软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007185]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

融合潜在联合词与异质关联兼容的 Web API 推荐*

胡 强[1,2], 綦浩泉[1], 李浩杰[1], 杜军威[1]

[1](青岛科技大学 信息科学技术学院, 山东 青岛 266061)

[2](云南省服务计算重点实验室, 云南 昆明 650221)

通信作者: 杜军威, E-mail: dujunwei@qust.edu.cn

摘 要: 服务描述中包含的应用场景信息有限, 使得以功能相似度计算为主的 Mashup 服务组件 Web API 推荐与需求预期常存在差异, 功能匹配精确度有待进一步提高. 部分研究者虽利用 Web API 的协作关联提升推荐兼容性, 但忽视了功能关联对 Mashup 服务创建的负反馈影响, 从而限制了推荐多样性的提升. 为此, 提出一种融合潜在联合词与异质关联兼容的 Mashup 服务的组件 Web API 推荐方法. 该方法为 Mashup 需求和 Web API 提取潜在应用场景联合词并融入到功能向量的生成中, 进而提高二者功能相似度的匹配精确度, 以获得高质量的候选组件 Web API 集合. 将功能关联与协作关联建模为异质服务关联, 并利用异质关联兼容替代传统方法中的协作兼容, 以提升 Web API 的推荐多样性. 相较于对比方法, 所提方法在评价指标 *Recall*、*Precision* 和 *NCDG* 上分别提升了 4.17%–16.05%, 4.46%–16.62% 与 5.57%–17.26%, 多样性指标 *ILS* 降低了 8.22%–15.23%. 冷启动 Web API 推荐的 *Recall* 与 *Precision* 指标值分别为非冷启动 Web API 推荐的 47.71% 和 46.58%. 实验表明所提方法不仅提升了 Web API 推荐质量, 而且对冷启动 Web API 具有很好的推荐效果.

关键词: Mashup 服务; 异质关联; Web API 推荐; 多样性

中图法分类号: TP311

中文引用格式: 胡强, 綦浩泉, 李浩杰, 杜军威. 融合潜在联合词与异质关联兼容的Web API推荐. 软件学报. http://www.jos.org.cn/1000-9825/7185.htm

英文引用格式: Hu Q, Qi HQ, Li HJ, Du JW. Web API Recommendation by Fusing Latent Related Words and Heterogeneous Association Compatibility. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7185.htm

Web API Recommendation by Fusing Latent Related Words and Heterogeneous Association Compatibility

HU Qiang[1,2], QI Hao-Quan[1], LI Hao-Jie[1], DU Jun-Wei[1]

[1](College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China)

[2](Yunnan Key Laboratory of Service Computing, Kunming 650221, China)

Abstract: The service descriptions provide limited information about application scenarios, creating a gap between Mashup service component Web API recommendations based on functional similarity calculation and desired expectations. Consequently, there is a need to enhance the accuracy of function matching. While some researchers utilize collaborative associations among Web APIs to enhance recommendation compatibility, they overlook the adverse effects of functional associations on Mashup service creation, thereby limiting the enhancement of recommendation diversity. To address this issue, this study proposes a Web API recommendation method for Mashup service components that integrates latent related words and heterogeneous association compatibility. The study extracts latent related words associated with application scenarios for both Mashup requirements and Web APIs, integrating them into the generation of function vectors. By enhancing the accuracy of functional similarity matching, it obtains a high-quality candidate set of Web API components.

* 基金项目: 国家自然科学基金 (61973180); 国家重点研发计划 (2023YFF0612100); 山东省自然科学基金 (ZR2021MF092); 山东省重点研发计划 (软科学)(2023RKY01009); 云南省服务计算重点实验室开放课题 (YNSC23116)
收稿时间: 2023-09-17; 修改时间: 2023-11-30; 采用时间: 2024-03-27; jos 在线出版时间: 2024-09-11

Function association and collaboration association are modeled as heterogeneous service association. The study utilizes heterogeneous association compatibility to replace collaboration compatibility in traditional methods, thus enhancing the recommendation diversity of Web APIs. In comparison, the proposed approach demonstrates improvements in evaluation indicators, with *Recall*, *Precision*, and *NCDG* enhanced by 4.17% to 16.05%, 4.46% to 16.62%, and 5.57% to 17.26%, respectively. Additionally, the diversity index *ILS* is reduced by 8.22% to 15.23%. The *Recall* and *Precision* values for cold-start Web API recommendation are 47.71% and 46.58% of those for non-cold-start Web API recommendation, respectively. Experimental results demonstrate that the proposed method not only enhances the quality of Web API recommendation but also yields favorable results for cold-start Web API recommendations.

Key words: Mashup service; heterogeneous association; Web API recommendation; diversity

Web API 是一种通过网络调用的应用程序接口, 可用于构建各类业务系统[1]. 为便于调用和集成, Web API 功能粒度通常较小, 单个 Web API 难以完成复杂的业务需求. Mashup 服务将多个 Web API 混搭形成新的增值服务, 可以整合更多的数据资源, 提供丰富的业务功能[2]. Mashup 服务中包含的 Web API 称为其组件 Web API (或组件服务). Mashup 服务的组件 Web API 推荐是指在已有 Web API 中, 为 Mashup 服务需求寻求合适的组件服务, 已经成为当前的热点研究问题[3].

网络中发布的大量 Web API 为 Mashup 服务提供了丰富的组件服务, 同时也增加了选择难度. 在为 Mashup 服务需求寻找组件服务时, 需提供所构建 Mashup 服务的功能关键词或者描述文本, 通过对功能关键词进行语义解析, 或者采用主题模型、神经网络模型等对功能描述文本进行特征提取, 将提取的 Mashup 服务需求的功能特征与已有 Web API 进行匹配, 获得在功能上满足 Mashup 服务需求的候选 Web API 集合[4].

部分研究者将协同过滤思想引入到组件服务的筛选. 寻找与 Mashup 服务需求相似的 Mashup 服务, 将这些 Mashup 服务的组件 Web API 作为候选服务, 开展功能和非功能匹配以便精准地推荐组件 Web API. 上述做法利用已有 Mashup 服务与 Web API 的历史调用关系以及 Web API 流行度等因素来提升推荐合理性. 然而, 在寻找与 Mashup 服务需求相似的 Mashup 服务时, 仍然以功能关键词或描述文本作为依据, 这些关键词和文本描述了服务的功能, 但在应用场景层面提供的信息有限. 因此, 难以挖掘 Mashup 服务与组成 Web API 之间隐含的场景交互信息, 从而影响了协同过滤的效果.

图 1 中的 Mashup 服务 Cool Bars, Restaurants and Clubs 是一个休闲社交 API. 使用者利用该服务可以查询附近的餐馆、酒吧和俱乐部等休闲场所, 并即时分享社交状态与照片. 它由 Google Maps 与 Twitter 两个 Web API 组成, 二者分别负责场馆位置定位与社交信息展示.

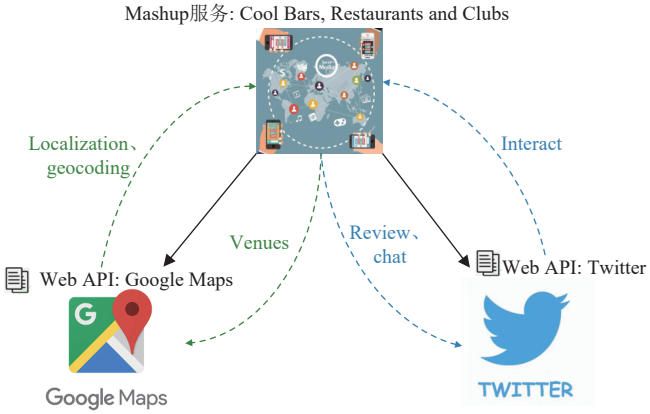

图 1　潜在联合词示例

在 Cool Bars, Restaurants and Clubs 服务的描述中, 存在 venues、review 和 chat 等词语, 这些词语不仅表达了 Mashup 服务的功能, 也是其组件 Web API 的应用场景特征词, 其中 venues 对应 Google Maps 的应用场景, review 和 chat 对应 Twitter 的应用场景. 同样, 从 Google Maps 与 Twitter 可以分别提取出表达 Mashup 服务功能场景的

词语 localization、geocoding 和 interact. 本文将上述表示应用场景的特征词称为潜在联合词, 如果为 Web API 和 Mashup 服务需求, 按照规则分别匹配潜在联合词, 然后进行功能匹配, 将有效提升 Mashup 服务需求的候选组件服务的生成质量.

获取候选 Web API 集合后, 组件服务之间的协作兼容性常用来决定最终推荐的 Web API[5]. 在评估组件服务的协作兼容时, 有研究者将 Web API 之间的输入输出参数的匹配作为依据[6]. 更多研究工作是构建 Web API 协作图, 将协作图中的节点向量化, 通过计算协作图中的节点相似度实现服务协作兼容性的度量. 也有研究工作利用子图的结构特性 (如文献 [7] 构建的最小斯坦纳树) 来搜寻协作兼容性最大的 Web API 组合. 协作兼容性的引入, 有效地提升了组件 Web API 的推荐质量.

然而, 现有以协作兼容性辅助 Web API 推荐的方法中主要考虑 Web API 之间协作关联, 通常认为协作次数多的 Web API 之间兼容性高, 优先被推荐用于构建 Mashup 服务. 在推荐的组件 Web API 中, 存在着较多具有功能关联的服务 (功能类似或部分重复 Web API), 这与“组件 Web API 之间的功能为互补关系”这一普遍共识存在差异.

组件 Web API 之间的协作次数越多、功能重复越少, 则意味着推荐质量越高. 对于两个组件 Web API, 它们之间的协作关联和功能关联对于二者共同出现在推荐列表分别起着正反馈和负反馈的作用. 现有单纯以协作兼容性辅助组件 Web API 推荐的方法, 以组件服务的协作强度最大化为目标, 忽视了功能关联的负反馈作用, 造成推荐服务中存在较多的功能重复的 Web API, 推荐多样性差. 若将 Web API 之间的协作关联和功能关联融合为一种新的关联, 基于这种新的关联兼容将有效提升组件 Web API 的推荐质量.

已有方法在 Mashup 服务需求与组件 Web API 进行功能匹配时缺乏对应用场景信息的考量, 功能匹配精确度有待于进一步提升. 同时, 以协作为主的 Web API 兼容性判定未考虑组件服务之间功能重复性, 造成推荐的多样性有待. 为此, 提出一种融合潜在联合词与异质关联兼容的组件 Web API 推荐方法, 主要工作和贡献如下.

(1) 提出一种用于描述 Mashup 服务与 Web API 应用场景信息的潜在联合词提取方法. 融合潜在联合词后的服务功能向量能够有效提升 Web API 与 Mashup 服务需求的功能匹配精确度, 提高候选组件 Web API 集合的求解质量.

(2) 面向协作关联和功能关联构建了 Web API 的异质关联图, 设计了面向服务关联强度的随机游走策略, 利用 GATNE 模型为 Web API 生成了关联向量. 利用异质关联兼容替代传统方法中的协作兼容, 提升了 Web API 的推荐多样性.

(3) 建立了一种融合功能匹配度与关联兼容度的组件 Web API 推荐方法. 引入注意力机制优化重组 Mashup 服务需求向量与候选 Web API 的关联向量, 与 Web API 功能向量一起输入全连接层, 根据匹配度的输出概率实现 Top-*K* 组件 Web API 推荐. 实验表明所提出方法的 Web API 推荐质量优于对比方法.

本文第 1 节介绍 Mashup 服务的组件 Web API 推荐研究现状. 第 2 节介绍与本文方法相关的定义和研究框架. 第 3 节给出 Mashup 服务需求的组件 Web API 推荐方法. 第 4 节开展实验, 验证本文所提出方法的效果. 第 5 节对全文进行总结与展望.

1 相关工作

近年来, Mashup 服务的组件 Web API 推荐受到广泛关注. 研究者从功能匹配、服务协作以及历史服务调用等多个角度构建了各类推荐系统, 相关推荐方法主要可以划分为基于内容匹配、基于协同过滤和基于神经网络模型这 3 类.

基于内容匹配的方法主要依据 Web API 与 Mashup 服务需求描述的相似度推荐组件 Web API. 例如, Gu 等人[8]将 Mashup 服务需求划分为表示不同功能的短句, 通过计算短句与 Web API 描述之间的相似度, 预测不同 Web API 相对需求的评分, 选取评分最高的 Web API 作为推荐服务. Li 等人[9]使用关系主题模型对 Mashup 服务需求和 Web API 的标签和主题信息建模以计算二者的相似度, 同时融合因子分解机获取多个维度的信息, 以推荐 Mashup 服务需求对应的 Web API. Zhang 等人[10]将关键词视为服务功能载体, 从服务描述中提取功能关键词, 并

通过扩展关键词的同义词来提升 Mashup 服务需求与 Web API 的功能匹配度. Tang 等人[11]通过 TF-IDF 技术挖掘 Web API 功能描述中的关键词, 以此扩展标签集并挖掘 Web API 之间的标签关联, 然后基于标签关联相似度推荐 Web API.

基于协同过滤思想的方法, 通常使用矩阵分解与因子分解机提高 Web API 的推荐质量. 在使用矩阵分解的方法中, Su 等人[12]使用 Word2Vec 技术将上下文信息整合到一个概率分解矩阵中, 并将 Web API 和 Mashup 服务描述的上下文关系均融入到推荐模型, 取得了优异的推荐效果. 类似的, Botangen 等人[13]构建了一种基于概率矩阵分解的推荐方法, 在 Mashup-API 交互的偏好度派生中考虑了地理位置信息. 实验表明利用地理位置信息增强隐式数据可以提高组件 Web API 推荐的精度. Yao 等人[14]提出了一种带有隐式相关正则化的概率矩阵分解方法来解决 Mashup 服务的 Web API 推荐问题. 通过分析 Web API 的共同调用模式获取它们的潜在相关性, 增强推荐的多样性.

