功能互补关系增强的云 API 推荐方法

陈真[1,2]，陈文辉[1]，刘啸威[1]，尤殿龙[1,2]，刘林林[3]，申利民[1,2]

（1. 燕山大学信息科学与工程学院，河北 秦皇岛 066004；
2. 燕山大学河北省计算机虚拟技术与系统集成重点实验室，河北 秦皇岛 066004；
3. 中国科学院文献情报中心，北京 100190）

摘 要：当前云 API 推荐方法主要采用相似性计算或者利用 Mashup 的历史调用来生成推荐结果，忽略了 Mashup 与云 API 之间有益的功能互补关系。针对上述问题，提出一种基于功能互补关系增强的云 API 推荐方法。首先，利用标签共现对功能互补关系进行刻画。然后，计算功能互补得分来刻画云 API 和 Mashup 之间的功能互补程度，学习功能互补向量来刻画云 API 和 Mashup 之间的潜在功能互补关系。在此基础上，将功能互补得分和功能互补向量嵌入云 API 推荐模型中，使功能互补关系在推荐云 API 的过程中起到关键性的作用。在真实世界云 API 数据集上进行实验，所提方法在稀疏场景下的 AUC、F1、HR@5 指标上平均提升了 2.32%、1.86%、9.15%，最终验证了所提方法可以在提高云 API 推荐结果准确性的同时，提升对长尾云 API 的推荐性能。

关键词：云 API 推荐；功能互补；标签共现；长尾云 API

中图分类号：TP393

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023093

Functional complementarity relationship enhanced cloud API recommendation method

CHEN Zhen[1,2], CHEN Wenhui[1], LIU Xiaowei[1], YOU Dianlong[1,2], LIU Linlin[3], SHEN Limin[1,2]

1. School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
2. Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Yanshan University, Qinhuangdao 066004, China
3. National Science Library, Chinese Academy of Sciences, Beijing 100190, China

Abstract: Current cloud application program interface (API) recommendation methods mainly use similarity calculation or historical calls of Mashup to generate recommendation results, while ignoring the beneficial functional complementarity (FC) between Mashup and cloud API. To address the above issue, a FC relationship enhanced cloud API recommendation approach was proposed. Firstly, label co-occurrence was applied to describe the FC relationship. Then, the FC score was calculated to describe the degree of FC between the cloud API and the Mashup, and the FC vector was learned to describe the potential FC relationship. Based on this, FC scores and FC vectors were embedded into the cloud API recommendation model, so that FC relationship played a key role in the cloud API recommendation process. Experiments were conducted on real-world cloud API datasets, and the AUC, F1 and HR@5 of the proposed approach improved by an average of 2.32%, 1.86% and 9.15%, respectively, in the sparse scenario. Finally, the proposed approach can improve the accuracy of cloud API recommendation results, while improving the recommendation performance of long-tail cloud API.

Keywords: cloud API recommendation, functional complementarity, label co-occurrence, long tail cloud API

收稿日期：2023−02−07；**修回日期：**2023−04−18

基金项目：国家自然科学基金资助项目（No.62102348, No.62276226）；河北省自然科学基金资助项目（No.F2022203012, No.F2021203038）；河北省创新能力提升计划基金资助项目（No.22567626H）；河北省研究生创新基金资助项目（No.CXZZSS2023048）

Foundation Items: The National Natural Science Foundation of China (No.62102348, No.62276226), The Natural Science Foundation of Hebei Province (No.F2022203012, No.F2021203038), Innovation Capability Improvement Plan Project of Hebei Province (No.22567626H), Graduate Innovation Funding Project of Hebei Province (No.CXZZSS2023048)

0 引言

过去 50 年软件工程的发展史已经证明，复用技术是解决软件危机、提高软件生产效率和质量，推进软件产业工程化和工业化的现实可行途径。为了适应动态开放网络环境下的软件复用，人们提出了面向服务体系架构（SOA, service-oriented architecture）[1]，倡导通过网络以服务混搭（Mashup，即混搭一个或多个云 API 的应用程序）的方式进行软件开发[2]，采用中立的访问协议和显式的服务契约实现服务的提供者和使用者解耦，深刻改变了软件的形态、开发范式和运行模式。软件形态由单一的盒装源代码演变成源代码与服务的混合体；软件开发方式从企业和组织内应用程序集成发展成由服务连接的跨界异构业务功能协作与数据交换；软件运行方式从封闭孤立系统转向由服务合成的开放协同系统。

进入万物互联的云时代，开放成为了发展趋势，越来越多的产品走向开放化。云应用程序接口（API, application program interface）作为能力和数据开放的核心载体，成为能力复制、数据输出和服务交付的最佳实践。区别于本地 API，云 API 对服务端函数库进行封装，从网络服务的层面体现出 API 服务。由于云 API 使用网络协议进行数据传输，与开发语言无关，开发者能够通过相应 HTTP 请求使用云 API 提供的服务，进而有效支撑面向服务的软件开发，提高软件生产效率和质量。云 API 作为实现 SOA 的新技术，凭借松耦合、轻量级和易组合等优势将迄今为止一直隐藏在企业与组织背后的数据与计算能力便捷地提供给合作伙伴和对其感兴趣的开发人员，辅助企业和组织提升自主创新服务的能力，以低成本和快速度响应市场需求，以跨界融合方式打造一个更加开放的产业生态。与此同时，随着微服务架构的流行，云 API 的使用也变得越来越普遍。据统计[3]，当前整个 Web 系统超过 83%的流量都是通过云 API 访问的；超过 44%的企业正在建造和维护超过 100 个或更多的云 API，云 API 流量正在以极快的速度增长；截至 2022 年 1 月，全球最大的云 API 聚合服务平台 Programmable Web 已提供 24 701 个云 API。

然而，动态开放网络中云 API 的爆发式增长，导致了软件开发者在选择云 API 时面临云 API 过载问题。数量庞大的云 API 在给开发人员提供更多选择的同时，也使开发人员很难快速识别和选择满足其特定功能需求与潜在兴趣的云 API。为了解决云 API 选择困难问题，研究人员提出了许多推荐算法，通过计算相似度或利用交互记录向开发者推荐符合其需求的云 API。尽管现有研究在云 API 推荐方面做了很多工作[4-5]，但仍然存在以下不足。

1) 云 API 与 Mashup 之间的功能互补关系没有得到充分重视。现有云 API 推荐方法很少关注被推荐云 API 与 Mashup 功能需求之间的互补关系。值得注意的是，当开发者构建 Mashup 时，对 Mashup 的功能起到补充作用的云 API 会更加受到开发者的青睐，但现有推荐方法并没有注意到云 API 与 Mashup 之间的功能互补关系。

2) 长尾云 API 的推荐准确率低。因为开发者的个性化需求，云 API 生态中存在大量长尾云 API。由于长尾云 API 数量多、被调用次数少的特点，现有推荐方法很难充分学习到长尾云 API 的信息，进而导致现有方法对长尾云 API 的推荐准确率较低。

为了解决上述问题，本文提出功能互补关系增强的云 API 推荐方法。首先，使用真实云 API 生态中的标签特征表示云 API 和 Mashup 的功能，利用标签之间的共现次数刻画云 API 和 Mashup 之间的功能互补关系。然后，基于标签共现矩阵设计功能互补程度刻画算法计算云 API 和 Mashup 之间的功能互补得分；基于标签共现关系图，利用图卷积网络计算表征云 API 和 Mashup 之间潜在功能互补关系的功能互补向量。最后，将云 API 和 Mashup 之间的功能互补得分和功能互补向量作为新特征嵌入推荐模型中，使功能互补关系在云 API 推荐过程中起到关键性作用，以此在提高云 API 推荐准确率的同时提升对长尾云 API 的推荐效果。

总体来说，本文的主要贡献如下。

1) 提出了基于标签共现刻画云 API 和 Mashup 之间互补关系的方法。基于标签特征能够反映功能的事实，提出通过标签的共现频率挖掘不同功能之间的互补关系，进而刻画云 API 和 Mashup 之间的功能互补关系。

