Manifold-learning based API recommendation for Mashup creation//Proceedings of the 2015 IEEE International Conference on Web Services. New York, USA, 2015: 432-439

[10] Liu Y, Li L, Xiang J. Using clustering labels to supervise Mashup service classification//Proceedings of the International Conference on Conceptual Modeling. Xi'an, China, 2018: 35-38

[11] Pan W, Chai C. Structure-aware Mashup service Clustering for cloud-based Internet of Things using genetic algorithm based clustering algorithm. Future Generation Computer Systems, 2018, 87: 267-277

[12] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003, 3: 993-1022

[13] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988, 24(5): 513-523

[14] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013

[15] Cao Bu-Qing, Xiao Qiao-Xiang, Zhang Xiang-Ping, et al. An API service recommendation method via combining self-organization map-based functionality clustering and deep factorization machine-based quality prediction. Chinese Journal of Computers, 2019, 42(6): 1367-1383(in Chinese) (曹步清, 肖巧翔, 张祥平等. 融合 SOM 功能聚类与 DeepFM 质量预测的 API 服务推荐方法. 计算机学报, 2019, 42(6): 1367-1383)

[16] MacQueen J. Some methods for classification and analysis of multivariate observations//Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, USA, 1967, 1(14): 281-297

[17] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492-1496

[18] Surianarayanan C, Ganapathy G. An approach to computation of similarity, inter-cluster distance and selection of threshold for service discovery using clusters. IEEE Transactions on Services Computing, 2015, 9(4): 524-536

[19] Elgazzar K, Hassan A E, Martin P. Clustering WSDL documents to bootstrap the discovery of Web services//Proceedings of the 2010 IEEE International Conference on Web Services. Miami, USA, 2010: 147-154

[20] Cao Y, Liu J, Shi M, et al. Relationship network augmented Web services clustering//Proceedings of the 2019 IEEE International Conference on Web Services (ICWS). Milan, Italy, 2019: 247-254

[21] Zou G, Qin Z, He Q, et al. DeepWSC: A novel framework with deep neural network for Web service clustering//Proceedings of the 2019 IEEE International Conference on Web Services (ICWS). Milan, Italy, 2019: 434-436

[22] Wan Y, Chen L, Xu G, et al. SCSMiner: Mining social coding sites for software developer recommendation with relevance propagation. World Wide Web, 2018, 21(6): 1523-1543

[23] Sun L, Liu R, Xu J, et al. An adaptive density peaks clustering method with fisher linear discriminant. IEEE Access, 2019, 7: 72936-72955

[24] Liu R, Wang H, Yu X. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Information Sciences, 2018, 450: 200-226

[25] Liu R, Huang W, Fei Z, et al. Constraint-based clustering by fast search and find of density peaks. Neurocomputing, 2019, 330: 223-237

[26] Ruan S, Mehmood R, Daud A, et al. An adaptive method for clustering by fast search-and-find of density peaks: Adaptive-DP

//Proceedings of the 26th International Conference on World Wide Web Companion//Proceedings of the International World Wide Web Conferences Steering Committee. Perth, Australia, 2017: 119-127

[27] Parmar M, Wang D, Zhang X, et al. REDPC: A residual error-based density peak clustering algorithm. Neurocomputing, 2019, 348: 82-96

[28] Tang M, Xia Y, Tang B, et al. Mining collaboration patterns between APIs for Mashup creation in Web of things. IEEE Access, 2019, 7: 14206-14215

[29] Zhao J S, Zhu Q M, Zhou G D, Zhang L. Review of research in automatic keyword extraction. Journal of Software, 2017, 28(9): 2431-2449(in Chinese)
(赵京胜，朱巧明，周国栋等. 自动关键词抽取研究综述. 软件学报，2017，28(9)：2431-2449)

[30] Jiang Bo, Ye Ling-Yao, Pan Wei-Feng, et al. Service clustering based on the functional semantics of requirements. Chinese Journal of Computers, 2018, 41(6): 1255-1266(in Chinese)
(姜波，叶灵耀，潘伟丰等. 基于需求功能语义的服务聚类方法. 计算机学报，2018，41(6)：1255-1266)

[31] Shi Min, Liu Jian-Xun, Zhou Dong, et al. Multi-relational topic model-based approach for Web services clustering. Chinese Journal of Computers, 2019, 42(4): 820-836(in Chinese)
(石敏，刘建勋，周栋等. 基于多重关系主题模型的 Web 服务聚类方法. 计算机学报，2019，42(4)：820-836)

[32] Heyer L J, Kruglyak S, Yooseph S. Exploring expression data: Identification and analysis of coexpressed genes. Genome Research, 1999, 9(11): 1106-1115

[33] Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research, 2008, 9(9): 2579-2605

附　录.

附表 1　实验数据类别分布

序号	说明	下载地址
①	爬取于 ProgrammableWeb 平台的数据，包括 Mashup 服务 6217 条，Web API 11930 条	https://github.com/viivan/Mashup-and-Web-Api-data1
②	1346 条进行过人工分类和复检的 Mashup 服务实验集	https://github.com/viivan/Mashup-data2
③	12 类 Mashup 服务中人工提取的功能名词集和 FSAC 方法提取的功能名词集	https://github.com/viivan/Mashup-FSAC
④	T+Q、T+K、LDA、L+K、F+K、F+C+K、F+C+D 等算法代码	https://github.com/viivan/Mashup-service-clustering-algorithms
⑤	12 类 Mashup 服务降维可视化数据及代码	https://github.com/viivan/Mashup-service-feature-vector-visualization
⑥	4.2.3 节聚类方法性能测试数据及代码	https://github.com/viivan/Mashup-clustering-methods-Performance-test
⑦	4.2.4 节实验数据及代码	https://github.com/viivan/Mashup-weight-parameter-Influence-test

LU Jia-Wei, M. S., lecturer. His current research interests include service computing, software architecture, and big data visualization.

WU Han, M. S. candidate. His research interest is service computing.

ZHANG Yuan-Ming, Ph. D., associate professor. His current research interests include service computing and big data.

LIANG Qian-Hui, Ph. D., researcher. Her current research interest is data science.

XIAO Gang, Ph. D., professor. His current research interests include intelligent manufacturing and cloud manufacturing.

Background

Service-oriented Architecture (SOA) has become a driving force for Web applications development. Service-oriented Computing (SOC) is a computing paradigm that is driven by SOA. Recently, Mashup technology has emerged, which is a web page, or web application, which uses content from more than one source to create a single new service displayed in a single interface. With the rapid growth in the number and type of Mashup service on the Internet, how to quickly and