在使用因子分解机的方法中, Tang 等人[15]提出了一种基于组合模式的 Web API 推荐方法, 该方法基于 Web API 与 Mashup 之间的调用模式以及 Web API 的共现性, 构建多维特征矩阵, 通过深度分解机模型学习 Web API 与 Mashup 服务之间潜在的链接关系, 为目标 Mashup 服务需求推荐 Top-*K* 最佳 Web API. Cao 等人[16]使用关系主题模型提取 Mashup 服务、Web API 间的调用关系, 利用因子分解机训练潜在主题, 预测 Mashup 和 Web API 之间的链接关系, 为目标 Mashup 服务需求推荐 Top-*K* 相关的 Web API. Nguyen 等人[17]提出带有注意力机制的矩阵分解机, 将注意力分数和 Mashup-API 上下文相似度注入到矩阵分解结构中进行训练, 利用 Web API 描述和协作历史中其他 Web API 的关系作为正则项, 学习潜在特征的重要性并提高 Web API 的预测性能. 以上基于协同过滤思想的方法局限于近邻 Mashup 服务匹配与直观调用记录, 忽视了 Mashup 服务需求与组件 Web API 之间的潜在应用场景交互特征, 导致推荐出的 Web API 与开发者的需求产生差异.

当前, 神经网络逐渐成为 Mashup 服务组件 Web API 推荐的热点模型. 例如, Kang 等人[18]提出了一种具有新型神经网络架构的混合因子分解机模型, 该模型集成了深度神经网络来捕获非线性特征交互, 并使用注意力机制捕获 Mashup 服务与 Web API 交互特征的不同重要性, 提高 Web API 推荐准确性. Xiao 等人[19]提出了一种基于结构增强和属性弱化网络的 Mashup 服务的组成服务推荐方法. 在特征提取层中, 从 Web API 关系网络图中捕获结构关系和属性信息, 获得每个 API 对应的表示向量, 在匹配演化层来捕捉 API 之间的协作关系, 并增量地选择 Web API 实现推荐. Liu 等人[20]将 Web API 和 Mashup 服务的功能描述作为输入, 提出一个具有 3 个线性层的微小模型 T2L2. 前两个线性层用于对齐 Web API 和 Mashup 服务的表示空间, 最后一个线性层用于计算 Web API 和 Mashup 服务的匹配度, 显著降低了模型复杂性和所需数据, 同时提高了推荐准确率. Ma 等人[21]分别提取需求与 Mashup 服务、需求与 Web API、Web API 之间的交互特征信息, 构建了一个合并上述 3 种类型交互信息的深度神经网络, 为每个候选 Web API 输出相对于 Mashup 服务需求的评分, 实现 Web API 的推荐.

Yan 等人[22]构建了一种基于深度学习的服务推荐框架, 将 Web API 协作信息融合到 Web API 向量嵌入中, 推荐质量得到显著改善. Cao 等人[23]提出一种基于图注意力表示和 DeepFM 质量预测 Mashup 组件 Web API 推荐方法. 利用 Web API 组合关系构建网络, 计算相邻节点的注意力系数, 根据相邻节点特征的重要性加权生成 Web API 的节点表征. 使用 DeepFM 对特征之间的交互关系进行建模, 通过对 Web API 的调用分数排名, 实现 Web API 推荐. Huang 等人[24]将 Mashup 服务和 Web API 作为网络节点, 调用关系作为边. 将节点的文本特征向量与其他非功能特征相结合, 使用 GAT 捕获相邻节点的不同权重贡献, 混合节点信息输出新的节点特征, 通过链路预测为 Mashup 服务推荐合适的 Web API.

此外, 研究者也尝试将 Web API 推荐转化为多目标优化或图搜索问题. 例如, Almarimi 等人[25]将 Mashup 服务的组成 Web API 集合求解转化为多目标优化组合问题, 以 Web API 历史共同使用、Web API 功能与 Mashup 服务需求的匹配和 Web API 功能多样性为目标, 使用 NSGA-II 搜索最优服务集. Gong 等人[26]将 Mashup 服务的 Web API 推荐建模为一个图搜索问题, 通过在 Web API 的协作关联图中找到最小斯坦纳树来解决 Web API 的推荐问题.

以上方法虽然类别众多, 其相同之处在于均仍以 Mashup 服务与 Web API 的功能相似和二者的调用或协作关系作为主要的推荐依据. 上述工作的不同在于通过改进和设计各类模型, 提高服务功能特征的提取质量, 或者在推

荐过程中融入不同的辅助信息, 如服务之间的协作、调用、位置等要素. 这些方法为研究 Mashup 服务需求的组件 Web API 推荐提供了有益的参考. 然而, 在构建 Mashup 服务时, Web API 与需求之间不仅需要功能匹配, 更需要应用场景的契合. 此外, 现有方法缺乏对推荐服务多样性的考量. 为此, 本文在构建 Web API 推荐方法时, 将应用场景信息融入到功能匹配中, 同时, 利用异质关联图将 Web API 的功能关联与协作关联进行融合度量, 增加推荐的多样性.

2 预备知识与研究框架

2.1 相关定义

本节给出 Web 服务、Mashup 服务以及异质服务关联的相关定义, 并介绍研究框架.

定义 1. Web API. Web API 为一个三元组 $s = (n, L, d)$, 其中, n 为 Web API 的名称, L 为 Web API 的标签集合, d 为 Web API 的描述信息.

定义 2. Mashup 服务. Mashup 服务为一个四元组 $m = (n, L, d, S)$. 其中, n 为 Mashup 服务的名称, L 为 Mashup 服务的标签集合, d 为 Mashup 服务的描述信息, S 为组件 Web API 集合.

图 2 和图 3 分别是 ProgrammableWeb 上发布的 Web API 和 Mashup 服务示例. 二者均具有服务名称、标签、服务功能描述. 此外, Mashup 服务还包含一组相关 Web API 集合.

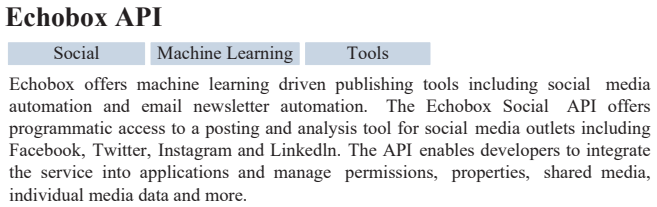

图 2 Web API 示例

图 3 Mashup 服务示例

定义 3. 功能关联. 若存在标签 l, 使得 $l \in s_i.L \cap s_j.L$, 则称服务 s_i 与 s_j 存在功能关联, 记作: $s_i \sim s_j$.

定义 4. 协作关联. 存在 Mashup 服务 m, 使得 $s_i \in m.S \wedge s_j. \in m.S$, 则称服务 s_i 与 s_j 存在协作关联, 记作: $s_i \leftrightarrow s_j$.

定义 5. Web API 异质关联图. Web API 异质关联图为一个无向加权图 $HAG = (V, E, W)$, 其中,

(1) $V = \{v_1, v_2, \ldots, v_n\}$为节点集合, 节点 v_i 表示 Web API s_i.

(2) $E = \{Ef, Ec\}$, Ef 和 Ec 分别为功能关联边集合与协作关联边集合.

1) $e = (v_i, v_j) \in Ef$ 中的节点 v_i 与 v_j 所对应的 s_i 和 s_j 满足 $s_i \sim s_j$.

2) $e = (v_i, v_j) \in Ec$ 中的节点 v_i 与 v_j 所对应的 s_i 和 s_j 满足 $s_i \leftrightarrow s_j$.

(3) $W = \{w_{ij}\}$为边的权值集合, 对 $\forall e = (v_i, v_j) \in Ef$, $w_{ij} = Nl(s_i, s_j)$; $\forall e = (v_i, v_j) \in Ec$, $w_{ij} = Nc(s_i, s_j)$.

Web API 的异质关联图中包含以下两类边.

(1) 功能关联边, 如果两个 s_i 与 s_j 存在相同的标签, 则在 HAG 中的节点 v_i 和 v_j 之间建立一条功能关联边, 功能关联边的权重为 s_i 与 s_j 共同标签的数量, 记作 $Nl(s_i, s_j)$.

(2) 协作关联边. 如果两个 s_i 与 s_j 共存于同一个 Mashup 服务的组成服务集合中, 则在 HAG 中的节点 v_i 和 v_j 之间建立一条协作关联边, 协作关联边的权重为 s_i 与 s_j 共同参与 Mashup 服务的数量, 记作 $Nc(s_i, s_j)$. 图 4 为 ProgrammableWeb 网站中注册的部分 Mashup 服务的组件 Web API 所构建的异质关联图片段. 其中, 节点为 Web API, 虚线边为功能关联边, 实线边为协作关联边, 边上的数值为边权.

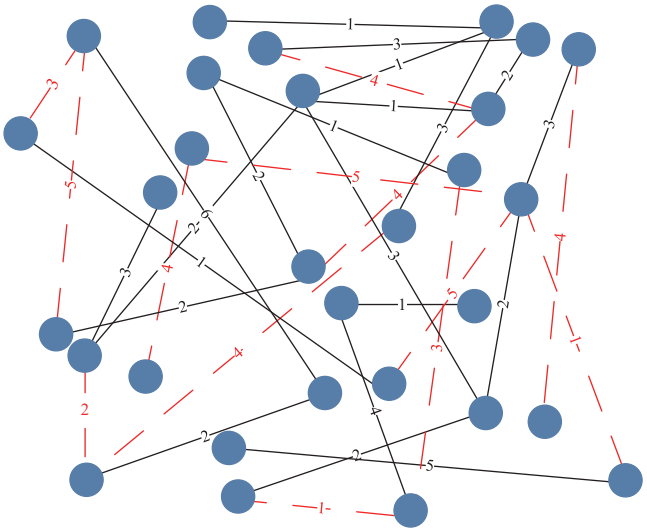

图 4 Web API 异质关联图

定义 6. Mashup 服务需求. Mashup 服务需求为一个二元组 $mr = (d, L)$. 其中, d 为 Mashup 服务需求描述信息, L 为 Mashup 服务需求的类别标签集合.

2.2 研究框架

本文采取如图 5 所示的研究框架, 为 Mashup 服务需求进行组件 Web API 推荐的主要流程如下.

首先, 在服务注册平台中爬取 Web API (组件服务) 以及 Mashup 服务. 将上述服务进行数据清洗, 构建研究使用的数据集.

然后, 从功能层面获取 Mashup 服务需求 mr 的候选组件 Web API 集合, 主要步骤分为以下两个阶段.

第 1 阶段, 获取服务需求 mr 的候选组件集合. 在 Mashup 服务集合中寻找与 mr 功能相似 Mashup 服务, 记为 $Nm(mr)$. 将 $Nm(mr)$ 中每一个 Mashup 服务的组件 Web API 取出, 构建 mr 的一阶潜在组件服务集合 $Cw\text{-}1(mr)$. 为 $Cw\text{-}1(mr)$ 中的每个 Web API, 选取一定数量的相似 Web API 作为 mr 的二阶潜在组件服务集合 $Cw\text{-}2(mr)$, 将 $Cw\text{-}1(mr)$ 与 $Cw\text{-}2(mr)$ 中的服务合并, 形成 mr 的候选组件集合 $Cw(mr)$.

第 2 阶段, 为服务需求 mr 和候选 Web API 生成潜在联合词. 利用改进的 YAKE 模型从 $Nm(mr)$ 中 Mashup 服务和 $Cw(mr)$ 中 Web API 的描述文本提取特征词集合, 通过标签适配度, 分别为 mr 和 $Cw(mr)$ 中的每个服务 s 筛选潜在联合词 $Law(mr)$ 和 $Law(s)$.

其次, 利用 $Law(mr)$ 和 $Law(s)$, 为 mr 和 $Cw(mr)$ 中的每个服务 s 生成潜在联合词向量. 将潜在联合词向量与功能向量相拼接, 分别为 Mashup 服务需求和 Web API 生成融合潜在联合词的需求向量和功能向量.

再次, 构建融合功能关联和协作关联的 Web API 异质关联图. 基于改进 GATNE 模型与设计的面向关联强度的游走策略, 为异质关联图中的服务节点生成关联向量.

最后, 结合自注意力机制优化 *mr* 的需求向量和 Web API 的关联向量, 与 Web API 功能向量一起输入全连接层, 输出每个候选 Web API 的概率值进行 Top-*K* 推荐.

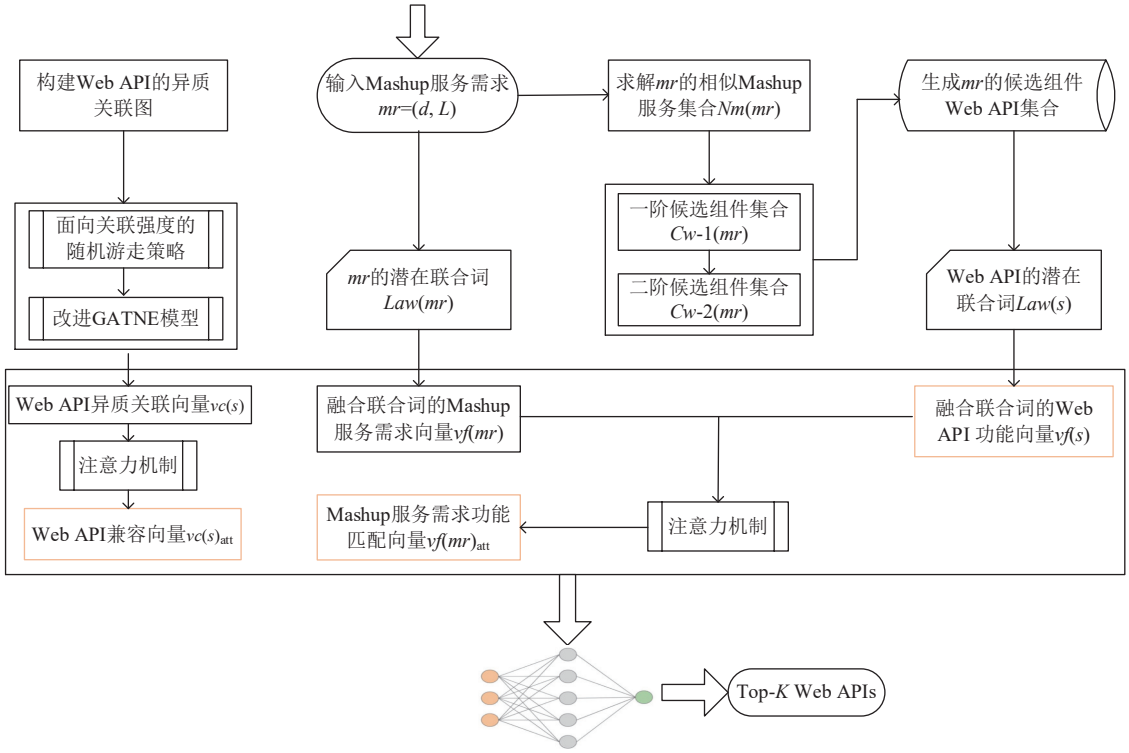

图 5 研究框架

3 研究方法

本节从服务功能向量的生成、服务关联向量的生成以及组件 Web API 的推荐这 3 个层次, 介绍所构建融合潜在联合词与异质关联兼容的 Mashup 服务组件 Web API 推荐方法.