2) 提出了功能互补关系增强的云 API 推荐方法。基于标签共现矩阵设计功能互补程度刻画算法，计算云 API 和 Mashup 之间的功能互补得分，用数值直观表示功能互补程度的强弱。基于标签共现关系图采用图卷积网络学习功能互补向量，表征云 API 和 Mashup 之间的潜在功能互补关系。功能互

补得分和功能互补向量作为新特征嵌入云 API 推荐模型，使功能互补关系在推荐过程中发挥关键作用。

3) 在真实世界云 API 数据集上进行大量实验。所提方法的 AUC、F1、HR@5 指标在稀疏场景下平均提升了 2.32%、1.86%、9.15%，在长尾场景下平均提升了 13.68%、13.61%、19.78%，因而所提方法具备低耦合的特性，能够嵌入目前主流的推荐模型中，可有效提高数据稀疏场景和长尾场景下的推荐效果。

1　相关工作

1.1　互补推荐

在经济学中，互补商品被定义为经常与另一种商品一起被消费的商品[6]。具体来说，若 2 个商品之间的需求交叉弹性系数为负值，则一种商品的价格下降会增加其互补商品的需求。购买一种商品的互补商品与购买该商品本身是正相关的。在这一理论指导下，根据共同购买频率来识别 2 个商品是否为互补商品成为了绝大多数研究的第一经验法则。例如，手机和手机壳通常被很多用户一同购买，两者具有较高的共同购买频率，因此可以认为两者是互补关系。互补推荐的核心思想就是推荐互相起到补充作用的物品或项目，给出精准且多元的结果，提升推荐准确率和用户体验。

目前，电商领域已有推荐方法尝试引入互补的思想，结合物品的互补关系给出推荐结果[7]。文献[8]考虑针对不同种类的物品，互补关系的表示应从不同属性出发，例如，衣服的互补属性更多从风格和用途考虑，图书的互补属性更多从内容和类别考虑。该研究提出在计算物品的互补程度时，对于不同物品使用不同的属性。文献[9]提出生成物品在语义空间内的映射向量，通过语义空间内的距离来预测不同物品之间是否存在互补关系，以解决推荐过程中的冷启动问题。文献[10]不仅考虑了物品之间的互补关系，还考虑了物品之间的替代关系，通过被推荐物品的文本描述、品牌和类别等特征信息，对物品实现主题建模，预测物品之间的替代关系和互补关系，并给出替代品和互补品的 2 种推荐列表。文献[11]设计了一种无监督学习模型，先向模型中输入大量真实的穿搭图片，让模型学习图片中不同服饰的共现性，以共现频率来表示互补关系的强弱，然后采用生成对抗网络给出互补推荐结果。

当前，针对电商领域的物品互补推荐已有一定研究，但是在云 API 推荐领域，对于互补关系的研究关注度较低，而功能之间的互补关系正是开发者在 Mashup 开发过程中选择云 API 时重要的因素。本文通过引入云 API 和 Mashup 之间的功能互补关系，为解决云 API 推荐领域难以回避的长尾问题提供新的研究思路。

1.2　云 API 推荐

协同过滤技术利用相似Mashup之间具有相似调用历史记录的特点，学习 Mashup 开发者对云 API 的潜在调用偏好。该技术已被工业界广泛采用，同时人们也在不断提出新的基于协同过滤技术的云 API 推荐方法。文献[12]提出了一种基于矩阵分解的协同过滤推荐方法，该方法基于协同过滤技术计算 Mashup 开发者调用云 API 的偏好时，综合考虑地理信息与描述信息对推荐结果的影响。文献[13]提出了一种基于深度学习的协同过滤方法，该方法使用交互信息和特征描述信息对 Mashup 开发者的偏好进行建模，Mashup 和云 API 之间的历史调用记录以及它们的特征被输入深度神经网络中，训练结果用于描述 Mashup 和云 API 之间的相关性。文献[14]提出了一种带有隐性相关正则化的概率矩阵分解方法，以解决推荐问题并增强推荐的多样性。文献[15]考虑到云 API 的特性和开发者的需求可能会随着时间变化，设计了一种具有时间感知的协同过滤云 API 推荐方法。值得注意的是，动态开放的云 API 生态中每天都会有新的云 API 加入，这些新加入的云 API 没有交互记录或者交互记录很少，进而产生数据稀疏和长尾问题。而基于协同过滤的云 API 推荐方法的性能强烈依赖大量的历史交互记录，所以难以回避的数据稀疏和长尾问题限制了该方法的性能表现。

基于内容的云 API 推荐方法通过云 API 的大量描述信息以及开发者的喜好来表示云 API。在推荐过程中，使用云 API 与需求模型之间的相关性进行推荐。文献[16]提出了一种通过云 API 调用之间的组合关系进行聚类，自动提取调用需求的方法。文献[17]将 Mashup 构建表述为一个多目标云 API 组合问题，并使用非支配排序遗传算法作为搜索方法来提取最佳的云 API 集合。文献[18]定义了一种语义模型，通过使用频繁概念结构来计算云 API 的语义模型，以便找到与其语义模型匹配的云 API 集合。文献[19]提出了基于语义的云 API 知识图谱推荐模型，该模型捕获了云 API、Mashup 以及开发者之间的关系，允许用户个性化定制自己的偏好信息，通过定制的偏好信息向其推荐云 API。

通过云 API 领域主流推荐算法分析可见，现有方法忽略了云 API 与 Mashup 之间的互补关系，而 Mashup 开发者在选择云 API 时却非常看重其是否对 Mashup 的功能需求起到补充作用。不同于现有方法，本文基于标签共现刻画功能互补关系，并利用功能互补关系来为开发者进行个性化云 API 推荐。

2 研究动机与问题定义

2.1 研究动机

在真实云 API 生态 Programmable Web 中的数据分析发现，标签特征隐含着云 API 与 Mashup 之间的功能互补关系。基于标签共现的功能互补关系示意如表 1 所示，通过对一些常见云 API 历史使用情况分析发现，表示功能属性的标签之间存在一些较高的共现组合。例如，PayPal 是一款具有支付功能的云 API，在调用 PayPal 的 Mashup 中，电商类占比为 37.14%，即开发电商类 Mashup 时在很大程度上会调用具有支付功能的云 API。如上文所述，经济学领域根据共同购买频率来识别 2 个商品是否为互补商品，同样地，可以根据标签共现频率判断云 API 与 Mashup 是否功能互补。因此，可进一步判断这些云 API 对调用它的 Mashup 功能起到补充作用，可作为软件主体功能的一部分，以实现一个完整的应用程序。综上，功能互补在云 API 的选择过程中是一个较重要的因素，在构建云 API 推荐模型时也应该考虑到这一因素，但是现有推荐方法尚未注意到这一有益的功能互补关系。本文提出利用功能互补这一特性来增强云 API 推荐，提升推荐结果的准确性。

表 1 基于标签共现的功能互补关系示意

云 API	云 API 标签	Mashup 标签	Mashup 占比
PayPal	支付	电商	37.14%
Last.FM	播客	音乐	74.03%
Twitter	博客	社交	51.63%
YouTube	视频	媒体	71.23%

2.2 问题定义

云 API 推荐问题的核心任务是 Mashup 调用云 API 概率预测问题，具体说明如下。

定义 Mashup 集合 $M=\{m_1,m_2,\cdots\}$，云 API 集合 $A=\{a_1,a_2,\cdots\}$，云 API 与 Mashup 的特征集合 $F=\{(\mathrm{mf}_1,\mathrm{mf}_2,\cdots,\mathrm{af}_1,\mathrm{af}_2,\cdots)\}$，其中，mf 表示 Mashup 特征，af 表示云 API 特征。定义 Mashup 云与 API 之间的互补关系集合 $C=\{C_{m,a}\mid m\in M,a\in A\}$，其中，$C_{m,a}$ 表示 Mashup m 和云 API a 的之间互补关系。因此，功能互补关系 C 增强的 Mashup 调用云 API 概率预测问题可定义为

$$\hat{y}_{m,a}=f(m,a\mid F,C) \tag{1}$$

其中，$\hat{y}_{m,a}\in[0,1]$ 是 Mashup m 调用云 API a 的调用概率预测值。

在此基础上，对于任意 $m\in M$，计算被 m 调用概率最大的 k 个云 API，作为 m 的推荐结果，实现功能互补关系增强的云 API 推荐。

3 方法设计

3.1 总体框架

功能互补关系增强的云 API 推荐模型总体框架如图 1 所示，主要包括以下 4 个部分。

1) 基于标签共现的功能互补关系表示。根据 Mashup 对云 API 的调用记录，构建标签共现矩阵和标签共现关系图，利用标签之间的共现次数表示 2 个标签所代表的功能之间的功能互补关系。