3.1 融合潜在联合词的服务功能向量生成

高质量的服务功能向量能够提升 Web API 与 Mashup 服务需求之间的功能匹配精确度, 从而提高候选组件 Web API 集合的求解质量. 本文构建一种融入场景契合度的服务描述特征词提取模型 SYAKE, 用于提取服务描述中的特征词, 并结合标签适配度, 为 Mashup 服务需求和候选 Web API 生成潜在联合词. 建立融合潜在联合词的 Mashup 服务需求向量和 Web API 功能向量, 以提升功能匹配的合理性, 具体流程见图 6.

3.1.1 融入场景契合度的服务描述特征词提取模型 SYAKE

YAKE 是一种无监督关键词抽取模型, 不依赖于外部训练语料, 在不同长度文本的关键词提取中均具有优异性能[27]. 为生成高质量的潜在联合词, 需要将能够表示服务应用场景的特征词优先提取出来. 然而, YAKE 模型从文本统计的角度提取特征词, 难以保障提取的特征词与服务应用场景具有较高的语义关联度. 为此, 本文构建了一种融入场景契合度的服务描述特征词提取模型 SYAKE.

SYAKE 保留原始 YAKE 模型的四种权重指标: 词形权重 W_{case} (首字母大写的单词重要度高于小写的单词)、位置权重 W_{pos} (位置越靠前的词重要度越高)、词汇共现权重 W_{rel} (两侧不同词的数量越多, 单词重要度越低) 和句频权重 W_{dif} (出现在句子数量越多的单词重要度越高). 在此基础上, 设计了语境权重和场景适配权重 (合称为场景契合度), 提高筛选特征词与文本语境及应用场景的契合度.

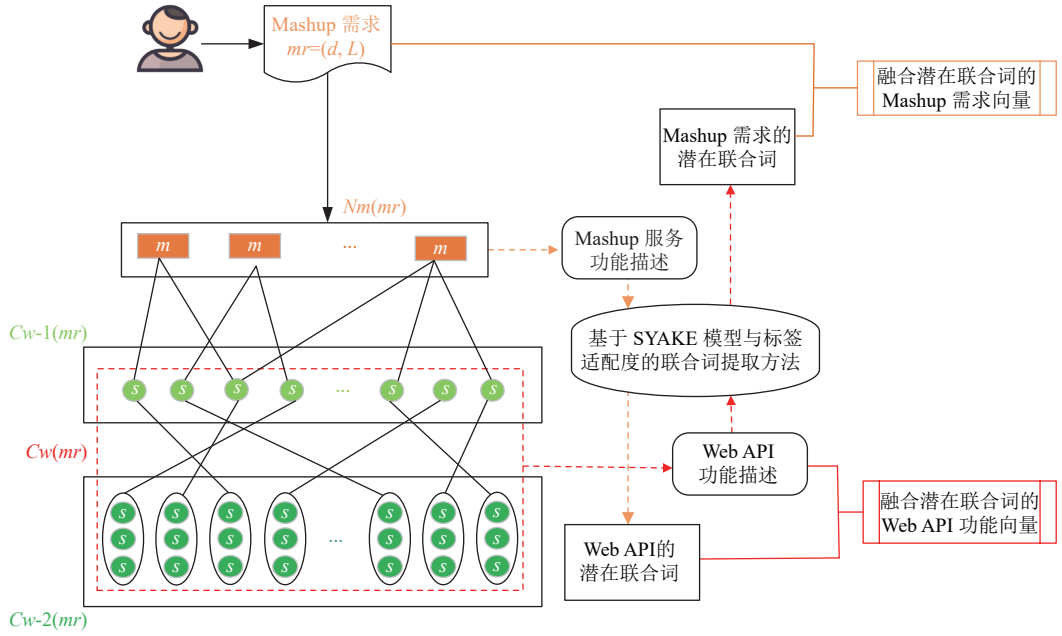

图 6 融合潜在联合词的服务功能向量生成

定义 7. 语境权重. D 为一组服务描述集合, w 为服务描述 d 所包含的一个词, $MeanSemSim(w, d)$ 是 w 与 d 中词语的语义相似度的均值, $TF\text{-}IDF(w, d, D)$ 是词 w 的 $TF\text{-}IDF$ 词频, 词 w 的语境权重定义见公式 (1), 语境权重表示了词语在服务描述中的重要度, 其值越大, 词语的重要度越高.

$$W_{\text{Con}} = MeanSemSim(w,d)\cdot(TF\text{-}IDF(w,d,D)) \tag{1}$$

定义 8. 场景适配权重. Ld 为服务描述集合 D 中的服务所对应标签的集合. 对于 $d \in D$ 中的词语 w, 其场景适配权重如公式 (2):

$$W_{\text{scen}} = \frac{\sum_{l\in d.L} sim(h_w,h_l)}{|Ld|}\cdot\left(-\log\left(\frac{\sum_{l\in(Ld-d.L)} sim(h_w,h_l)}{|Ld-d.L|}\right)\right) \tag{2}$$

在本文中, $sim(h_i, h_j) = h_i h_j/||h_i||\cdot||h_j||$为余弦相似度函数. $d.L$ 为服务描述 d 的标签集合, h_w 与 h_l 分别为 w 与 l 所对应的词向量, 由 Word2Vec 训练得到. 标签表示了服务的应用场景, 如果词 w 与其所在服务描述对应标签的相似度高, 且与其他服务的标签相似度低, 则其场景适配权重高, 说明该词与服务描述的应用场景契合度高[28].

受 Campos 等人[27]在 YAKE 中对各项词语权重组合方式的启发, 在 SYAKE 中, 基于以上 6 种特征权重获得词语 w 在服务描述 d 中的综合权重如公式 (3) 所示. 权重值越低, 表示词 w 在服务描述 d 中重要度越高.

$$\text{SYAKE}(w) = \frac{W_{\text{pos}}\cdot W_{\text{rel}}}{W_{\text{case}} + \frac{W_{\text{con}}}{W_{\text{rel}}} + \frac{W_{\text{dif}}}{W_{\text{rel}}} + W_{\text{scen}}} \tag{3}$$

3.1.2 潜在联合词的推荐

为 Mashup 服务需求和候选组件 Web API 提取潜在联合词, 可以补充 Mashup 服务需求描述和 Web API 功能描述的应用场景特征, 丰富二者匹配过程中的功能语义, 提高匹配精确度.

在进行潜在联合词提取时, 需要为 Mashup 服务需求 mr 确定候选组件 Web API 集合. 首先, 获取与 Mashup 服务需求相似的 Mashup 服务集合 $Nm(mr)$. 然后, 分别求得 $Nm(mr)$ 的一阶和二阶组件 Web API 集合 $Cw\text{-}1(mr)$ 和 $Cw\text{-}2(mr)$. 将 $Cw\text{-}1(mr)$ 和 $Cw\text{-}2(mr)$ 合并为 $Cw(mr)$, 作为 mr 的候选组件 Web API 集合. 为 Mashup 服务需求 mr 推荐的潜在联合词将在 $Cw(mr)$ 集合中的 Web API 的服务描述中提取. $Cw(mr)$ 中的 Web API 所对应的潜在联合词则提取至

Nm(*mr*) 中的 Mashup 服务的服务描述. 因此, 推荐潜在联合词的首要任务是构建 *Nm*(*mr*) 和 *Cw*(*mr*) 集合.

构建 *Nm*(*mr*) 和 *Cw*(*mr*) 集合时, 需要进行相似度的判定. 服务相似度的判定通常需要借助主题模型或神经网络模型将服务功能描述转化为服务功能向量, 然后利用服务功能向量的相似度计算服务之间相似度. 服务功能多采用短文本自然语言进行描述. 相比主题模型, 以 BERT 为代表的预训练神经网络模型更适合服务功能描述的特征提取. 然而, BERT 所生成向量空间的各向异性影响了服务功能向量的相似度计算精确度. Gao 等人提出的 SimCSE 可以有效缓解 BERT 生成向量空间的各向异性, 提升服务功能向量的生成质量[29]. 在 SimCSE 中, 采用 BERT 模型为服务描述文本生成功能向量. 设 d 为一段服务描述文本, 令 $h=f_\theta(d)$, f_θ 为调参的 BERT 模型, h 为 d 对应的文本向量.

在为 *mr* 及 *Cw*(*mr*) 中的服务生成潜在联合词时, 需确定 *Nm*(*mr*) 和 *Cw*(*mr*) 中的具体服务. 在此, 利用 SimCSE 框架分别为 Mashup 服务需求 *mr*、已有 Mashup 服务和 Web API 分别生成功能向量. 通过计算 *mr* 与已有 Mashup 服务之间的功能相似度, 为 *mr* 寻找相似度最高的 Top-X 个 Mashup 服务, 构建如公式 (4) 所示的集合 *Nm*(*mr*), 其中 *MS* 为已有 Mashup 服务集合.

$$Nm(mr)=\{m|\text{Top-}X(sim(h_{mr},h_m))\wedge m\in MS\} \tag{4}$$

获取 *Nm*(*mr*) 中每一个 Mashup 服务的组件 Web API, 构建如公式 (5) 所示 *mr* 的一阶候选组件服务集合 *Cw*-1(*mr*):

$$Cw\text{-}1(mr)=\bigcup_{m\in Nm(mr)} m.S \tag{5}$$

为提高候选组件 Web API 推荐的合理性, 在一阶候选组件集合 *Cw*-1(*mr*) 的基础上构建二阶候选组件服务集合 *Cw*-2(*mr*), 见公式 (6):

$$Cw\text{-}2(mr)=\bigcup_{s\in Cw\text{-}1(mr)}\{\text{Top-}Y(sim(h_s,h_{s'}))\wedge s'\in WS\} \tag{6}$$

在 Web API 集合 *WS* 中, 选择与 *Cw*-1(*mr*) 中每一个组件 Web API 功能相似度最高的 Top-Y 个邻居 Web API 加入二阶候选组件服务集合 *Cw*-2(*mr*). 最后, 将 *Cw*-1(*mr*) 与 *Cw*-2(*mr*) 中的 Web API 合并, 形成 Mashup 服务需求 *mr* 的候选组件集合 *Cw*(*mr*).

接下来, 使用 SYAKE 提取 *Nm*(*mr*) 与 *Cw*(*mr*) 中服务描述的功能特征词, 并通过标签适配度为 *mr* 和候选组件服务生成潜在联合词. 见公式 (7)–公式 (9):

$$F(w,l)=sim(h_w,h_l)\cdot\left(-\log\left(\frac{\sum_{l'\in s.L-l}sim(h_w,h_{l'})}{|s.L-l|}\right)\right) \tag{7}$$

$$Law(mr)=\bigcup_{l\in mr.L}\left\{\text{Top-}n(F(l,w))\wedge w\in\bigcup_{s\in Cw(mr)}\text{SYAKE}(s.d)\right\} \tag{8}$$

$$Law(s)=\bigcup_{l\in s.L}\left\{\text{Top-}n(F(l,w))\wedge w\in\bigcup_{m\in Nm(mr)}\text{SYAKE}(m.d)\right\} \tag{9}$$

服务 *s* 的功能描述 *d* 中词语 *w* 与其标签 *l* 的适配度 *F*(*w*, *l*) 参见公式 (7). 标签适配度由两部分组成: 词语 *w* 与标签 *l* 的相似度, 以及 *w* 与标签组除 *l* 外的标签相似度的倒数. *F*(*w*, *l*) 值越大, 说明词语 *w* 与标签 *l* 所表示的当前服务的功能场景越匹配.

采用公式 (8) 为 *mr* 生成潜在联合词, 首先为 *Cw*(*mr*) 中每个 Web API 采用 SYAKE 模型提取特征词, 形成一个特征词集合. 在该特征词集合中, 利用标签适配度, 为 *mr* 的每个标签筛选 Top-n 适配度最高的词语, 构成 *mr* 的潜在联合特征词 *Law*(*mr*). 类似地, 采用公式 (9) 为 *Cw*(*mr*) 中的每个服务 *s* 推荐潜在联合词 *Law*(*s*).

3.1.3 融合潜在联合词的服务功能向量

Mashup 服务需求的潜在联合词提取自相似功能 Mashup 服务的组件服务描述中, 而候选组件 Web API 的潜在联合词提取至其可能参与的 Mashup 服务, 因此, 潜在联合词从需求-供给互换的角度为 Mashup 服务需求和候选组件 Web API 分别增加功能场景特征描述词. 将潜在联合词向量融入到 Mashup 服务需求与候选组件 Web

API, 可以提高二者的匹配合理性与准确性.

h_{mr} 为采用 SimCSE 为需求 *mr* 生成的功能向量. 采用 Word2Vec 为 *mr* 的潜在联合词 *Law*(*mr*) 生成词向量. 将这些词向量的均值作为 *mr* 的潜在联合词向量与 h_{mr} 拼接, 形成的向量作为 *mr* 的需求向量, 记作 *vf*(*mr*), 参见公式 (10), 其中, 符号||表示向量拼接.

$$vf(mr)=\frac{\sum_{w\in Law(mr)}\text{Word2Vec}(w)}{|Law(mr)|}\|h_{mr} \tag{10}$$

同样, 对于 *Cw*(*mr*) 中的服务 *s*, 使用 SimCSE 为其生成功能向量 h_s. 将其与潜在联合词向量均值拼接, 形成的向量作为 *s* 的服务功能向量, 记作 *vf*(*s*), 参见公式 (11):

$$vf(s)=\frac{\sum_{w\in Law(s)}\text{Word2Vec}(w)}{|Law(s)|}\|h_{s} \tag{11}$$

将融合潜在联合词的服务功能向量命名为 SY-SimCSE. 采用 SY-SimCSE 为 Mashup 服务需求生成需求向量 *vf*(*mr*) 和候选 Web API 生成服务功能向量 *vf*(*s*) 的方法参见算法 1.

算法 1. 融合潜在联合词的 Mashup 服务需求向量与候选 Web API 功能向量生成.

输入: Mashup 服务需求 *mr*, Mashup 服务集合 *MS*, Web API 集合 *WS*;
输出: 融合潜在联合词需求向量 *vf*(*mr*), 候选组件服务的功能向量集合{*vf*(*s*)}.