2) 基于标签共现的 Mashup 与云 API 功能互补程度刻画。标签共现矩阵表示任意 2 个功能之间的功能互补关系。Mashup 和云 API 通常包括多个功能，利用功能之间的功能互补关系计算 Mashup 和云 API 之间的功能互补得分，用以表示 Mashup 和云 API 之间的功能互补程度。

3) 基于图卷积网络的 Mashup 与云 API 功能互补向量学习。标签共现关系图表示功能之间存在的潜在功能互补关系。从图的视角采用图卷积网络，通过集成多个互补邻居节点的信息学习云 API 和 Mashup 的功能互补向量。

4) 功能互补得分和功能互补向量嵌入的云 API 推荐。将互补得分和功能互补向量嵌入推荐模型的交互训练中，进而利用功能互补关系增强推荐模型的学习效果。

3.2 基于标签共现的功能互补关系表示

为了表示功能互补关系，首先采用标签表征云 API 和 Mashup 的功能；其次物品之间的频繁共现意味着 2 个物品之间存在互补性，如果共现次数越多，那么 2 个物品之间的互补关系越强。类似地，根据云 API 和 Mashup 标签之间的共现次数，可以表示功能之间的互补关系。首先，基于 Mashup 对云 API 的调用记录，计算得到 Mashup 与云 API 之间不同标签组合的共现次数。例如，当 Mashup m 调

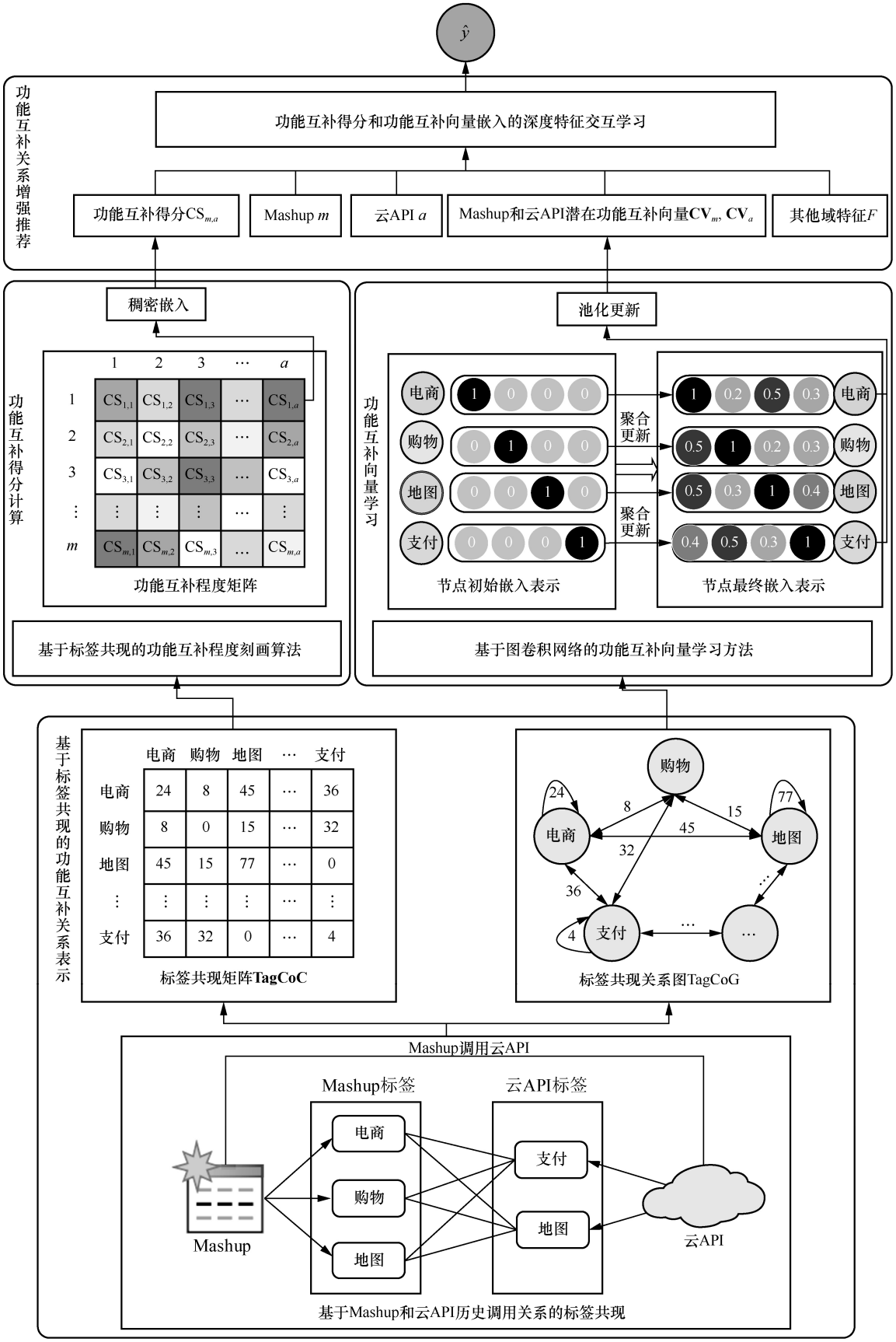

图 1　功能互补关系增强的云 API 推荐模型总体框架

用了云 API a 时，两者所具有的标签可以视为共现一次。通过统计所有 Mashup 与云 API 之间的历史调用关系，可得到所有组合过的标签共现次数。标签共现次数越多，其代表的功能之间的互补关系越紧密。

为了计算标签共现频率，设 Mashup 与云 API 的标签集合为 $T=\{t_i \mid i=1,2,\cdots,N\}$。令 T_m 为 Mashup 标签集合，T_a 为云 API 标签集合。通过解析所有 Mashup 调用云 API 的记录，获得标签之间的共现次数，进而构建标签共现矩阵 **TagCoC** 为

$$\mathbf{TagCoC}=\left[\mathrm{CoC}_{p,q}\right]_{|T_m|\times|T_a|} \tag{2}$$

其中，$\mathrm{CoC}_{p,q}$ 表示标签 $t_p\in T_m$ 和 $t_q\in T_a$ 在所有 Mashup 与云 API 调用记录中的共现次数。$\mathrm{CoC}_{p,q}$ 的值越大，说明标签 t_p 对应的功能和标签 t_q 对应的功能之间的功能互补关系越强。特别地，当 $\mathrm{CoC}_{p,q}$=0 时，标签 t_p 对应的功能和标签 t_q 对应的功能之间不存在功能互补关系。

为了从图的视角集成互补标签邻居节点的信息，学习 Mashup 与云 API 之间潜在的功能互补关系及其向量表示，定义标签共现关系图 TagCoG 为一个四元组

$$\mathrm{TagCoG}=(T_m,T_a,\zeta,\mathbf{TagCoC}) \tag{3}$$

其中，ζ 是 $T_m\times T_a\rightarrow\mathbf{TagCoC}$ 的映射。若标签 $t_p\in T_m$ 和 $t_q\in T_a$ 存在共现关系，那么 $\zeta(t_p,t_q)=\mathrm{CoC}_{p,q}$。在此基础上，将标签集合 T_m 和 T_a 中的元素作为节点，ζ 作为边，$\mathrm{CoC}_{p,q}$ 作为相应边的权重，结合获取的数据，可获得 Mashup 与云 API 标签共现关系图。由此可见，TagCoG 是一张无向同质图，图中只有一种节点类型，即标签集合 T 中的标签。图的边只有一种，表示标签之间的共现次数。值得注意的是，2 个相同的标签之间也可能存在共现次数，因此该图存在自环。