1. $h_{mr}=SimCSE(mr)$
2. $h_m=SimCSE(m)$ **for each** $m\in MS$
3. $h_s=SimCSE(s)$ **for each** $s\in WS$
4. $Nm(mr)=\{m|\text{Top-}X(sim(h_{mr}, h_m)\wedge m\in MS)$
5. **for** $\forall m\in Nm(mr)$
6. $\quad Cw\text{-}1(mr)=Cw\text{-}1(mr)\cup m.S$
7. **endfor**
8. **for** $\forall s\in Cw\text{-}1(mr)$
9. $\quad Ns(s)=\{s|\text{Top-}Y(sim(h_s, h_{s'})\wedge s'\in WS)$
10. $\quad Cw\text{-}2(mr)=Cw\text{-}2(mr)\cup Ns(s)$
11. **endfor**
12. $Cw(mr)=Cw\text{-}1(mr)\cup Cw\text{-}2(mr)$
13. **for** $\forall s\in Cw(mr)$ ***or*** $\forall m\in Nm(mr)$
14. $\quad Lmr=Lmr\cup\text{SYAKE}(s.d)$
15. $\quad Ls=Ls\cup\text{SYAKE}(m.d)$
16. **endfor**
17. **for** $\forall l\in mr.L$
18. $\quad Law(mr)=\text{Top-}n(F(w, l))\wedge w\in Lmr$
19. **endfor**
20. generate *vf*(*mr*) by Formula (10)
21. **for** $\forall s\in Cw(mr)$ **and** $l\in s.L$
22. $\quad Law(s)=\text{Top-}n(F(w, l))\wedge w\in Ls$
23. **endfor**
24. generate *vf*(*s*) by Formula (11) **for each** $s\in Cw(mr)$
25. **return** (*vf*(*mr*), {*vf*(*s*)})

算法 1 第 1–3 行采用 SimCSE 为 Mashup 服务需求 *mr*、Mashup 服务与 Web API 生成功能向量. 第 4–7 行从已有 Mashup 服务中为需求 *mr* 查找 *X* 个相似 Mashup 服务集合 *Nm*(*mr*), 并将它们调用的组件 Web API 作为 *mr* 的一阶候选 Web API 集合 *Cw*-1(*mr*). 第 8–12 行通过为 *Cw*-1(*mr*) 中每一个服务查找 *Y* 个相似 Web API, 作为 *mr* 的二阶候选组件 Web API 集合, 并将 *Cw*-1(*mr*) 与 *Cw*-2(*mr*) 合并形成 *mr* 的候选组件集合 *Cw*(*mr*). 第 13–16 行使用 SYAKE 模型分别从 *Cw*(*mr*) 中的 Web API 与 *Nm*(*mr*) 中的 Mashup 服务的功能描述中分别提取特征词集合 *Lmr* 与 *Ls*.

第 17–20 行通过标签适配度为需求 *mr* 从 *Lmr* 中匹配自身标签数量 *n* 倍的特征词, 作为潜在联合词集合 *Law*(*mr*), 然后利用公式 (10) 生成 *mr* 对应的融合潜在联合词的需求表达向量. 同样处理方式, 算法的第 21–24 行为 *Cw*(*mr*) 中的每一个服务 *s* 通过标签适配度从 *Ls* 中匹配自身标签数量 *n* 倍的特征词, 作为服务的潜在联合词集合 *Law*(*s*). 利用公式 (11) 为 *Cw*(*mr*) 的 Web API 生成功能向量集合{*vf*(*s*)}. 算法第 25 行返回 *vf*(*mr*) 和{*vf*(*s*)}.

3.2 融合异质关联的 Web API 关联向量生成

获取候选组件 Web API 集合后, 多数方法通过计算候选组件服务之间的协作兼容度来选取最终的推荐服务, 此类方法未考虑 Web API 之间的功能重复问题. 本文提出一种融合功能关联和协作关联的 Web API 关联兼容度计算方法, 以解决已有方法在推荐多样性方面存在的不足.

3.2.1 基于改进 GATNE 的节点关联特征初始化

为了计算关联兼容度, 需将定义 5 中 Web API 异质关联图的节点向量化. GATNE 是当前流行的异质图节点嵌入模型, 适合处理具有大数量级边的异质图[30]. 图 7 为 Web API 关联向量生成过程.

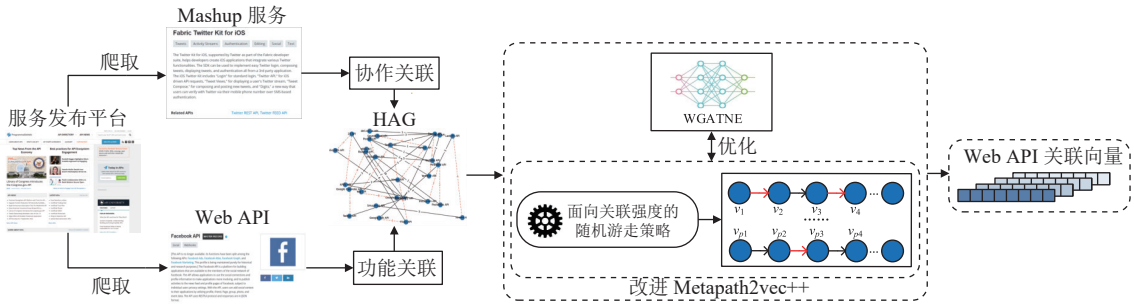

图 7 Web API 关联向量建模

在 GATNE 模型中, 节点 v_i 在 *r* 类型边上获取的表征分为节点基础表征与边嵌入表征. 第 *k* 层 v_i 节点的 *r* 类型边嵌入表征 $u_{i,r}^{(k)}$ 为:

$$u_{i,r}^{(k)} = aggregator\left(\left\{u_{j,r}^{(k-1)}, \forall v_j \in N_{i,r}\right\}\right) \tag{12}$$

其中, $N_{i,r}$ 是 *r* 类型边上的 v_i 关联节点集合, $u_{j,r}^{(0)}$ 为随机初始化的边向量, 采用公式 (13) 对 $u_{j,r}^{(k)}$ 实施最大池化聚合操作, $\phi(x)$ 为 ReLU 激活函数.

$$u_{i,r}^{(k)} = \max\left(\left\{\varphi\left(\hat{Q}_{\text{pool}}^{(k)} u_{j,r}^{(k-1)} + \hat{b}_{\text{pool}}^{(k)}\right), \forall v_j \in N_{i,r}\right\}\right) \tag{13}$$

将节点 v_i 所有的边嵌入表征进行拼接. 假设 v_i 关联的 *r* 类型的边为 *m* 条, 边嵌入维度为 *d*, 则拼接后的边嵌入表征为 $U_{i,r} = (u_{i,1}, u_{i,2}, \ldots, u_{i,m})$, 构建如公式 (14) 的自注意力机制系数 $a_{i,r} \in R^m$, 其中 $w_r \in R^{da}$, $W_r \in R^{[da \times d]}$是需要训练的参数, T 表示向量或者矩阵的转置.

$$a_{i,r} = Softmax\left(w_r^{\mathrm{T}} \tanh\left(W_r U_i\right)\right)^{\mathrm{T}} \tag{14}$$

然而, GATNE 模型未考虑异质图的边权. 在 HAG 中边权反映了 Web API 间关联紧密度, 忽视边权导致模型难以区分节点在不同关联中的重要度, 进而影响 Web API 关联表征的生成质量.

为此, 本文将边权融入到 GATNE 的特征聚合过程, 对于 $w_{i,j}$ 构造如公式 (15) 所示的偏置向量, 融入到边嵌入

表征生成的注意力计算.

$$b_{i,j}^{k}=\sigma\left(W_{w}^{k}u_{i,j}^{k}\right) \tag{15}$$

其中, σ 为 *Sigmoid* 激活函数, $u_{i,j}^{k}$ 为 v_i 与邻居节点 v_j 之间 r 类型边第 k 层嵌入表征, 边嵌入维度为 d, $w_{w}^{k}\in R^{[m\times d]}$为可训练权重矩阵, $b_{i,j}^{k}\in R^{m}$, m 为 r 类型边的数量.

然后, 根据 Web API 关联权重 w_{ij} 与偏置向量 b_{ij}^{k}, 将原有公式 (14) 所示的 GATNE 注意力系数改进为公式 (16):

$$a_{i,r}=Softmax\left(w_{r}^{\mathrm{T}}\tanh\left(W_{r}U_{i}\right)+\sum_{j=1}^{m}w_{i,j}b_{i,j}^{k}\right)^{\mathrm{T}} \tag{16}$$

节点 v_i 对于 r 类型边的总体表征如下:

$$v_{i,r}=b_{i}+\alpha_{r}M_{r}^{\mathrm{T}}U_{i}a_{i,r} \tag{17}$$

其中, b_i 为节点 v_i 的节点基础表征, α_r 是超参数, 为不同类型边分配的注意力系数, M_r 为可训练权重矩阵. 改进后的 GATNE 能够在特征聚合过程中考虑到关联边的边权对关联向量生成的影响, 将上述模型命名为 WGATNE.

3.2.2 关联向量的优化生成

Metapath2vec++适用于 GATNE 的节点嵌入优化[30,31]. 在生成训练语料时, Metapath2vec++依照异质边的类型, 等概率的随机采样关联节点. 因此, 在处理 Web API 异质关联图采样时, 无法将关联性最强的节点优先纳入路径序列, 进而影响关联兼容性的判定. 为此, 本文设计一种面向关联强度的随机游走策略. s_i 与 s_j 之间的关联强度参见公式 (18):

$$Cor\_d(s_i,s_j)=\frac{(Nc(s_i)+Nc(s_j))\cdot Nc(s_i,s_j)}{\sum_{l\in\left(Im(s_i).S\cup Im(s_j).S\right)}Nc(s)}-\frac{\sum_{l\in s_{i}.L\cap s_{j}.L}NS(l)\cdot Nl(s_i,s_j)}{\sum_{l\in s_{i}.L}NS(l)+\sum_{l\in s_{j}.L}NS(l)} \tag{18}$$

公式 (18) 等号右侧第 1 项为协作关联边产生的关联强度. $Nc(s)$ 为包含 Web API s 的 Mashup 服务数量. 分子为 Web API s_i 与 s_j 所隶属的 Mashup 服务总和. $Im(s_i)$ 与 $Im(s_j)$ 分别表示包含 Web API s_i 与 s_j 的 Mashup 服务. 令 s 为所有包含 s_i 与 s_j 的 Mashup 服务的组件服务集合中的 Web API, 将包含 s 的 Mashup 服务数量作为分母, 分子与分母的比值为协作关联边的重要度. 协作关联边的重要度与边权的乘积即为协作关联强度.

公式 (18) 等号右侧第 2 项为功能关联边产生的关联强度. $\sum_{l\in si.L\cap sj.L}NS(l)$ 为 s_i 与 s_j 共用标签集中的标签被所有 Web API 所调用的总量, $\sum_{l\in si.L}NS(l)$ 和 $\sum_{l\in sj.L}NS(l)$ 分别为 s_i 与 s_j 各自标签集中的标签被所有 Web API 所调用的总量, 二者比值为 s_i 与 s_j 的功能关联重要度. $Nl(s_i, s_j)$ 是共用标签数量, 由定义 5 可知, 共用标签数量为功能关联边的边权. 功能关联边的重要度与边权的乘积即为功能关联强度.

将协作关联强度与功能关联强度的差值定义为 Web API 关联强度, 使用 *Softmax* 函数对 Web API 关联强度的值进行归一化:

$$NCor\_d(s_i,s_j)=\frac{\mathrm{e}^{Cor\_d\left(s_i,s_j\right)}}{\sum_{s_j\in N_{t+1}\left(v_{i,r}\right)}\mathrm{e}^{Cor\_d\left(s_i,s_j\right)}} \tag{19}$$

假定在元路径模式 $T=v_1\rightarrow v_2\rightarrow\ldots\rightarrow v_t\rightarrow v_{t+1}\rightarrow v_l$ 中, 每步游走的边类型均与上一步边类型不同. 若第 $t-1$ 步游走边类型为 r, 则第 t 步从 v_i 到 v_j 的转移概率优化为:

$$p\left(v_j\middle|v_i,T\right)=\begin{cases}\dfrac{NCor\_d(s_i,s_j)}{|N_{t+1}(v_{i,r})|}, & (v_i,v_j)\in E,\ v_j\in N_{t+1}(v_{i,r})\\ 0, & (v_i,v_j)\in E,\ v_j\in N_{t+1}(v_{i,r})\\ 0, & (v_i,v_j)\notin E\end{cases} \tag{20}$$

其中, $N_{t+1}(v_{i,r})$ 表示 v_i 不同于边类型 r 的关联节点集. 若 v_i 节点上一跳对应关联边类型为 r, 在选择下一跳节点 v_j 时, 需要确保 v_j 隶属于不同于边类型 r 的关联节点集合, 然后从未曾访问过的节点中选择概率 $p(v_j \mid v_i, T)$ 最高的

节点 v_j 进行游走.

依据此随机游走模式, 假设 v_k 为节点 v_i 的上一跳节点, v_i 的下一跳节点集合 $v_J=\{v_{j1}, v_{j2},\ldots, v_{jm}\}$, 在选取下一跳节点时采取如下规则.

1) 若 $v_i \leftrightarrow v_k$, $\exists v_{Jf} \subseteq v_J$ 且 $\forall v \in v_{Jf}$ 满足 $v \sim v_i$, 则 v_i 下一跳节点 $v_j = \{v \mid v \in v_{Jf} \wedge \max(p(v \mid v_i, T))$.

2) 若 $v_i \sim v_k$, $\exists v_{Jc} \subseteq v_J$ 且 $\forall v \in v_{Jc}$ 满足 $v \leftrightarrow v_i$, 则 v_i 下一跳节点 $v_j = \{v \mid v \in v_{Jc} \wedge \max(p(v \mid v_i, T))$.

该游走策略综合协作关联与功能关联计算关联强度, 不但能够高效的捕获 Web API 节点在不同关联类型中产生的语义关联强度, 而且可以最大化节点序列中 Web API 兼容性, 提升了训练语料采样质量, 可以有效提高关联向量的生成质量.

对于长度为 l 的随机游走路径 $P = (v_{p1},\ldots, v_{pl})$, v_{pt} 的上下文为 $C = \{v_{pk} \mid v_{pk} \in P, |k-t| \leqslant c, t \neq k\}$, 其中 c 为窗口大小的半径. 基于上述采用策略获得给定节点 v_i 与其路径上下文 C, 最小化如下的负对数似然:

$$-\log P_\theta\left(\left\{v_j \middle| v_j \in C\right\} \middle| v_i\right) = \sum_{v_j \in C} -\log P_\theta\left(v_j \middle| v_i\right) \tag{21}$$

接下来, 使用异质 *Softmax* 函数, 对节点 v_j 的上下文出现概率进行归一化, 即给定节点 v_i 条件下, v_j 的概率:

$$P_\theta(v_j|v_i) = \frac{\exp(C^{\mathrm{T}} \cdot v_{i,r})}{\sum_{k \in V_t} \exp\left(c_k^{\mathrm{T}} \cdot v_{i,r}\right)} \tag{22}$$

其中, $v_j \in V_t$, c_k 为节点 v_k 的上下文嵌入, $v_{i,r}$ 为节点 v_i 对边类型 r 的整体嵌入. 最后, 通过负采样函数为每一对节点构建目标函数:

$$E = -\log\sigma\left(c_j^{\mathrm{T}} \cdot v_{i,r}\right) - \sum_{l=1}^{L} E_{v_k \sim P_t(v)}\left[\log\sigma(-c_k^{\mathrm{T}} \cdot v_{i,r})\right] \tag{23}$$

其中, $\sigma(x)$ 为 *Sigmoid* 激活函数, L 是针对每个正样本进行负采样得到的负样本数, v_k 从节点 v_j 对应节点集 V_t 上的噪声分布 $Pt(v)$ 中随机抽取. 当目标函数收敛时, 为 Web API 异质关联图中的节点生成最终的关联向量, 将 v 所对应的服务 s 的关联向量记作 $vc(s)$.

在关联向量生成过程中, 将协作关联和功能关联分别作为 Web API 组合兼容性判定过程中的正向因素和负向因素融入到服务关联向量中. 因此, 相比单一通过协作关联辅助 Web API 的推荐, 采用 Web API 关联向量进行兼容性判定将能进一步提高组件 Web API 推荐的多样性. 本文将在异质关联图中利用 WGATNE 模型和改进的 Metapath2vec++模型生成 Web API 关联向量的方法命名为 WGAT-HAG. 算法 2 为 Web API 关联向量的生成算法.

算法 2. 融合异质关联的 Web API 关联向量生成.

输入: Mashup 服务集合 *MS*, Web API 集合 *S*;
输出: Web API 的关联向量集合{*vc*(*s*)}.

1. HAG $\leftarrow v: s \in S$
2. **for each** $s_i, s_j \in S$
3. **if** $s_i \sim s_j$ **then**
4. HAG.*Ef* = HAG.*Ef* $\cup (v_i, v_j)$
5. $w_{ij} = w_{ij} + 1$
6. **endif**
7. **endfor**
8. **for** $\forall m \in MS$
9. **if** $s_i \leftrightarrow s_j \in m.S$ **then**
10. HAG.*Ec* = HAG.*Ec* $\cup (v_i, v_j)$
11. $w_{ij} = w_{ij} + 1$

12. **endif**
13. **endfor**
14. {*vc*(*s*)} = WGAT-HAG(*S*, HAG)
15. **return** {*vc*(*s*)}

算法 2 第 1–13 行用于构建 Web API 异质关联图 HAG. 第 1 行首先初始化图 HAG, 将 Web API 集合 S 中服务 s 映射为 HAG 中的节点 v. 第 3–7 行实现功能关联边的建立. 如果集合 S 中的两个服务 s_i 与 s_j 存在功能关联, 则在对应节点 v_i 与 v_j 之间增加一条边, 边权增加 1. 第 8–13 行实现协作关联边的建立. 通过遍历集合 *MS* 中的 Mashup 服务, 对存在协作关联的服务 s_i 与 s_j, 在对应节点 v_i 与 v_j 之间增加一条边, 边权增加 1.

在构建的 Web API 异质关联图 HAG 的基础上, 算法第 14 行利用 WGAT-HAG 为集合 S 中的服务生成关联向量集合{*vc*(*s*)}, 算法第 15 行返回关联向量集合{*vc*(*s*)}.

3.3 融合功能匹配度与关联兼容度的组件 Web API 推荐

组件 Web API 对 Mashup 服务需求的功能贡献度体现了其是否能符合开发者对于功能需求的匹配预期, 而 Web API 在候选集合内的兼容度反映了与其他 Web API 的组合协调性, 以及推荐集合的合理性, 二者缺一不可, 共同决定组件 Web API 推荐集合是否符合开发需求.

对 Mashup 服务需求 *mr* 功能贡献度高的 Web API 参与构建 *mr* 的概率更大. 为了计算候选 Web API 在 Mashup 服务需求中的功能贡献度, 引入注意力机制, 计算每个 Web API 对于 Mashup 服务需求 *mr* 的贡献注意力系数.

$$xs_i = Sigmoid(W_{xs}[vf(mr)||vf(s_i)] + b_{xs}) \tag{24}$$

$$afs_i = \frac{\exp(xs_i)}{\sum_{s_j \in Cw(mr)} \exp(xs_j)} \tag{25}$$

其中, "||"表示向量拼接, $vf(s_i)$ 为服务 s_i 的功能向量, $vf(mr)$ 为需求 *mr* 的功能向量, W_{xs} 为可训练的矩阵, b_{xs} 为偏置参数, 优化重组后的 *mr* 功能匹配向量为:

$$vf(mr)_{\text{att}} = \sum_{s_i \in Cw(mr)} afs_i \cdot vf(s_i) \tag{26}$$

为了获取 s_i 与候选 Web API 集合中不同服务的兼容度, 引入注意力机制, 计算候选 Web API 集合中服务之间的兼容性注意力系数. 服务 s_j 对服务 s_i 的注意力系数为:

$$acs_{ji} = \frac{\exp(vc(s_i)^{\mathrm{T}} \cdot vc(s_j))}{\sum_{s_k \in Cw(mr)} \exp(vc(s_i)^{T} \cdot vc(s_k))} \tag{27}$$

s_i 优化重组后的关联向量为:

$$vc(s_i)_{\text{att}} = \sum_{s_j \in Cw(mr)} acs_{ji} \cdot vc(s_j) \tag{28}$$

根据候选 Web API 的兼容性优化 s_i 的关联向量, 可以降低功能重复的 Web API 组合的可能性, 提升推荐合理性. 最后, 将优化重组的 *mr* 功能向量、Web API 的关联向量以及功能向量输入全连接层, 从 s_i 对于 *mr* 功能需求匹配预期和参与组合的 Web API 兼容度两个角度得到 s_i 对于 *mr* 的推荐概率 $p(s_i, mr)$, 经排序后取 Top-*K* 作为推荐结果, 见公式 (29):

$$p(s_i, mr) = \sigma(W_{p1}^{\mathrm{T}}[vf(mr)_{\text{att}}||vf(s_i)] + W_{p2}^{\mathrm{T}} vc(s_i)_{\text{att}} + b_p) \tag{29}$$

其中, σ 为 *Sigmoid* 激活函数, W_{p1} 与 W_{p2} 为可训练权重矩阵, b_p 为偏置项.

使用交叉熵损失函数 (cross-entropy) 优化模型, 以最小化模型预测结果和真实结果之间的差异, 如公式 (30) 所示:

$$L=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{|m_i.S|}y_{s_j,m_i}\log\left(p(s_j,m_i)\right) \tag{30}$$

其中, N 为训练集中 Mashup 服务样本的数量, $|m_i.S|$为 Mashup 服务 m_i 的 Web API 数量. $y_{sj,mi}$ 表示服务 s_j 出现在 m_i 的 Web API 集合的概率 (若存在则为 1, 否则为 0), $p(s_j, m_i)$ 表示模型预测概率. 该模型预测的概率值和真实值越接近时, 交叉熵损失函数的值越小. 将所构建的面向 Mashup 服务的组件 Web API 推荐方法命名为 R-CWSE, 具体实施步骤见算法 3.

算法 3. Mashup 服务需求的组件 Web API 推荐方法.

输入: Mashup 服务需求 *mr*, Web API 集合 *WS*;
输出: *mr* 的 Top-*K* 推荐 Web API 集合 *RWA*(*mr*).

1. get $vf(mr)$ and $\{vf(s)\}$ by Algorithm 1
2. get $\{vc(s)\}$ by Algorithm 2
3. generate $vf(mr)_{att}$ and $\{vc(s)_{att}\}$
4. **for each** $s \in Cw(mr)$
5. compute $p(s, mr)$ **by** Formula (29)
6. **endfor**
7. $RWA(mr) = \{s|\text{Top-}K(p(s, mr)) \wedge s \in Cw(mr)\}$
8. **return** $RWA(mr)$

算法 3 第 1 行调用算法 1 获取 *mr* 对应的需求向量 $vf(mr)$ 以及 $Cw(mr)$ 的 Web API 功能向量集合$\{vf(s)\}$. 第 2 行通过算法 2 获取 $Cw(mr)$ 中 Web API 的关联向量集合$\{vc(s)\}$. 第 3 行通过注意力机制优化重组需求向量和关联向量, 分别得到 $vf(mr)_{att}$ 与$\{vc(s)_{att}\}$. 第 4–6 行通过全连接层, 为每个 *s*, 从其对于 *mr* 功能需求匹配和参与组合 Web API 的关联兼容度两个角度得到 *s* 对于 *mr* 的推荐概率 $p(s, mr)$. 第 7 行将 Top-*K* 最高的概率的服务作为最终推荐的 Web API 集合 $RWA(mr)$. 算法第 8 行返回集合 $RWA(mr)$.

算法在训练阶段, 时间主要耗费在关联向量生成、功能匹配向量权重矩阵训练和推荐概率权重矩阵训练这 3 部分. 根据文献 [30], WGATNE 优化训练的复杂度为 $O(2|WS|\times l\times d_{GAT})$. 其中, 2 是 HAG 中边类型数量, l 是每个训练样本的负样本数, 且 $l\geqslant 1$, d_{GAT} 是 GATNE 总嵌入 $vc(s)$ 的维度. 功能匹配向量权重矩阵训练的复杂度为 $O((2d_{vf})\times|Cw(mr)|)$, 推荐概率权重矩阵训练的复杂度为 $O((2d_{vf}+d_{GAT})\times|Cw(mr)|)$, 其中, d_{vf} 为服务功能向量的维度. 因此, 训练阶段的时间复杂度为 $O(2|WS|\times l\times d_{GAT}+(2d_{vf})\times|Cw(mr)|+(2d_{vf}+d_{GAT})\times|Cw(mr)|)$.

在真实推荐阶段, 不涉及到模型训练, 算法主要耗时在求解 $Nm(mr)$、求解 $Cw(mr)$、为 *mr* 和 $Nm(mr)$ 中的 Web API 匹配潜在联合词以及推荐评分. 其中, 除推荐评分涉及注意力计算, 其他几项的复杂度均为所遍历集合元素数量的整数倍, 推荐评分的复杂为 $O(|Cw(mr)|^2)$. $Cw(mr)$ 中数量 Web API 的数量不超过 100, 所以真实推荐的时间复杂度为常数量级 O(1), 训练后的模型可以实现高时效性的推荐.

4 实验评估与分析

本节开展实验验证 Mashup 服务的 Web API 推荐过程中面临的以下问题.

(1) SY-SimCSE 生成的服务功能向量的质量是否优于其他模型?

(2) 异质关联兼容的引入能否提升 Web API 的推荐精确度?

(3) R-CWSE 方法是否优于最新的 Web API 推荐方法?

(4) R-CWSE 方法对于冷启动 Web API 的推荐性能如何?

(5) 关键实验参数设置.

1) Mashup 服务需求的相似服务与二阶组件 Web API 的数量设置多少时推荐质量最佳?

2) 单个标签对应潜在联合词融入数量为多少时推荐质量最佳?

4.1 数据集与实验环境

实验中使用的 Mashup 服务及 Web API 爬取自 ProgrammableWeb 网站, 删除描述过短、重复注册的 Mashup 服务后, 数据集中包含 5 305 个 Mashup 服务, 19 240 个 Web API. 其中, 每个 Mashup 服务调用组件 Web API 的平均数量为 2.8 个, 每条服务描述包含的单词均值为 27. 对所有的服务描述文本均采取分词、去停用词、词干还原等处理后用于推荐实验, 采用十折交叉验证法, 取全部实验数据平均值作为最终实验结果.

实验环境如下: CPU 为 RYZEN9-5900HS, 内存 40 GB, GPU 为 RTX3060 (一张), 显存 6 GB. 操作系统为 Windows 11, 编程语言为 Python 3.8. 所采用的模型或方法参数设置如表 1 所示.

表 1 实验参数

参数名称	参数值
批处理大小	32
训练轮次	100
W-GATNE维度	384×2
W-GATNE窗口大小	5
W-GATNE邻居采样个数	30
SimCSE维度	768
Word2Vec维度	384
单标签对应潜在联合词融入个数	6

4.2 评价指标

采用推荐精度评价指标 *Recall* 和 *Precision*[32]、排序评价指标 *NDCG*[25]、多样性评价指标 *ILS*[26,33]评估 Web API 的推荐质量. *TP*、*FP* 和 *FN* 分别表示预测为正类的正样本、预测为正类的负样本和预测为负类的正样本.

(1) 召回率 (*Recall*): 推荐列表中正确的 Web API 与 Mashup 服务需求对应的真实 Web API 数量的比值. 其值越大, 表示推荐 Web API 中被预测正确的比例越大, 推荐质量越高.

$$Recall = \frac{TP}{TP+FN} \tag{31}$$

(2) 精确率 (*Precision*): 推荐列表中正确的 Web API 与推荐 Web API 数量的比值. 其值越大, 表示预测正确的 Web API 数量越多, 推荐质量越高.

$$Precision = \frac{TP}{TP+FP} \tag{32}$$

(3) 归一化折损累积增益 (normalized discounted cumulative gain, *NDCG*): *NDCG* 用于衡量推荐列表的整体质量. 推荐正确的 Web API 数量越多, 在推荐列表中整体越靠前, *NDCG* 的值越大, 如公式 (33)–公式 (35) 所示:

$$DCG = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \tag{33}$$

$$IDCG = \sum_{i=1}^{|REL|} \frac{rel_i}{\log_2(i+1)} \tag{34}$$

$$NDCG = \frac{DCG}{IDCG} \tag{35}$$

其中, *DCG* 为非归一化损失累计增益, 若第 i 个推荐 Web API 为目标 Mashup 需求的真实组件服务, rel_i 为 1, 否则为 0. 公式 (33) 根据每个推荐 Web API 的位置进行加权, 排名越靠前的 Web API, 位置权重越大, 对推荐列表的整体质量评分有更大的贡献. *IDCG* 为真实折损累计增益, 是 *DCG* 的理想情况, 用于对 *DCG* 进行归一化, 计算 *NDCG*. 其中, *REL* 为理想情况下的推荐列表.