3.3 基于标签共现的 Mashup 与云 API 功能互补程度刻画

通过标签共现可以刻画任意 2 个功能之间的互补关系，但是最终目标是获得 Mashup 与云 API 之间的功能互补关系。因此，引入功能互补得分（CS, complementary score）刻画 Mashup 与云 API 之间的功能互补程度。

为了计算 Mashup 与云 API 之间的功能互补得分，首先定义 Mashup m 与标签 $t_p\in T_m$ 的关系如下

$$\mathrm{MHasTag}(m,t_p)=\begin{cases}1, & m\text{含标签}t_p\\ 0, & m\text{不含标签}t_p\end{cases} \tag{4}$$

类似地，定义云 API a 和标签 $t_q\in T_a$ 的关系如下

$$\mathrm{AHasTag}(a,t_q)=\begin{cases}1, & a\text{含标签}t_q\\ 0, & a\text{不含标签}t_q\end{cases} \tag{5}$$

在式(4)和式(5)的基础上，为 Mashup m 和云 API a 遍历标签集合 T_m 和 T_a，可获得 Mashup m 和云 API a 包含标签的数量 tm 和 ta 分别为

$$\mathrm{tm}=\sum_{t_p\in T_m}\mathrm{MHasTag}(m,t_p) \tag{6}$$

$$\mathrm{ta}=\sum_{t_q\in T_a}\mathrm{AHasTag}(a,t_q) \tag{7}$$

标签共现矩阵 **TagCoC** 刻画了标签对应功能之间互补关系的大小。在实际中，由于 Mashup 和 API 通常会被打上多个标签，因此在刻画 Mashup 和 API 功能互补程度时需要考虑标签之间的全部组合关系。例如，Mashup m 有 3 个标签，云 API a 有 2 个标签，在计算两者之间的功能互补得分时，需要考虑 Mashup m 的 3 个标签与 API a 的 2 个标签之间所有的组合关系。基于上述思想，Mashup m 和云 API a 功能互补得分可以表示为

$$\mathrm{CS}_{m,a}=\frac{1}{\mathrm{tm}\times\mathrm{ma}}\mathrm{CoC}_{p,q}\cdot\sum_{t_p\in T_m}\sum_{t_q\in T_a}\mathrm{MHasTag}(m,t_p)\mathrm{AHasTag}(a,t_q) \tag{8}$$

由式(8)可知，$\mathrm{CS}_{m,a}$ 的值越大，Mashup m 和 API a 之间的功能互补程度越高，即云 API a 被推荐给对应 Mashup m 的可能性越大；反之亦然。

3.4 基于图卷积网络的 Mashup 与云 API 功能互补向量学习

3.3 节根据标签共现矩阵计算的功能互补得分可以表示 Mashup 和 API 功能之间的直接功能互补关系。然而，功能之间也存在潜在功能互补关系。本文假设如果 2 个功能之间不存在直接功能互补关系，但同时与另一个功能存在功能互补关系，则认为这 2 个功能之间存在潜在的功能互补关系。例如，在图 1 标签共现关系图中，功能“地图”和“支付”不存在共现关系，说明两者之间不存在直接功能互补关系。但是功能“地图”和“电商”共现 45 次，功能“电商”和“支付”共现 36 次，则认为功能“地图”和“支付”之间存在潜在功能互补关系。

捕捉功能之间这种潜在的功能互补关系，不仅可以提高推荐性能，同时也为解决云 API 推荐中的长尾问题提供了新的思路。

为了挖掘并表征功能间潜在的互补关系，本文在标签共现关系图 TagCoG 的基础上引入图卷积网络。图卷积网络通过当前节点与邻居节点特征向量的卷积操作，可以将邻居节点的信息聚合到当前节点上，从而实现节点的表示学习。而在 TagCoG 中，由于边的权重，即标签之间的共现次数表征互补关系的强度，在聚合邻居信息的过程中，能够更多地聚合与其互补的邻居节点信息，进而使功能标签学习到具有间接功能互补关系的其他标签信息，从而得到互补邻居信息增强的标签节点向量表示。

对于具体的云 API 和 Mashup，池化其标签特征作为功能互补向量。对于标签共现关系图 TagCoG 中的任意标签节点 t_i，可通过聚合和更新这 2 个操作实现互补信息的更新。

1) 聚合邻居信息。对于标签节点 $t_i \in T$，其互补邻居信息聚合后的表示 $\boldsymbol{\varphi}_{t_i}^{l}$ 可定义为

$$\boldsymbol{\varphi}_{t_i}^{l}=\sum_{t_j\in \mathrm{Nei}(t_i)}\frac{\zeta(t_i,t_j)}{\sqrt{\left|\mathrm{Nei}(t_i)\right|}\sqrt{\left|\mathrm{Nei}(t_j)\right|}}\boldsymbol{e}_{t_j}^{l} \tag{9}$$

其中，$\mathrm{Nei}(t_i)$ 和 $\mathrm{Nei}(t_j)$ 分别表示标签节点 t_i 和 t_j 的邻居集合，$\left|\mathrm{Nei}(t_i)\right|$ 和 $\left|\mathrm{Nei}(t_j)\right|$ 分别表示 t_i 和 t_j 邻居集合的大小；$\zeta(t_i,t_j)$ 表示 t_i 和 t_j 连边的权值，即 t_i 和 t_j 的共现次数；l 表示互补信息更新的迭代次数，$\boldsymbol{e}_{t_j}^{l}$ 表示邻居标签节点 t_j 第 l 次互补信息更新前的节点信息。特别地，$\boldsymbol{e}_{t_j}^{0}$ 表示标签节点 t_j 的初始嵌入表示，即 t_j 的独热编码向量表示，其维度等于标签集合 T 的大小 N。

2) 更新标签节点信息。利用聚合操作学习到的互补邻居信息 $\boldsymbol{\varphi}_{t_i}^{l}$，更新节点本身 $\boldsymbol{e}_{t_i}^{l}$ 的表示。集成互补邻居信息后节点 $\boldsymbol{e}_{t_i}^{l}$ 将被更新为 $\boldsymbol{e}_{t_i}^{l+1}$，表示为

$$\boldsymbol{e}_{t_i}^{l+1}=\alpha\boldsymbol{e}_{t_i}^{l}+(1-\alpha)\boldsymbol{\varphi}_{t_i}^{l} \tag{10}$$

其中，$\alpha\in[0,1]$ 用来控制当前节点信息的保留率。

经过一次互补信息更新之后，每个标签节点能够学到其一跳邻居的信息。随着互补信息更新次数的增加，每个标签节点能够学到更远距离的邻居信息。最后，对于具体的 Mashup 和云 API，如果其仅包含一个标签，那么直接把该标签对应的功能互补向量 $\boldsymbol{e}_{t_i}^{l+1}$ 作为其最终的功能互补向量。若 Mashup 和云 API 包含多个标签，则对这些标签对应的多个功能互补向量按照每维对齐的方式进行池化。具体而言，Mashup m 和云 API a 经过池化后的潜在功能互补向量 $\mathbf{CV}_m$ 和 $\mathbf{CV}_a$ 分别定义为

$$\mathbf{CV}_m=\sum_{t_p\in T_m}\mathrm{pool}(\boldsymbol{e}_{t_p}^{l+1}) \tag{11}$$

$$\mathbf{CV}_a=\sum_{t_q\in T_a}\mathrm{pool}(\boldsymbol{e}_{t_q}^{l+1}) \tag{12}$$