(4) 多样性 (indra-list similarity, *ILS*): *ILS* 用于评价 Web API 推荐列表的多样性, 如公式 (36) 所示:

$$ILS(R)=\frac{2}{N(N-1)}\sum_{s_i\in R}\sum_{s_j\in R, i\neq j} sim(s_i,s_j) \tag{36}$$

其中, R 表示 Web API 推荐列表, N 为 R 中的服务数量, $sim(s_i, s_j)$ 为度量 s_i 与 s_j 相似度的余弦相似度函数. *ILS* 值越小, 推荐列表的多样性越好.

4.3 实验结果与分析

数据集中 Mashup 服务 Web API 平均调用数量为 2.8, 因此选取 Top-3、Top-5 和 Top-7 这 3 种长度的推荐结果进行验证. 在结果展示中, 表格中的加粗数据为对比指标的最优值.

4.3.1 SY-SimCSE 的服务功能向量生成质量评估

选取 LDA[34]、RoBERTa[35]、GSDMM[36]和 SimCSE[29]作为对比模型, 其中 LDA 与 GSDMM 为主题模型, RoBERTa 和 SimCSE 是以 BERT 为基础改进的神经网络模型. 在 Y-SimCSE 方法中, 采用原始 YAKE 模型提取功能特征词. 通过不同模型为 Mashup 服务需求和 Web API 的服务描述生成功能向量, 进行相似度匹配实现 Web API 推荐, 各评价指标参见表 2.

表 2 服务功能表征向量生成质量评估

Top-*K*	指标	LDA	GSDMM	RoBERTa	SimCSE	Y-SimCSE	SY-SimCSE
K=3	*Recall*	0.192	0.212	0.347	0.378	0.405	**0.419**
	Precision	0.168	0.195	0.313	0.349	0.366	**0.381**
	NDCG	0.198	0.223	0.344	0.384	0.409	**0.420**
	ILS	0.821	0.779	0.634	0.622	0.601	**0.589**
K=5	*Recall*	0.218	0.321	0.469	0.478	0.502	**0.517**
	Precision	0.115	0.161	0.241	0.249	0.268	**0.286**
	NDCG	0.224	0.245	0.382	0.404	0.433	**0.461**
	ILS	0.792	0.798	0.661	0.626	0.599	**0.562**
K=7	*Recall*	0.328	0.399	0.561	0.579	0.603	**0.633**
	Precision	0.112	0.131	0.237	0.242	0.262	**0.271**
	NDCG	0.256	0.287	0.425	0.443	0.467	**0.499**
	ILS	0.731	0.759	0.602	0.589	0.557	**0.518**

LDA 生成的服务功能向量获得的推荐质量最差. GSDMM 生成的功能向量的 Web API 推荐质量要显著优于 LDA. 相比 LDA, GSDMM 模型的短文本主题特征提取效果更佳, 适用于单词数量较少的服务描述文本的功能特征提取[4,32]. RoBERTa 模型由大规模文本数据进行预训练, 作为神经网络模型, 捕获文本上下文语义特征的能力显著优于主题模型, 能够更精准地获得服务描述的功能语义特征, 其效果优于 LDA 和 GSDMM. 在此基础上. SimCSE 缓解了 RoBERTa 向量空间的各向异性, 在功能匹配环节的推荐质量得以进一步提升, 因此, 各指标得分均优于前 3 种模型.

在 Y-SimCSE 和 SY-SimCSE 中, 通过为 Mashup 服务需求和候选组件 Web API 匹配潜在联合词, 补充应用场景特征. 相对 SimCSE, 在 *Recall*、*Precision* 和 *NDCG* 指标分别平均提升 5.23%、6.67% 和 6.26%, 验证了应用场景特征的融入能有效提高功能匹配精度. 应用场景特征的融入弥补了原有仅依靠服务描述文本进行功能特征提取时存在的功能语义完备性以及特征区分度的不足. 在所有模型中, SY-SimCSE 的 *Recall*、*Precision* 和 *NDCG* 得分最高, 性能优于其他模型. 相对于 Y-SimCSE 分别平均提升约 3.91%、4.69% 和 5.47%, 具有更高的推荐质量, 说明本文对 YAKE 模型的改进效果明显.

不同 K 值下, SY-SimCSE 的 ILS 指标值均低于其他模型, 说明 SY-SimCSE 推荐 Web API 之间的相似度更低, 推荐结果多样性更好. 相比 SimCSE, 推荐结果中 SY-SimCSE 的 *ILS* 平均降低 9.36%, 其原因是潜在联合词增加了服务的应用场景特征, 使得功能匹配度的计算更近契合 Mashup 服务构建的真实场景. 综上所述, SY-SimCSE 综合考虑了功能和应用场景, 可以推荐高质量的 Web API 用于构建 Mashup 服务, 推荐效果优于各个对比模型.

4.3.2 异质关联兼容对推荐质量的提升效能评估

为了验证异质关联兼容对 Web API 推荐的影响, 本节验证仅采用服务功能向量、引入协作兼容和异质关联兼容这 3 种情形下的推荐质量, 主要包含以下方法.

(1) SY-SimCSE: 仅采用 SY-SimCSE 生成的服务功能向量进行 Web API 推荐.

(2) SY-SimCSE-GAT-C: 采用 SY-SimCSE 生成服务功能向量, 仅融合以 GATNE 在 Web API 的协作关联图中提取的关联向量进行推荐.

(3) SY-SimCSE-GAT-HAG: 采用 SY-SimCSE 生成服务功能向量, 融合以 GATNE 在 Web API 的异质关联图(协作和功能关联) 中提取的关联向量进行推荐.

(4) SY-SimCSE-WGAT-C: 采用 SY-SimCSE 生成服务功能向量, 仅融合以 WGATNE 在 Web API 的协作关联图中提取的关联向量进行推荐.

(5) R-CWSE (SY-SimCSE-WGAT-HAG): 本文提出的服务推荐方法, 即采用 SY-SimCSE 生成服务功能向量, 融合以 WGATNE 在 Web API 的异质关联图 (协作和功能关联) 中提取的关联向量进行推荐.

根据表 3 中不同 *K* 值实验中的前 3 行数据可知, 引入协作关联后, 相比 SY-SimCSE, SY-SimCSE-GAT-C 和 SY-SimCSE-WGAT-C 在 *Recall*、*Precision*、*NDCG* 和 *ILS* 上均有所改善. 其中, SY-SimCSE-GAT-C 在 *Recall*、*Precision* 和 *NDCG* 平均提升了 2.91%、2.38% 和 4.46%, 在 *ILS* 平均降低 5.11%; SY-SimCSE-WGAT-C 在 *Recall*、*Precision* 和 *NDCG* 平均提升了 8.73%、7.25% 和 10.07%, 在 *ILS* 平均降低 17.44%. 上述数据表明, 在引入协作兼容后, 推荐的质量得到提升. 同时, 采用 WGATNE 模型融入协作兼容得到的推荐质量要优于 GATNE 模型, 说明本文对 GATNE 的改进是有效的.

表 3 异质关联兼容对推荐质量的提升效能评估

Top-*K*	模型	*Recall*	*Precision*	*NDCG*	*ILS*
K=3	SY-SimCSE	0.419	0.381	0.420	0.595
	SY-SimCSE-GAT-C	0.431	0.388	0.434	0.557
	SY-SimCSE-WGAT-C	0.459	0.402	0.451	0.502
	SY-SimCSE-WGAT-HAG	0.445	0.395	0.442	0.511
	R-CWSE (SY-SimCSE-WGAT-HAG)	**0.478**	**0.428**	**0.470**	**0.462**
K=5	SY-SimCSE	0.517	0.286	0.461	0.562
	SY-SimCSE-GAT-C	0.535	0.298	0.495	0.533
	SY-SimCSE-WGAT-C	0.571	0.319	0.515	0.461
	SY-SimCSE-WGAT-HAG	0.559	0.311	0.51	0.494
	R-CWSE (SY-SimCSE-WGAT-HAG)	**0.598**	**0.338**	**0.529**	**0.422**
K=7	SY-SimCSE	0.633	0.271	0.499	0.518
	SY-SimCSE-GAT-C	0.649	0.274	0.513	0.499
	SY-SimCSE-WGAT-C	0.676	0.285	0.553	0.420
	SY-SimCSE-WGAT-HAG	0.665	0.281	0.534	0.448
	R-CWSE (SY-SimCSE-WGAT-HAG)	**0.685**	**0.292**	**0.577**	**0.398**

表 3 中不同 *K* 值实验中后两行是引入异质关联兼容后的推荐结果, 相比 SY-SimCSE, SY-SimCSE-GAT-HAG 在 *Recall*、*Precision* 与 *NDCG* 平均提升 6.36%、5.28% 和 7.65%, 在 *ILS* 平均降低 13.29%; R-CWSE 在 *Recall*、*Precision* 与 *NDCG* 平均提升 12.24%、12.79% 和 14.17%, 在 *ILS* 平均降低 23.46%. 上述数据表明, 引入异质关联后, Web API 的推荐质量得到显著提升, 并且提升幅度明显高于仅引入协作关联兼容. 同时, 数据表明, 采用 WGATNE 模型融入异质关联兼容得到的推荐质量要优于 GATNE 模型, 进一步验证了本文对 GATNE 的改进是有效的.

4.3.3 R-CWSE 方法与其他 Web API 推荐方法的对比

选取近 3 年在国内重要刊物发表的 Mashup 服务的组件 Web API 推荐方法进行对比, 验证所提出方法的先进性. 推荐质量指标见表 4, 主要包含以下方法.

(1) R-NGCF[37]: 建立 Mashup 服务与 Web API 之间的调用图谱, 提出了一种基于神经图协同过滤的 Web API 推荐方法. 该方法关注节点关系的嵌入传播, 通过挖掘 Mashup 服务和 Web API 之间的高阶交互信息, 为 Mashup 服务推荐组件 Web API.

(2) SRMG[38]: 利用 BERT 为新需求与历史 Mashup 服务生成描述向量, 通过相似度计算获得需求的邻居 Mashup 服务. 建立 Mashup 服务与 Web API 的历史调用图, 使用 GraphGAN 生成 Mashup 服务与 Web API 的嵌入表示, 并获取 Mashup 服务的偏好. 新需求通过相似邻居的调用偏好进行组件 Web API 推荐.

(3) MISR[21]: 构建了一个合并 Web API 和 Mashup 服务之间 3 种类型的交互信息的深度神经网络. 通过学习提取需求与邻居 Mashup 服务、需求与候选 Web API、候选 Web API 中隐藏的结构和特征, 每个候选 Web API 对于 Mashup 服务需求的评分, 实现 Web API 的推荐.

(4) DySR[39]: 将 Mashup 创建问题建模为一个 Web API 集合推荐任务, 以保证候选 Web API 之间的多样性. 提出了一个基于动态图神经网络的模型, 构建语义差异度量方法, 以合理计算需求与候选 Web API 的功能匹配度.

(5) LightGCL[40]: 根据 Mashup 服务和组件 Web API 之间的调用关系构建异质图, 通过近似奇异值分解获取增强图, 分别使用 GCN 学习服务节点的特征表示, 使用对比损失函数进行节点特征优化, 根据标注数据和成对损失函数进行推荐对象的优化.

(6) DLOAR[41]: 将已有 Mashup 服务聚类, 计算 Mashup 服务需求关键词与聚类主题词的相似度, 获取最相似的服务聚类集合. 根据服务描述进行相似度计算, 从聚类中获取最相似 Mashup 服务, 进而获取它们所调用组件服务中最相似的 Web API 完成推荐.

表 4　R-CWSE 推荐方法横向评估

Top-*K*	指标	LightGCL	R-NGCF	DySR	MISR	SRMG	DLOAR	R-CWSE
K=3	*Recall*	0.411	0.421	0.453	0.440	0.431	0.436	**0.478**
	Precision	0.361	0.372	0.405	0.391	0.384	0.386	**0.428**
	NDCG	0.415	0.428	0.465	0.459	0.441	0.446	**0.470**
	ILS	0.574	0.571	0.523	0.547	0.568	0.541	**0.462**
K=5	*Recall*	0.512	0.538	0.579	0.558	0.547	0.552	**0.598**
	Precision	0.295	0.301	0.329	0.319	0.307	0.313	**0.338**
	NDCG	0.443	0.452	0.496	0.484	0.472	0.483	**0.529**
	ILS	0.478	0.469	0.448	0.460	0.462	0.456	**0.422**
K=7	*Recall*	0.594	0.603	0.659	0.634	0.618	0.625	**0.685**
	Precision	0.251	0.253	0.278	0.267	0.260	0.261	**0.292**
	NDCG	0.486	0.488	0.532	0.508	0.490	0.499	**0.577**
	ILS	0.460	0.456	0.426	0.442	0.452	0.437	**0.398**

相比 LightGCL、R-NGCF、DySR、MISR、SRMC 和 DLOAR, R-CWSE 的 *Recall* 指标分别提升 16.05%、12.76%、4.17%、7.95%、10.29% 与 9.24%, 说明本文方法推荐的组件 Web API 中能够涵盖更多的 Mashup 服务构建所需的真实 Web API. 相比其他 4 种方法, 在 *Precision* 指标上, R-CWSE 分别平均提升 16.62%、14.17%、4.46%、8.26%、11.31% 与 10.09%, 因此在本文方法推荐的组件 Web API 中, Mashup 服务构建所需的真实 Web API 比例高于其他方法. 在 *NDCG* 分别平均提升约 17.26%、15.2%、5.57%、8.61%、12.33% 和 10.36%, 说明本文所推荐的正确 Web API 在推荐列表中位置更靠前. 此外, 在多样性方面, R-CWSE 的 *ILS* 指标分别平均降低 15.23%、14.28%、8.22%、11.55%、13.47% 和 10.6%, 验证了本文方法推荐的组件 Web API 的多样性更高.

特别是在 Top-3 的推荐中, 相对其他方法在 *Recall* 指标提升 16.29%、13.6%、5.48%、8.65%、10.78% 和 9.71%, *Precision* 指标提升 18.43%、15.03%、5.63%、9.38%、11.59% 与 10.75%, 高于不同 *K* 值下对应指标的平均提升比例, 这说明本文方法在 Top-3 推荐中性能更佳, 即在推荐 3 个组件 Web API 的情况下, 其能推荐出的正确 Web API 的数量要远高于其他方法. 根据数据集的统计数据可知, Mashup 服务的平均组件 Web API 个数为 2.8, 因此, 本文方法在 Top-3 推荐中的优异性能使其具有较高的实际应用价值.

在上述方法中, LightGCL 与 R-NGCF 的推荐质量较低, 它们仅从 Mashup 服务与 Web API 的组合调用关系建立异质图模型, 通过相似 Mashup 服务进行 Web API 推荐, 在部分组件 Web API 已知的情况下具有较好的推荐效果; 在需要获取全部组件 Web API 时, 推荐质量较差. SRMG 使用 BERT 模型生成服务功能向量进行功能匹配, 通过 Mashup 服务-Web API 异构图获取交互信息, 但推荐评分时没有根据候选 Web API 对需求的潜在功能参与度分配注意力权重, 容易推荐出功能匹配度低的 Web API.

DLOAR 提出了三级相似度匹配机制, 能够获取与 Mashup 服务需求相似度高的组件 Web API. 然而, 其仅考虑功能需求的匹配, 忽视了组件 Web API 间的功能互补, 缺乏对推荐列表整体兼容性的考量, 容易推荐大量功能相似的 Web API. MISR 使用 CNN 提取服务功能特征, 通过相似度为 Mashup 服务需求查找邻居 Mashup 服务, 提升了协同过滤的效果, 该方法利用 3 种类型的交互信息对服务节点向量化, 提高了候选组件对于 Mashup 服务需求的评分合理性, 一定程度上提升了推荐质量.

DySR 明确提出组件 Web API 多样性概念, 从服务功能语义匹配与 Web API 协作两方面进行推荐, 在四种对比方法中推荐质量最高. 相比本文方法, 该方法欠缺对服务潜在应用场景词的考虑, 无法从外界获取更丰富的特征提升功能匹配的合理性, 缺乏 Web API 标签所代表的功能关联考量, 因此, 其推荐质量低于本文方法.

4.3.4 冷启动 Web API 推荐性能评估

冷启动是推荐系统所面临的一个重要问题. 在为 Mashup 服务需求推荐 Web API 时, 冷启动 Web API 是指新发布未参与过真实 Mashup 服务构建的服务.

在理论上, 本文构建的二阶候选组件 Web API 集合 *Cw*-2(*mr*) 中可以有效地将与 Mashup 服务需求 *mr* 相关的冷启动 Web API 纳入到候选组件 Web API 集合中, 从而为冷启动 Web API 提供参与推荐的机会. *Cw*-1(*mr*) 是一阶候选组件 Web API 集合, 该集合采用协同过滤思想构建, 将与需求 *mr* 相似的 Mashup 服务中的真实组件 Web API 放置在集合 *Cw*-1(*mr*) 中, 而 *Cw*-2(*mr*) 在构建时, 是将与 *Cw*-1(*mr*) 中的 Web API 相似度高的服务纳入到该集合中, 因此, 若冷启动 Web API 和 *Cw*-1(*mr*) 中的某个 Web API 相似度高, 该 Web API 可以进入 *Cw*-2(*mr*) 集合, 从而参与最终的 Web API 推荐.

由于 Web API 在发布后, 通常需要经历一段时间的使用才有可能被选中参与 Mashup 服务的构建. 此外, 并非所有新发布的 Web API 都会参与 Mashup 服务的构建. 为了验证本文方法对冷启动 Web API 的推荐能力, 将只参加过一次 Mashup 服务创建的 Web API 作为冷启动推荐对象, 将其所在的 Mashup 服务功能描述作为 Mashup 服务需求.

经过统计, 共计 392 个包含只参与一次 Web API 的 Mashup 服务, 将它们的服务描述作为 Mashup 服务需求, 利用本文方法进行 Web API 推荐, 划分为 4 个数据集, 见表 5. 数据集采取增量式划分, DS2 中的训练集 Mashup 服务即为 DS1 中训练集和测试集 Mashup 服务总和, DS3 与 DS2、DS4 与 DS3 的关系类同.

表 5 冷启动 Web API 数据集

数据	训练集Mashup服务个数	测试集Mashup服务个数	测试集中冷启动Web API个数
DS1	4913	100	100
DS2	5013	100	100
DS3	5113	100	100
DS4	5213	92	92

在指标 *Recall*、*Precision*、*NDCG* 和 *ILS* 基础上, 增加命中率 (hits ratio, *HR*)[4,42]评估模型对冷启动 Web API 的推荐效果, 见公式 (37):

$$HR = \frac{1}{N}\sum_{i=1}^{N} hit(i) \tag{37}$$

其中, N 为参与测试的 Mashup 服务总数. 若冷启动 Web API 作为正例出现在推荐列表, $hit(i)$ 值为 1, 否则为 0.

采用本文方法在 DS1–DS4 数据集进行对比, 推荐质量指标如表 6 所示. 从表中数据统计可知, 在 Top-3 推荐时, 冷启动 Web API 推荐对应的 *Recall* 与 *Precision* 约占非冷启动 Web API 对应 *Recall* 与 *Precision* 的 49.2% 与

50.42%. 冷启动 Web API 质量指标评价值接近于非冷启动 Web API 的一半. 同样, 在 Top-5 和 Top-7 的推荐过程中, 冷启动 Web API 推荐的 *Recall* 分别为非冷启动 Web API 的 47.39% 与 46.53%, *Precision* 分别为非冷启动 Web API 的 46.7% 与 42.62%, 与 Top-3 基本保持一致. 因此, 本文方法较好地解决了冷启动 Web API 的推荐问题.

表 6 DS1–DS4 推荐质量指标

数据	Top-*K*	*Recall*	*Precision*	*NDCG*	*ILS*	*HR*
DS1	*K*=3	0.481	0.426	0.467	0.465	0.151
	K=5	0.597	0.337	0.524	0.427	0.192
	K=7	0.692	0.295	0.580	0.404	0.218
DS2	*K*=3	0.468	0.431	0.462	0.453	0.154
	K=5	0.602	0.341	0.537	0.412	0.191
	K=7	0.689	0.294	0.582	0.388	0.219
DS3	*K*=3	0.449	0.428	0.432	0.464	0.153
	K=5	0.596	0.335	0.524	0.431	0.191
	K=7	0.696	0.288	0.583	0.410	0.218
DS4	*K*=3	0.473	0.433	0.472	0.454	0.159
	K=5	0.606	0.345	0.527	0.438	0.198
	K=7	0.694	0.301	0.580	0.384	0.225

4.3.5 关键实验参数设置

本节对 Web API 推荐过程中的关键参数进行优化设置.

(1) 为 Mashup 服务需求获取 *X* 个相似 Mashup 服务以及 *Y* 个二阶潜在候选组件 Web API 时, *X* 值与 *Y* 值分别取多少推荐质量最高?

X 和 *Y* 的值决定了候选组件集合中 Web API 的数量和质量, 对 Web API 推荐精确度有较大影响. 考虑到数据集中 Mashup 服务包含的 Web API 的数量平均为 2.8 个, 参考与本文方法类似的文献所设置邻居 Mashup 服务的数量, 将 *X* (*Nm*(*mr*) 中服务的数量) 取值设置为 [1, 10]. 为了提高推荐精确度, 候选组件 Web API 集合中的服务数量不宜设置过大, 在此将 *Y* 的取值范围也设置为 [1, 10]. 在 3 种 *K* 值下开展实验. *Recall*、*Precision* 与 *NDCG* 指标三维曲面趋势参见图 8.

在 *X* 值不变的情况下, 随 *Y* 值上升, 或在 *Y* 值不变的情况下, 随 *X* 值上升, 推荐质量均呈先上升, 后下降的趋势. 在不同的 Top-*K* 推荐场景下, *Precision* 由推荐出来正确组件 Web API 个数与 *K* 值的比值确定, 而 *Recall* 则是由推荐正确的组件 Web API 个数与总服务个数的比值确定, *NDCG* 侧重正确推荐的 Web API 在推荐列表中的排序. 在同一组 (*K*, *X*, *Y*) 取值下, 正确推荐的组件 Web API 数量是确定的, 因此, *Precision*、*Recall* 和 *NDCG* 在同一个 *K* 取值的推荐场景下, 三者的取值增减随 (*X*, *Y*) 变化同步, 曲面图在同一个 (*X*, *Y*) 坐标点到达拐点. 经过统计, *K*=3、5、7 时, Web API 推荐效果分别在 (4, 5)、(6, 3)、(5, 4) 处取得最佳质量

从图 8 中的曲面变化趋势可以看出, *X* 值的变化相对于 *Y* 值的变化会对推荐效果的影响更大, 说明 *Nm*(*mr*) 中服务数量对推荐质量会产生较大影响. 其原因是 *Nm*(*mr*) 是与 Mashup 服务需求相似的 Mashup 服务集合, *X* 值决定了 *Nm*(*mr*) 中服务的数量, 从而影响了 *Nm*(*mr*) 中服务与 Mashup 服务需求的相似度, 从协同过滤角度, 选择与推荐目标相似度越高的参照对象, 推荐效果越好, 因此 *X* 值的变化对于推荐效果会更敏感.

此外, *X* 和 *Y* 值决定了最终候选组件 Web API 的数量和质量. 当 *X* 值或 *Y* 值过低, 则 Mashup 服务需求的邻居服务与候选组件 Web API 数量较少, 导致无法将真实组件 Web API 加入候选集合, 也不利于潜在联合词的提取, 从而影响推荐质量. 当 *X* 值或 *Y* 值过高, 会将与 Mashup 服务需求契合度低的 Web API 加入到候选组件 Web API 集合, 同样影响潜在联合词提取和异质关联兼容的计算质量, 从而降低推荐质量.

(2) 单个标签对应潜在联合词融入数量为多少时推荐质量最高?

潜在联合词为 Mashup 服务和 Web API 的功能向量补充了应用场景特征. 为验证潜在联合词数量对推荐质量影响, 将标签对应特征词的提取数量 *n* 设置为区间 [0, 10], 在 3 种 *K* 值下开展实验, *Recall*、*Precision* 和 *NDCG* 的指标折线趋势参见图 9–图 11, 图中横坐标为 *n* 的值, 纵坐标为评价指标值.

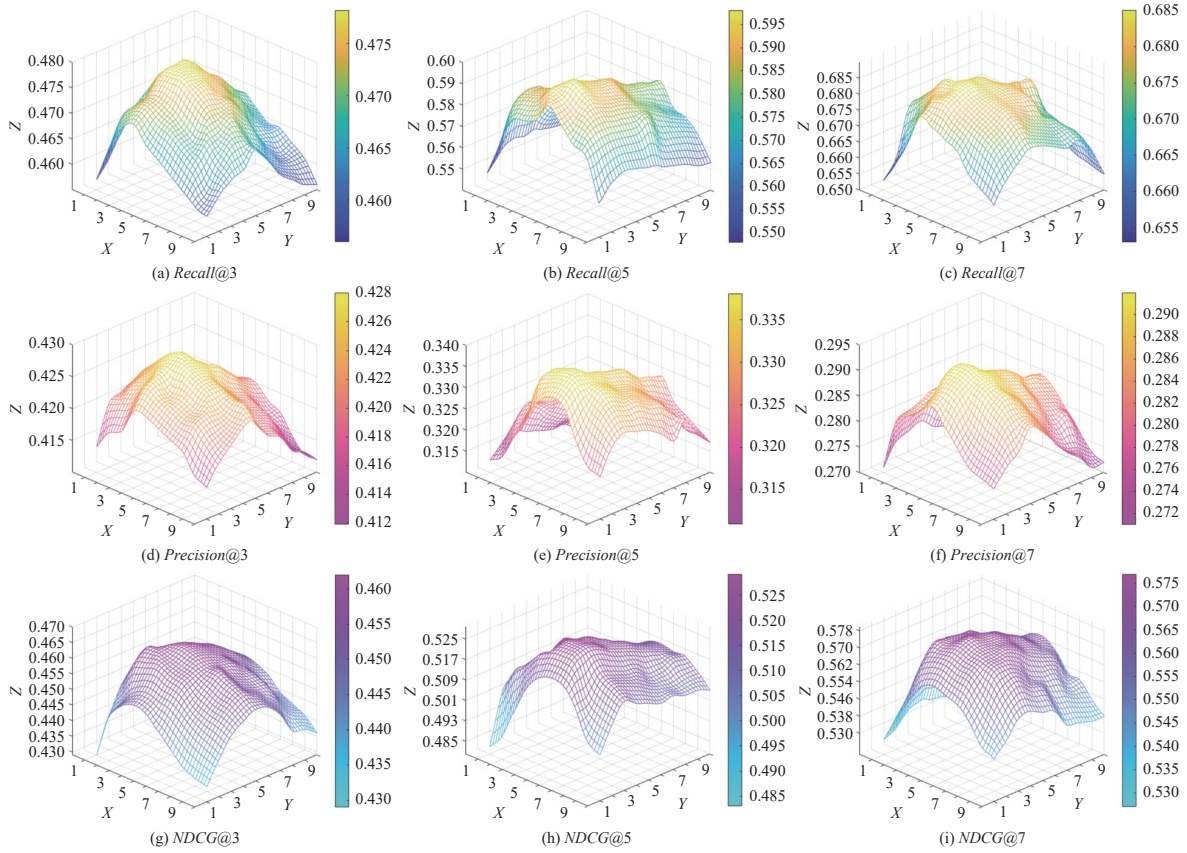

图 8　采样邻居数量指标图

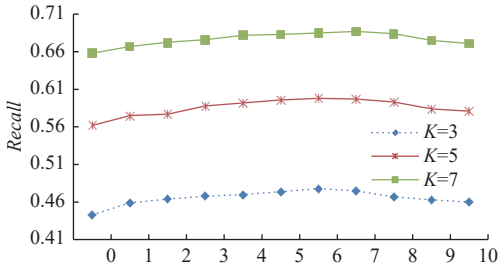

图 9　不同 *K* 值下的 *Recall* 对比

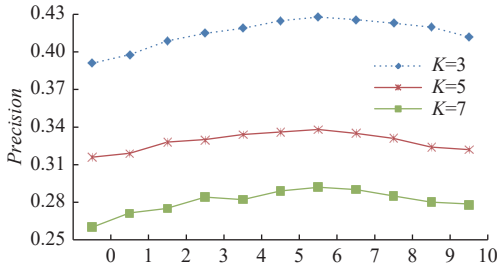

图 10　不同 *K* 值下的 *Precision* 对比

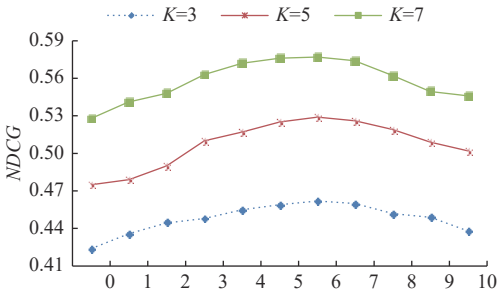

图 11　不同 *K* 值下的 *NDCG* 对比

由图 9–图 11 可见, 除 *K*=7 时 *Recall* 与 *Precision* 分别在 6 个词与 4 个词中出现轻微波动外, 随潜在特征词融入数量增加, *Recall*、*Precision* 与 *NDCG* 指标随之提升, 在词数为 6 时到达峰值. 当超过 6 个词, 模型效果会出现下降趋势. 上述趋势与服务描述文本较短有关, 当融入的场景特征词过多时, 会影响功能特征的语义表达强度. 基于此, 将潜在特征词提取数量统一设置为 6.