其中，池化函数 pool 可以采用求和、均值、最大值和最小值等函数来计算。

3.5　功能互补得分和功能互补向量嵌入的云 API 推荐

在计算得到 Mashup 与云 API 的功能互补得分和功能互补向量之后，将两者嵌入推荐模型中，使其可以捕捉到功能互补关系以提高模型推荐准确率。其中，功能互补得分表示 Mashup 和云 API 之间的功能互补程度，功能互补向量表示云 API 和 Mashup 之间的潜在功能互补关系，两者本质上都是稠密特征，可以在线性层和全连接层的输入处与其他特征进行拼接再融入模型中。令 Mashup 和云 API 的功能互补关系表示为 $C=(\mathrm{CS}_{m,a},\mathbf{CV}_m,\mathbf{CV}_a)$，由式(1)可知，功能互补关系增强的 Mashup 调用云 API 概率预测问题可进一步定义为

$$\hat{y}_{m,a}=f\left(m,a\mid F,\mathrm{CS}_{m,a},\mathbf{CV}_m,\mathbf{CV}_a\right) \tag{13}$$

融入功能互补得分和功能互补向量的推荐模型能够通过深度特征交互学习到 Mashup 和 API 的功能互补关系，实现功能互补的云 API 推荐。本文对所采用的推荐模型均采用对数损失 LogLoss 作为模型的损失函数，其定义为

$$\mathrm{LogLoss}=-\frac{1}{n}\sum_{i=1}^{n}(y_i\ln(\hat{y}_i)+(1-y_i)\ln(1-\hat{y}_i)) \tag{14}$$

其中，n 是样本的数量，$y_i\in\{0,1\}$ 是第 i 个样本的真实标签，y_i=1 表示该条样本中的 Mashup 真实调用了该条样本中的云 API，y_i=0 则表示没有调用；$\hat{y}_i\in[0,1]$ 是模型对第 i 个样本的预测结果，值越大表示 Mashup 调用云 API 的概率越大。

4　实验

为了检验提出的功能互补关系增强的云 API 推

荐方法的有效性，结合近年主流推荐模型，设计实验进行验证，以此来回答以下几个研究问题。

问题 1：利用功能互补得分和功能互补向量表征的功能互补关系是否可以提升云 API 推荐性能？

问题 2：引入功能互补关系是否可以提升长尾云 API 推荐效果？

问题 3：具有不同比例功能互补关系增强的数据对推荐模型的影响如何？

问题 4：功能标签节点的互补邻居跳数对推荐模型的影响如何？

4.1 准备工作

1) 数据集。实验数据来自云 API 聚合服务平台 Programmable Web，该平台的数据已被广泛用于云 API 推荐模型的训练和测试。本文首先通过设计的爬虫程序爬取了网页源文件，然后对网页中目标数据进行抽取和清洗，最终得到实验数据，其中包含 1 014 个 Mashup 和 746 个云 API，以及 4 573 条 Mashup 和云 API 的历史交互记录，稀疏度为 $1-\frac{4\,573}{1\,014\times 746}=99.40\%$。此外，每条数据记录包含云 API 的 12 个特征，如提供者、请求格式、验证格式等，同时还包含 Mashup 的标签特征。云 API 和 Mashup 的标签共有 352 个。在数据集中，如果云 API 和 Mashup 具有调用关系，那么预测目标值设置为 1，随机组合和无调用关系的负样本预测目标值设置为 0。正样本和负样本的比例为 1:1。实验中，数据集分为两部分，其中 80%用于训练，20%用于测试。

2) 评价指标。为了评估 Mashup 调用云 API 概率预测的准确性，采用 2 种被广泛使用的评价指标曲线下面积（AUC, area under curve） 和 F1。其中 AUC 的定义如下

$$\mathrm{AUC}=\frac{\sum_{\mathrm{ins}_i\in \mathrm{positive}}\mathrm{rank}_{\mathrm{ins}_i}-\frac{M(M-1)}{2}}{MN} \tag{15}$$

其中，$\mathrm{rank}_{\mathrm{ins}_i}$ 表示排序后第 i 条样本的序号，M 和 N 分别代表正样本和负样本的个数。AUC 越接近 1，表示预测方法的准确性越高。F1 的定义如下

$$\mathrm{F1}=\frac{2\times \mathrm{Precision}\times \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} \tag{16}$$

$$\mathrm{Precision}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{17}$$

$$\mathrm{Recall}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{18}$$

其中，TP 表示正样本中预测为正的个数，FN 表示正样本中预测为负的个数，FP 表示负样本中预测为负的个数；Precision 表示精确率，Recall 表示召回率。F1 是精确率和召回率的调和平均值，其最优值为 1，最差值为 0。

为了评估云 API 推荐结果的效果，选择命中率（HR, hit ratio）作为评价指标。首先为测试集中的 Mashup 预测其对数据集中全部云 API 的调用概率，并按照预测值从大到小排序，生成 Top-K 的排名列表，其中 K 表示列表的长度。然后针对测试集中的每条正样本，即样本中的 Mashup 真实调用了其中的云 API，利用 Hits@K 计算该云 API 是否在该 Mashup 的 Top-K 列表中：若存在，则为 1；否则为 0。最后计算测试集中的所有正样本的 Hits@K 并取平均值，作为最终的命中率。计算方法为

$$\mathrm{HR}@K=\frac{\mathrm{NumberOfHits}@K}{N} \tag{19}$$

其中，分子表示测试正样本的成功命中的个数，N 表示测试集中正样本的个数。

3) 实验环境。本文实验环境如下：操作系统为 Windows10，内存为 8 GB DDR4，CPU 为 Intel(R) Core(TM) i7-9700K@3.60 GHz；GPU 为 NVIDIA GeForce GTX1060 6 GB；编程语言为 Python。

4.2 功能互补得分和功能互补向量的有效性

为了验证表征功能互补关系的功能互补得分和功能互补向量在云 API 推荐中的有效性，本文将其应用于当前主流的 11 种云 API 推荐模型，其中包括以下几类模型。1) 线性模型：LR[20]。2) 高阶交互模型：FM[21]、IFM[22]、DIFM[23]、NFM[24]、AFM[25]。3) 深度集成模型：WDL[26]、DCN[27]、DeepFM[28]、xDeepFM[29]、AutoInt[30]。每个模型均使用 Adam 优化器，学习率为 0.001，嵌入向量的维度为 100，训练周期为 10，节点信息保留率 α 为 0.5。

稀疏场景下功能互补得分和功能互补向量的有效性分析如表 2 所示。其中，Base 表示模型仅学习 12 种原始特征；+CS 表示模型在 Base 的基础上增加了功能互补得分 CS；+CV 表示模型在 Base 的基础上增加功能互补向量 CV；+CSCV 表示模型在 Base 的基础上增加功能互补得分 CS 和功能互补向量 CV。从表 2 可以得出以下结论。

1) 分别对比 Base 列与+CS 列和+CV 列的实验结果可以看出，在分别加入了功能互补得分、功能

表 2　　**稀疏场景下功能互补得分和功能互补向量的有效性分析**

模型	AUC				F1				HR@5			
	Base	+CS	+CV	+CSCV	Base	+CS	+CV	+CSCV	Base	+CS	+CV	+CSCV
LR	0.892	0.897	0.899	0.905	0.774	0.777	0.777	0.785	0.379	0.392	0.393	0.408
FM	0.893	0.895	0.902	0.907	0.778	0.785	0.781	0.789	0.393	0.407	0.408	0.425
IFM	0.895	0.897	0.904	0.911	0.787	0.799	0.797	0.799	0.404	0.419	0.420	0.438
DIFM	0.897	0.899	0.904	0.912	0.795	0.804	0.802	0.807	0.408	0.424	0.424	0.443
NFM	0.898	0.901	0.905	0.913	0.799	0.808	0.807	0.812	0.426	0.445	0.445	0.464
AFM	0.899	0.903	0.905	0.924	0.807	0.817	0.816	0.820	0.437	0.458	0.458	0.478
WDL	0.894	0.901	0.901	0.918	0.809	0.823	0.820	0.825	0.444	0.467	0.468	0.486
DCN	0.896	0.901	0.902	0.918	0.820	0.834	0.825	0.837	0.449	0.473	0.473	0.492
DeepFM	0.898	0.905	0.905	0.925	0.826	0.838	0.837	0.843	0.457	0.482	0.483	0.502
xDeepFM	0.899	0.906	0.906	0.928	0.830	0.844	0.843	0.852	0.472	0.500	0.501	0.520
AutoInt	0.902	0.907	0.907	0.931	0.840	0.850	0.850	0.862	0.487	0.519	0.520	0.538
平均值	0.897	0.901	0.904	0.918	0.806	0.816	0.814	0.821	0.432	0.453	0.454	0.472