5 结束语

为提升 Mashup 服务的组件 Web API 推荐质量, 提出一种融合潜在联合词与异质关联兼容的 Web API 推荐方法. 设计了融入场景契合度的服务描述特征词提取模型 SYAKE, 为 Mashup 服务需求和 Web API 提取表示应用场景的潜在联合词, 借助潜在联合词提升了功能匹配的计算精度. 在此基础上, 借助协同过滤思想, 为 Mashup 服务需求构建高质量的候选组件 Web API 集合. 构建了一种利用异质关联兼容提升 Web API 推荐质量的方法, 相比当前流行的以协作兼容提升 Web API 推荐的方法, 异质关联兼容能够有效提升推荐质量和多样性. 实验证明, 本文方法在推荐的精确度、召回率和多样性等指标上显著优于对比方法.

未来研究工作主要是挖掘更多类型的 Web API 关联, 如 API 调用者关联、地理位置关联等, 以进一步改善关联兼容对推荐质量的提升效果. 需要注意的是在后续 Web API 的关联挖掘中, 需要分析不同类型的关联对 Web API 推荐质量评价指标的差异化影响, 以期将有利于提高推荐质量的关联纳入到推荐方法中. 同时, 关联密度对推荐质量的影响问题也需要加以关注和探讨.

References:

[1] Sang CY, Deng XY, Liao SG. Mashup-oriented Web API recommendation via full-text semantic mining of developer requirements. IEEE Trans. on Services Computing, 2023, 16(4): 2755–2768. [doi: 10.1109/TSC.2023.3245652]

[2] Lu JW, Zhao W, Zhang YM, Liang QH, Xiao G. TWE-NMF topic model-based approach for Mashup service clustering. Ruan Jian Xue Bao/Journal of Software, 2023, 34(6): 2727–2748 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6508.htm [doi: 10.13328/j.cnki.jos.006508]

[3] Lu JW, Wu H, Zhang YM, Liang QH, Xiao G. Mashup service clustering method via integrating functional semantic association calculation and density peak detection. Chinese Journal of Computers, 2021, 44(7): 1501–1516 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.01501]

[4] Qi LY, Song HB, Zhang XY, Srivastava G, Xu XL, Yu S. Compatibility-aware Web API recommendation for Mashup creation via textual description mining. ACM Trans. on Multimedia Computing, Communications, and Applications, 2021, 17(1s): 20. [doi: 10.1145/3417293]

[5] Qi LY, He Q, Chen FF, Zhang XY, Dou WC, Ni Q. Data-driven Web APIs recommendation for building Web applications. IEEE Trans. on Big Data, 2022, 8(3): 685–698. [doi: 10.1109/TBDATA.2020.2975587]

[6] Chen NX, Cardozo N, Clarke S. Goal-driven service composition in mobile and pervasive computing. IEEE Trans. on Services Computing, 2018, 11(1): 49–62. [doi: 10.1109/TSC.2016.2533348]

[7] Qi LY, He Q, Chen FF, Dou WC, Wan SH, Zhang XY, Xu XL. Finding all you need: Web APIs recommendation in Web of things through keywords search. IEEE Trans. on Computational Social Systems, 2019, 6(5): 1063–1072. [doi: 10.1109/TCSS.2019.2906925]

[8] Gu Q, Cao J, Peng QY. Service package recommendation for Mashup creation via Mashup textual description mining. In: Proc. of the 2016 IEEE Int'l Conf. on Web Services. San Francisco: IEEE, 2016. 452–459. [doi: 10.1109/ICWS.2016.65]

[9] Li HC, Liu JX, Cao BQ, Tang MD, Liu XQ, Li B. Integrating tag, topic, co-occurrence, and popularity to recommend Web APIs for Mashup creation. In: Proc. of the 14th IEEE Int'l Conf. on Services Computing. Honolulu: IEEE, 2017. 84–91. [doi: 10.1109/SCC.2017.19]

[10] Zhang N, Wang J, Ma YT. Mining domain knowledge on service goals from textual service descriptions. IEEE Trans. on Services Computing, 2020, 13(3): 488–502. [doi: 10.1109/TSC.2017.2693147]

[11] Tang MD, Xie FF, Liang W, Xia YM, Li KC. Predicting new composition relations between Web services via link analysis. Int'l Journal of Computational Science and Engineering, 2019, 20(1): 88–101. [doi: 10.1504/IJCSE.2019.103256]

[12] Su C, Huang DL. Hybrid recommender system based on deep learning model. Int'l Journal of Performability Engineering, 2020, 16(1): 118–129. [doi: 10.23940/ijpe.20.01.p13.118129]

[13] Botangen KA, Yu J, Sheng QZ, Han YB, Yongchareon S. Geographic-aware collaborative filtering for Web service recommendation.

Expert Systems with Applications, 2020, 151: 113347. [doi: 10.1016/j.eswa.2020.113347]

[14] Yao LN, Wang XZ, Sheng QZ, Benatallah B, Huang CR. Mashup recommendation by regularizing matrix factorization with API co-invocations. IEEE Trans. on Services Computing, 2021, 14(2): 502–515. [doi: 10.1109/TSC.2018.2803171]

[15] Tang B, Tang MD, Xia YM, Hsieh MY. Composition pattern-aware Web service recommendation based on depth factorisation machine. Connection Science, 2021, 33(4): 870–890. [doi: 10.1080/09540091.2021.1911933]

[16] Cao BQ, Liu JX, Wen YP, Li HT, Xiao QX, Chen JJ. QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications. Journal of Parallel and Distributed Computing, 2019, 132: 177–189. [doi: 10.1016/j.jpdc.2018.04.002]

[17] Nguyen M, Yu J, Nguyen T, Han YB. Attentional matrix factorization with context and co-invocation for service recommendation. Expert Systems with Applications, 2021, 186: 115698. [doi: 10.1016/j.eswa.2021.115698]

[18] Kang GS, Liu JX, Xiao Y, Cao BQ, Xu Y, Cao ML. Neural and attentional factorization machine-based Web API recommendation for Mashup development. IEEE Trans. on Network and Service Management, 2021, 18(4): 4183–4196. [doi: 10.1109/TNSM.2021.3125028]

[19] Xiao Y, Liu JX, Kang GS, Hu R, Cao BQ, Cao YC, Shi M. Structure reinforcing and attribute weakening network based API recommendation approach for Mashup creation. In: Proc. of the 2020 IEEE Int'l Conf. on Web Services. Beijing: IEEE, 2020. 541–548. [doi: 10.1109/ICWS49710.2020.00078]

[20] Liu MY, Zhu YQ, Xu HC, Tu ZY, Wang ZJ. T2L2: A tiny three linear layers model for service Mashup creation. In: Proc. of the 19th Int'l Conf. on Service-oriented Computing. Berlin: Springer, 2021. 317–331. [doi: 10.1007/978-3-030-91431-8_20]

[21] Ma YT, Geng X, Wang J. A deep neural network with multiplex interactions for cold-start service recommendation. IEEE Trans. on Engineering Management, 2021, 68(1): 105–119. [doi: 10.1109/tem.2019.2961376]

[22] Yan RY, Fan YS, Zhang J, Zhang JQ, Lin HZ. Service recommendation for composition creation based on collaborative attention convolutional network. In: Proc. of the 2021 IEEE Int'l Conf. on Web Services. Chicago: IEEE, 2021. 397–405. [doi: 10.1109/ICWS53863.2021.00059]

[23] Cao BQ, Peng M, Qing YY, Liu JX, Kang GS, Li B, Fletcher KK. Web API recommendation via combining graph attention representation and deep factorization machines quality prediction. Concurrency and Computation: Practice and Experience, 2022, 34(21): e7069. [doi: 10.1002/cpe.7069]

[24] Huang DL, Tong XL, Yang HD. Web service recommendation based on graph attention network (GAT-WSR). In: Proc. of the 2022 Int'l Conf. on Computer Communication and Informatics. Coimbatore: IEEE, 2022. 1–5. [doi: 10.1109/ICCCI54379.2022.9740941]

[25] Almarimi N, Ouni A, Bouktif S, Mkaouer MW, Kula RG, Saied MA. Web service API recommendation for automated Mashup creation using multi-objective evolutionary search. Applied Soft Computing, 2019, 85: 105830. [doi: 10.1016/j.asoc.2019.105830]

[26] Gong WW, Zhang XY, Chen YF, He Q, Beheshti A, Xu XL, Yan C, Qi LY. DAWAR: Diversity-aware Web APIs recommendation for Mashup creation based on correlation graph. In: Proc. of the 45th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Madrid: ACM, 2022. 395–404. [doi: 10.1145/3477495.3531962]

[27] Campos R, Mangaravite V, Pasquali A, Jorge AM, Nunes C, Jatowt A. A text feature based automatic keyword extraction method for single documents. In: Proc. of the 40th European on Conf. on Information Retrieval. Grenoble: Springer, 2018. 684–691. [doi: 10.1007/978-3-319-76941-7_63]

[28] Liu JX, Shi M, Zhou D, Tang MD, Zhang TT. Topic model based tag recommendation method for Mashups. Chinese Journal of Computers, 2017, 40(2): 520–534 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2017.00520]

[29] Gao TY, Yao XC, Chen DQ. SimCSE: Simple contrastive learning of sentence embeddings. In: Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing. Punta Cana: ACL, 2021. 6894–6910. [doi: 10.18653/v1/2021.emnlp-main.552]

[30] Cen YK, Zou X, Zhang JW, Yang HX, Zhou JR, Tang J. Representation learning for attributed multiplex heterogeneous network. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Anchorage: ACM, 2019. 1358–1368. [doi: 10.1145/3292500.3330964]

[31] Dong YX, Chawla NV, Swami A. Metapath2vec: Scalable representation learning for heterogeneous networks. In: Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Halifax: ACM, 2017. 135–144. [doi: 10.1145/3097983.3098036]

[32] Jiang R, Han SS, Yu YM, Ding WP. An access control model for medical big data based on clustering and risk. Information Sciences, 2023, 621: 691–707. [doi: 10.1016/j.ins.2022.11.102]

[33] Wu SQ, Shen SG, Xu XL, Chen Y, Zhou XK, Liu DN, Xue X, Qi LY. Popularity-aware and diverse Web APIs recommendation based on correlation graph. IEEE Trans. on Computational Social Systems, 2023, 10(2): 771–782. [doi: 10.1109/TCSS.2022.3168595]

[34] Zhao Y, Qiao Y, He KQ. A novel tagging augmented LDA model for clustering. Int'l Journal of Web Services Research, 2019, 16(3): 59–77. [doi: 10.4018/IJWSR.2019070104]

[35] Briskilal J, Subalalitha CN. An ensemble model for classifying idioms and literal texts using BERT and RoBERTa. Information Processing & Management, 2022, 59(1): 102756. [doi: 10.1016/J.IPM.2021.102756]

[36] Yin JH, Wang JY. A Dirichlet multinomial mixture model-based approach for short text clustering. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM, 2014. 233–242. [doi: 10.1145/2623330.2623715]

[37] Lian SX, Tang MD. API recommendation for Mashup creation based on neural graph collaborative filtering. Connection Science, 2022, 34(1): 124–138. [doi: 10.1080/09540091.2021.1974819]

[38] Yu T, Yu DJ, Wang DJ, Hu XY. Web service recommendation for Mashup creation based on graph network. The Journal of Supercomputing, 2023, 79(8): 8993–9020. [doi: 10.1007/s11227-022-05011-3]

[39] Liu MY, Tu ZY, Xu HC, Xu XF, Wang ZJ. DySR: A dynamic graph neural network based service bundle recommendation model for Mashup creation. IEEE Trans. on Services Computing, 2023, 16(4): 2592–2605. [doi: 10.1109/TSC.2023.3234293]

[40] Cai XH, Huang C, Xia LH, Ren XB. LightGCL: Simple yet effective graph contrastive learning for recommendation. In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: ICLR, 2023.

[41] Wang Y, Chen JW, Huang Q, Xia X, Jiang B. Deep learning-based open API recommendation for Mashup development. Science China Information Sciences, 2023, 66(7): 172102. [doi: 10.1007/s11432-021-3531-0]

[42] Wu SW, Sun F, Zhang WT, Xie X, Cui B. Graph neural networks in recommender systems: A survey. ACM Computing Surveys, 2023, 55(5): 97. [doi: 10.1145/3535101]

附中文参考文献:

[2] 陆佳炜, 赵伟, 张元鸣, 梁倩卉, 肖刚. 基于 TWE-NMF 主题模型的 Mashup 服务聚类方法. 软件学报, 2023, 34(6): 2727–2748. [doi: 10.13328/j.cnki.jos.006508]

[3] 陆佳炜, 吴涵, 张元鸣, 梁倩卉, 肖刚. 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. 计算机学报, 2021, 44(7): 1501–1516. [doi: 10.11897/SP.J.1016.2021.01501]

[28] 刘建勋, 石敏, 周栋, 唐明董, 张婷婷. 基于主题模型的 Mashup 标签推荐方法. 计算机学报, 2017, 40(2): 520–534. [doi: 10.11897/SP.J.1016.2017.00520]

胡强(1980－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为服务计算, 人工智能.

李浩杰(1990－), 男, 博士生, 主要研究领域为图数据挖掘, 自然语言处理.

綦浩泉(1998－), 男, 硕士生, CCF 学生会员, 主要研究领域为云计算, 自然语言处理.

杜军威(1974－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为人工智能, 软件工程.