互补向量之后，模型的 AUC 分别平均提升了 0.50%、0.78%，模型的 F1 分别提升了 1.28%、1.01%，模型的 HR@5 分别提升了 4.76%、4.91%。这说明功能互补得分刻画出了云 API 和 Mashup 之间的功能互补程度，能够提升模型的效果；功能互补向量通过挖掘云 API 和 Mashup 之间的潜在功能互补关系，对模型也有提升效果。

2) 对比+CS 列和+CV 列的实验结果可以看出，在 F1 指标上，加入 CS 比加入 CV 提升更高，而在 AUC 指标上，加入 CV 比加入 CS 提升更高。这是因为显式的功能互补得分可以通过数值高低更好地区分正负样本，而隐式的功能互补向量挖掘出了潜在的功能互补关系，使模型的预测结果更接近真实值。

3) 功能互补得分 CS 和功能互补向量 CV 同时加入模型中训练的提升效果大于单独加入模型中训练的提升效果。这说明功能互补关系和潜在功能互补关系能够相互促进，产生更好的推荐效果。

4) 对比线性模型、高阶交互模型、深度集成模型可以看出，在加入功能互补得分和功能互补向量之后，模型的 AUC 平均提升了 1.46%、1.90%、2.92%，模型的 F1 分别平均提升了 1.42%、1.54%、2.28%，模型的 HR@5 分别提升了 7.65%、8.69%、9.90%。不难看出随着模型变得复杂，功能互补得分和功能互补向量对模型的提升效果逐渐变得明显。这是因为相比于线性模型，高阶交互模型能够自适应地学习功能互补得分、功能互补向量和其他特征之间的高阶交互关系，从而提升模型的效果。同时相比于高阶交互模型，深度集成模型通常具有更加复杂的结构，能够更好地捕捉功能互补得分、功能互补向量和其他特征之间的复杂关系和序列信息，提高模型的效果。

4.3　长尾云 API 的推荐性能分析

图 2 给出了在实验数据中云 API 被 Mashup 调用次数的分布情况。

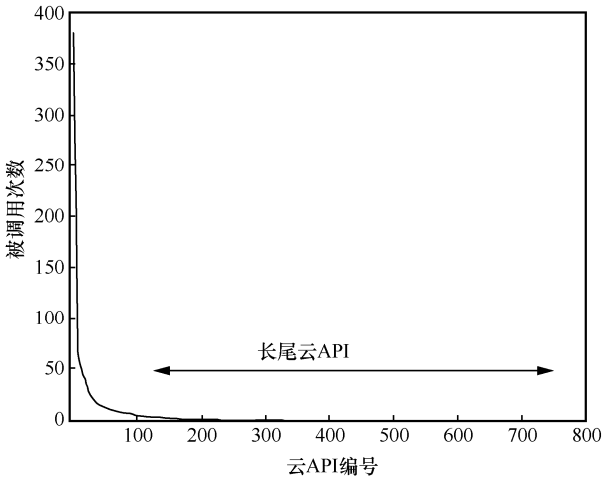

图 2　云 API 被 Mashup 调用次数的分布情况

本文将被调用次数小于或等于 4 的云 API 视为长尾云 API，其中长尾云 API 占据了云 API 总数的 81.6%，同时占据了云 API 调用次数的 19.9%。为了验证功能互补得分和功能互补向量对长尾云 API

表 3　　**功能互补得分和功能互补向量对长尾云 API 的推荐效果**

模型	AUC				F1				HR@5			
	Base	+CS	+CV	+CSCV	Base	+CS	+CV	+CSCV	Base	+CS	+CV	+CSCV
LR	0.616	0.666	0.660	0.707	0.685	0.724	0.718	0.774	0.278	0.307	0.308	0.327
FM	0.619	0.667	0.663	0.709	0.688	0.726	0.721	0.791	0.288	0.319	0.319	0.339
IFM	0.622	0.669	0.663	0.710	0.694	0.728	0.727	0.795	0.296	0.329	0.330	0.352
DIFM	0.628	0.670	0.665	0.713	0.700	0.736	0.735	0.797	0.313	0.350	0.350	0.372
NFM	0.629	0.673	0.669	0.715	0.714	0.741	0.739	0.799	0.320	0.357	0.357	0.381
AFM	0.632	0.674	0.671	0.717	0.716	0.743	0.744	0.807	0.325	0.365	0.364	0.389
WDL	0.630	0.671	0.665	0.714	0.722	0.758	0.759	0.809	0.323	0.364	0.364	0.389
DCN	0.632	0.673	0.669	0.716	0.724	0.764	0.763	0.820	0.340	0.384	0.384	0.410
DeepFM	0.635	0.674	0.672	0.719	0.726	0.769	0.778	0.825	0.353	0.398	0.399	0.427
xDeepFM	0.639	0.679	0.675	0.723	0.728	0.774	0.785	0.837	0.354	0.401	0.402	0.431
AutoInt	0.642	0.681	0.680	0.728	0.732	0.789	0.787	0.840	0.369	0.419	0.419	0.450
平均值	0.629	0.672	0.669	0.716	0.712	0.750	0.751	0.809	0.324	0.363	0.363	0.388

推荐的有效性，本节针对长尾云 API 进行实验。本节的实验设置与 4.2 节类似，不同之处在于在模型的预测阶段，仅统计长尾云 API 的预测结果。

功能互补得分和功能互补向量对长尾云 API 的推荐效果如表 3 所示。从表 3 可以观察到，在长尾场景下，现有 11 种推荐模型的推荐性能较差。以 AUC 为例，11 种推荐模型在未加入功能互补关系时的平均 AUC 仅为 0.629，这说明现有推荐模型未能有效解决长尾问题。在加入功能互补得分和功能互补向量之后，11 种模型的 AUC 均有提升，并且平均提升率约为 13.7%，对比表 2 稀疏场景下 AUC 的平均提升率 2.3%，提升效果十分明显。这是因为功能互补得分可有效刻画 Mashup 和云 API 之间的潜在调用关系，功能互补向量可从功能互补的角度丰富对 Mashup 和云 API 的表示，进而缓解了长尾云 API 推荐准确率低的问题。

4.4　不同比例互补关系增强数据的影响

为了直观地观察使用功能互补关系对云 API 推荐模型的推荐准确率带来的提升，在模型训练阶段调整受功能互补关系增强的数据比例，随机选择数据集中的一部分输入记录，加入功能互补得分和功能互补向量，作为受互补关系增强的记录。例如，10%表示训练集中有 10%的训练数据加入了功能互补关系，即功能互补得分和功能互补向量，而其余 90%的数据不引入功能互补关系。在本节实验过程中，设置受功能互补关系增强数据比例从 0 增加到 100%，步长为 10%，来研究不同比例互补关系增强数据对预测精度的影响。从图 3～图 5 的实验结果可以得出以下结论。

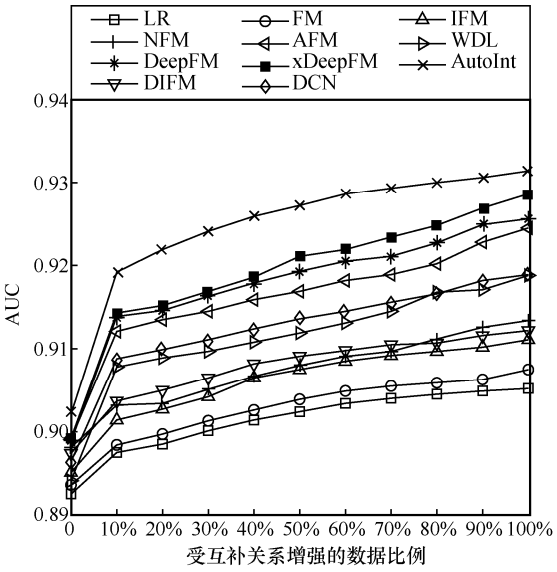

图 3　受功能互补关系增强的数据比例与 AUC 的关系

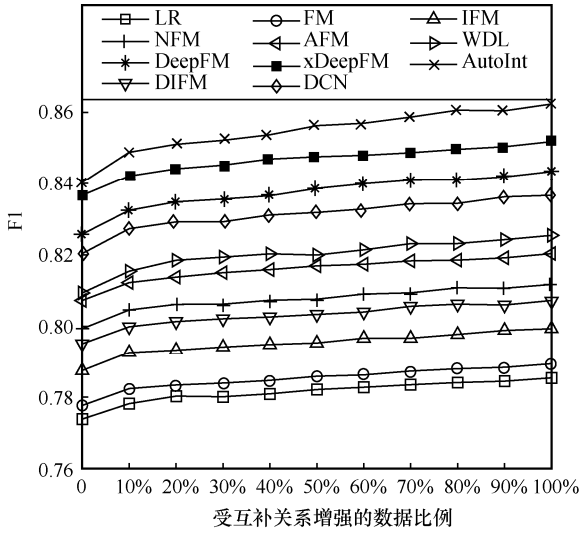

图 4　受功能互补关系增强的数据比例与 F1 的关系

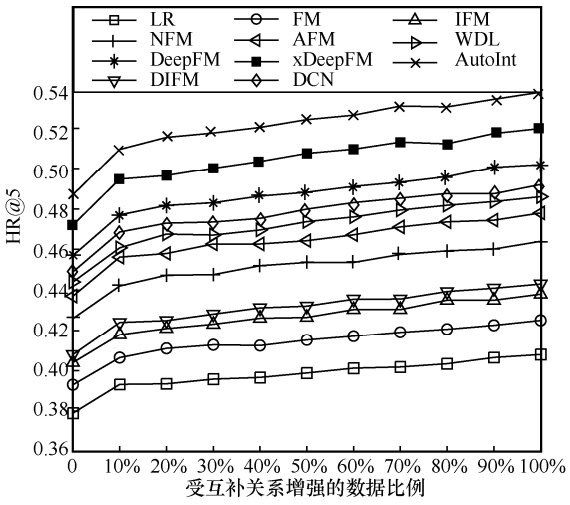

图 5　受功能互补关系增强的数据比例与 HR@5 的关系

1) 在受互补关系增强的数据比例由 0 到 10%的过程中，AUC、F1 和 HR@5 的提升效果最为显著。这说明功能互补的引入对模型有提升效果。

2) 当受功能互补关系增强的数据比例不断增加时，所有推荐模型的 AUC、F1、HR@5 指标越来越高，并且在数据比例由 10%到 100%的过程中，3 个指标呈现逐渐提升的趋势变化，性能没有出现较大波动。由此可以推断，受功能互补关系增强的训练数据越多，云 API 推荐模型的效果越好。这也进一步说明了基于标签共现计算得到的功能互补得分和学习到的功能互补向量所表征的功能互补关系的有效性。

4.5　互补标签邻居跳数对推荐模型的影响

在研究本节问题之前，需要建立标签共现关系图 TagCoG。首先分析测试集中标签之间的共现频率，建立标签共现矩阵 **TagCoC**，标签集合的大小为 352，因此矩阵的规模为 352×352。然后利用式 (3) 建立标签共现关系图 TagCoG，其中，图的节点数量为 352 个，边的数量为 14 749 条。

在功能互补向量学习过程中，互补信息更新次数的增加会使标签学到更远距离的互补标签邻居的信息，每更新一次，能学到的互补标签邻居的跳数增加一跳，进一步生成不同的功能互补向量。换句话说，互补信息更新次数的增加会使功能互补向量能够表示更远距离的潜在功能互补关系。为了确定最佳互补更新次数，生成最优的功能互补向量，本节针对聚合互补标签邻居跳数进行对比实验。图 6～图 8 分别给出了聚合互补邻居跳数从 0 增加到 5 时 AUC、F1、HR@5 的变化情况。

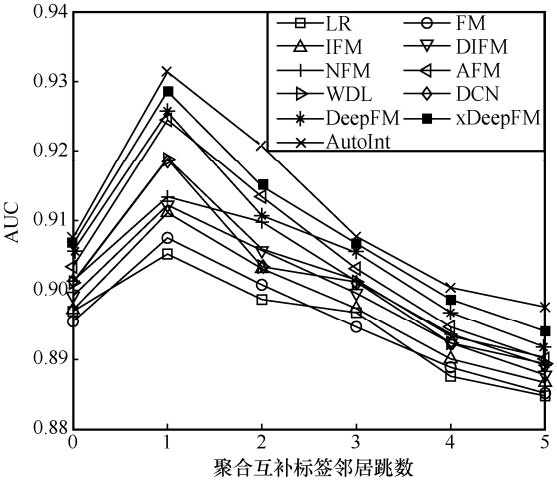

图 6　聚合互补标签邻居跳数与 AUC 的关系

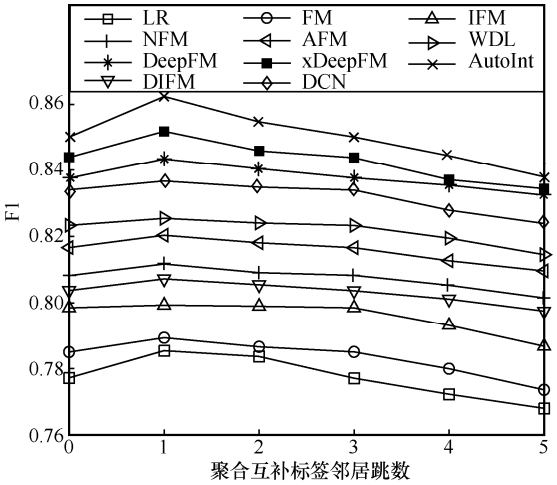

图 7　聚合互补标签邻居跳数与 F1 的关系

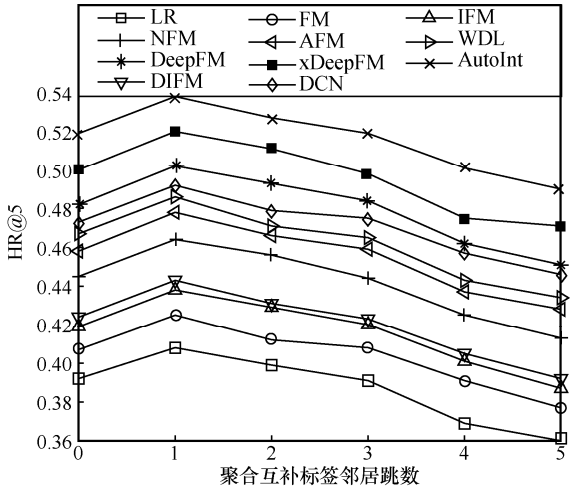

图 8　聚合互补标签邻居跳数与 HR@5 的关系

当聚合互补标签邻居跳数从 0 增加到 1 时，绝大部分模型的 AUC、F1、HR@5 均有显著的改进，这说明集成互补邻居信息学习到的功能互补向量

可以有效提高推荐模型的性能。特别地，大部分模型在跳数为 1 时，即功能互补向量表示的潜在功能互补关系距离为 1 时，模型的效果最好。随着跳数增加，效果逐渐变差。这说明虽然随着跳数的增加，功能互补向量能够表示更远距离的潜在功能互补关系，但同时标签节点原本的信息被互补邻居的信息覆盖更新，弱化了标签原本的表示功能特征，使学习到这些标签特征的模型效果变差。由此得出互补标签邻居的跳数最佳值为 1 跳。

5 结束语

针对云 API 推荐中对功能互补关系重视程度不足问题，本文提出一种功能互补关系增强的云 API 推荐方法。首先，提出基于标签共现表示功能之间的功能互补关系。其次，利用功能互补得分表示云 API 和 Mashup 之间的功能互补程度，利用功能互补向量表示云 API 和 Mashup 之间的潜在功能互补关系。最后，将功能互补得分和功能互补向量嵌入云 API 推荐模型中，使模型能够学习到云 API 和 Mashup 之间的功能互补关系并在推荐中起到关键作用。实验结果表明，所提出的基于功能互补关系数据增强的方法在处理云 API 推荐模型所面临的数据稀疏和长尾问题时具有一定的优势。

在未来的工作中，将在模型层面研究基于协同过滤与对比学习技术的功能互补云 API 推荐模型。一方面，基于云 API 与 Mashup 的 ID 特征构建历史交互矩阵进行协同过滤，继而可充分且高效地利用云 API 与 Mashup 丰富的历史交互信息；另一方面，对比学习可以同时利用云 API 生态知识图谱中丰富的语义信息，与云 API 互补关系图中的功能互补信息，生成统一的图向量表示，进一步丰富推荐模型的特征表示。此外，拟研究如何将本文提出的功能互补关系刻画方法拓展到云 API 与云 API 之间，进而向已知云 API 推荐互补云 API，即开发者在选用某一云 API 时，如何通过推荐算法得到与该云 API 功能互补的云 API 候选列表。

参考文献：

[1] PAPAZOGLOU M P, HEUVEL W J V D. Service oriented architectures: approaches, technologies and research issues[J]. The VLDB Journal, 2007, 16(3): 389-415.

[2] YU J, BENATALLAH B, CASATI F, et al. Understanding mashup development[J]. IEEE Internet Computing, 2008, 12(5): 44-52.

[3] 陈真, 乞文超, 贺鹏飞, 等. 云应用程序编程接口安全研究综述: 威胁与防护[J]. 电子与信息学报, 2023, 45(1): 371-382.
CHEN Z, QI W C, HE P F, et al. A survey for cloud application programming interface security: threats and protection[J]. Journal of Electronics & Information Technology, 2023, 45(1): 371-382.

[4] WU H, DUAN Y H, YUE K, et al. Mashup-oriented Web API recommendation via multi-model fusion and multi-task learning[J]. IEEE Transactions on Services Computing, 2022, 15(6): 3330-3343.

[5] CAO B, PENG M, QING Y, et al. Web API recommendation via combining graph attention representation and deep factorization machines quality prediction[J]. Concurrency and Computation: Practice and Experience, 2022, 34(21): E7069.

[6] ZHANG M Y, BOCKSTEDT J. Complements and substitutes in online product recommendations: the differential effects on consumers' willingness to pay[J]. Information & Management, 2020: doi.org/10.1016/j.im.2020.103341.

[7] WANG R, WANG J, SU Z. Learning compatibility knowledge for outfit recommendation with complementary clothing matching[J]. Computer Communications, 2022, 181: 320-328.

[8] ZHANG W, CHEN Z Y, ZHA H Y, et al. Learning from substitutable and complementary relations for graph-based sequential product recommendation[J]. ACM Transactions on Information Systems, 2022, 40(2): 1-28.

[9] WANG Z H, JIANG Z H, REN Z C, et al. A path-constrained framework for discriminating substitutable and complementary products in E-commerce[C]//Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. New York: ACM Press, 2018: 619-627.

[10] MCAULEY J, PANDEY R, LESKOVEC J. Inferring networks of substitutable and complementary products[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2015: 785-794.

[11] HUYNH C P, CIPTADI A, TYAGI A, et al. CRAFT: complementary recommendation by adversarial feature transform[C]//European Conference on Computer Vision. Berlin: Springer, 2019: 54-66.

[12] BOTANGEN K A, YU J, SHENG Q Z, et al. Geographic-aware collaborative filtering for Web service recommendation[J]. Expert Systems With Applications, 2020: doi.org/10.1016/j.eswa.2020.113347.

[13] XIONG R, WANG J, ZHANG N, et al. Deep hybrid collaborative filtering for Web service recommendation[J]. Expert Systems with Applications, 2018, 110: 191-205.

[14] YAO L N, WANG X Z, SHENG Q Z, et al. Mashup recommendation by regularizing matrix factorization with API co-invocations[J]. IEEE Transactions on Services Computing, 2021, 14(2): 502-515.

[15] TIAN G, WANG J, HE K Q, et al. Integrating implicit feedbacks for time-aware Web service recommendations[J]. Information Systems Frontiers, 2017, 19(1): 75-89.

[16] NIU H, KEIVANLOO I, ZOU Y. API usage pattern recommendation for software development[J]. Journal of Systems and Software, 2017, 129: 127-139.

[17] ALMARIMI N, OUNI A, BOUKTIF S, et al. Web service API recommendation for automated mashup creation using multi-objective

evolutionary search[J]. Applied Soft Computing, 2019, 85: 105830.

[18] NAÏM H, AZNAG M, DURAND N, et al. Semantic pattern mining based Web service recommendation[C]//Proceedings of International Conference on Service-Oriented Computing. Berlin: Springer, 2016: 417-432.

[19] DOJCHINOVSKI M, KUCHAR J, VITVAR T, et al. Personalised graph-based selection of Web APIs[C]// Proceedings of International Semantic Web Conference. Berlin: Springer, 2012: 34-48.

[20] RICHARDSON M, DOMINOWSKA E, RAGNO R. Predicting clicks: estimating the click-through rate for new ads[C]//Proceedings of the 16th International Conference on World Wide Web. New York: ACM Press, 2007: 521-530.

[21] RENDLE S. Factorization machines[C]//Proceedings of IEEE International Conference on Data Mining. Piscataway: IEEE Press, 2011: 995-1000.

[22] YU Y T, WANG Z, YUAN B. An input-aware factorization machine for sparse prediction[C]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. California: International Joint Conferences on Artificial Intelligence Organization, 2019: 1466-1472.

[23] LU W T, YU Y T, CHANG Y Z, et al. A dual input-aware factorization machine for CTR prediction[C]//Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. California: International Joint Conferences on Artificial Intelligence Organization, 2020: 3139-3145.

[24] HE X N, CHUA T S. Neural factorization machines for sparse predictive analytics[C]//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2017: 355-364.

[25] XIAO J, YE H, HE X, et al. Attentional factorization machines: learning the weight of feature interactions via attention networks[J]. arXiv Preprint, arXiv: 1708.04617, 2017.

[26] CHENG H T, KOC L, HARMSEN J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. New York: ACM Press, 2016: 7-10.

[27] WANG R X, FU B, FU G, et al. Deep & cross network for ad click predictions[C]//Proceedings of the ADKDD'17. New York: ACM Press, 2017: 1-7.

[28] GUO H F, TANG R M, YE Y M, et al. DeepFM: a factorization-machine based neural network for CTR prediction[C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence. New York: ACM Press, 2017: 1725-1731.

[29] LIAN J X, ZHOU X H, ZHANG F Z, et al. xDeepFM: combiningexplicit and implicit feature interactions for recommender systems[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM Press, 2018: 1754-1763.

[30] SONG W, SHI C, XIAO Z, et al. Autoint: automatic feature interaction learning via self-attentive neural networks[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. New York: ACM Press, 2019: 1161-1170.

[作者简介]

陈真（1987− ），男，陕西宝鸡人，博士，燕山大学副教授、博士生导师，主要研究方向为服务计算、推荐系统和服务化软件开发等。

陈文辉（1997− ），男，山东济宁人，燕山大学硕士生，主要研究方向为服务计算、云 API 推荐等。

刘啸威（2000− ），男，山东枣庄人，燕山大学硕士生，主要研究方向为服务计算、数据挖掘、云 API 推荐等。

尤殿龙（1981– ），男，内蒙古赤峰人，博士，燕山大学副教授，主要研究方向为数据挖掘、特征选择和推荐系统等。

刘林林（1990– ），男，山东泰安人，中国科学院文献情报中心馆员，主要研究方向为科技信息处理、知识挖掘和推荐系统等。

申利民（1962– ），男，黑龙江佳木斯人，博士，燕山大学教授，主要研究方向为协同计算、服务计算和信息安全等。