基于概率逻辑推理的高阶互补云API推荐方法

陈 真[1),2)] 谢登辉[1)] 王小龙[1)] 孙梦梦[1)]
刘啸威[1)] 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)](河北省计算机虚拟制造与系统集成重点实验室 河北 秦皇岛 066004)

摘 要 云时代,云API作为服务交付、数据交换和能力复制的最佳载体,已成长为当今面向服务软件开发和企业数字化转型不可或缺的核心要素.然而动态开放网络中持续增长的云API在给开发者提供了更多选择的同时,也将其淹没在海量的云API选择之中,设计有效的云API推荐方法就此成为API经济健康发展中迫切要解决的现实问题.但是,现有研究主要利用搜索关键词、服务质量和调用偏好进行建模,生成质量高功能单一的云API推荐列表,没有考虑服务化软件实际开发中开发者对多元化高阶互补云API的客观需要.高阶互补云API推荐旨在为多个查询云API生成多元互补云API列表,要求推荐结果与查询云API均互补,以满足开发者的联合需求.针对此问题,本文提出基于概率逻辑推理的高阶互补云API推荐方法(Probabilistic Logic Reasoning for High-order Complementary Cloud API Recommendation,PLR4HCCR).首先,通过云API生态真实数据分析论证云API互补推荐需求的必要性和互补关系建模中替补噪声的客观存在,为云API互补推荐问题研究提供动机和数据支持.其次,采用Beta概率嵌入对云API及其之间的关系约束进行编码,以刻画云API间互补关系的不确定性和支持互补逻辑推理.接着,设计由投影、取反和交并三个基本逻辑算子构建的互补关系逻辑推理网络,使查询集中的每个云API获得非对称互补关系感知和替补噪声消解约束下的互补云API表示.然后,引入注意力机制为查询云API的互补云API分配不同权重,增强高阶互补云API基向量的表征能力.在此基础上,采用KL散度度量高阶互补云API基向量与候选云API之间的距离,并根据KL散度排序生成高阶互补性可感知下的云API推荐结果.最后,我们利用两个真实云API数据集在不同阶互补推荐场景下进行实验,实验表明,与传统启发式推荐方法和深度学习推荐方法相比,PLR4HCCR在互补关系感知推理和替补噪声消解方面均具有较大的优势,继而使其在低阶、高阶和混合阶互补云API推荐中均展示出更优的推荐效果和更强的泛化能力.进一步,超参数敏感性实验、实例分析和用户调查验证了方法的有效性、实用性和可行性,这使结合高阶互补关系的云API推荐方法PLR4HCCR不仅更有可能生成开发者满意的结果,而且可有效提升云API服务提供者的收益.

关键词 面向服务软件开发;云API推荐;高阶互补;逻辑推理;Beta概率嵌入
中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2024.01922

Probabilistic Logic Reasoning for High-order Complementary Cloud API Recommendation

CHEN Zhen[1),2)] XIE Deng-Hui[1)] WANG Xiao-Long[1)] SUN Meng-Meng[1)]
LIU Xiao-Wei[1)] SHEN Li-Min[1),2)]

[1)](*School of Information Science and Engineering*, *Yanshan University*, *Qinhuangdao*, *Hebei* 066004)
[2)](*The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province*, *Yanshan University*, *Qinhuangdao*, *Hebei* 066004)

Abstract In the cloud era, cloud API, as the best carrier for service delivery, data exchange, and

收稿日期:2023-08-09;在线发布日期:2024-05-14.本课题得到国家自然科学基金(No.62102348,No.62276226)、河北省自然科学基金(No.F2022203012)、中央引导地方科技发展资金项目(236Z0103G)、河北省教育厅高等学校科技计划项目(No.QN2020183)和河北省创新能力提升计划项目(No.22567626H)资助.**陈 真**,(通信作者),博士,副教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为服务计算、推荐系统和软件工程.E-mail:zhenchen@ysu.edu.cn.**谢登辉**,硕士研究生,主要研究方向为云API推荐和互补推荐系统.**王小龙**,硕士研究生,主要研究方向为语义分析与云API互补推荐.**孙梦梦**,博士研究生,主要研究领域为深度学习与云API互补推荐.**刘啸威**,硕士研究生,主要研究方向为云API数据挖掘与互补推荐系统.**申利民**,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为服务计算、柔性软件、协同计算和信息安全.

capability replication, has grown into an indispensable core element in today's service-oriented software development and enterprise digital transformation. However, while the continued growth of cloud APIs in dynamic open networks provides developers with more options, it also inundates them with a sea of cloud API selection, so designing effective cloud API recommendation methods has become an urgent practical task to be resolved in the healthy development of the cloud API economy. But current existing researches focus on modeling and generating high-quality, single-function cloud API recommendation list using search keyword, service of quality, and call preference, without considering the objective concerns of developers for diversified and high-order complementary cloud APIs in actual service-oriented software development. The purpose of high-order complementary cloud API recommendation is to generate a diversified list of complementary cloud APIs for multiple query cloud APIs, requiring the recommendation results to be complementary to the query cloud APIs to meet the joint interests of developers. To solve this problem, we propose a probabilistic logic reasoning based high-order complementary cloud API recommendation (PLR4HCCR) approach. Firstly, the necessity of cloud API complementary recommendation requirements and the objective existence of substitution noise in complementary relationship modeling were demonstrated through the analysis of real cloud API ecological data, providing motivation and data support for the research on cloud API complementary recommendation problem. Secondly, Beta probability embedding is used to encode cloud APIs and their relationship constraints, so as to characterize the uncertainty representation and support logical reasoning of complementary relationships between cloud APIs. Thirdly, a complementary relationship logical reasoning network is designed, consisting of three basic logical operators: projection, negation, and intersection, so that each cloud API in the query set can obtain the complementary cloud API representation under the constraints of asymmetric complementary relationship perception and substitution noise resolution. Then, an attention mechanism is introduced to assign different weights to the complementary cloud APIs for querying cloud API, enhancing the representation ability of the higher-order complementary cloud API base vector. On this basis, KL divergence is used to measure the distance between the higher-order complementary cloud API base vector and the candidate cloud APIs, and the complementary cloud APIs are generated based on KL divergence ranking under the higher-order complementarity perception. Finally, we conducted a series of experiments using two real cloud API datasets in different order complementary recommendation scenarios. The experiments show that, PLR4HCCR has significant advantages in complementary relationship perception reasoning and substitution noise resolution compared to traditional heuristic recommendation methods and deep learning recommendation methods, which demonstrates better recommendation performance and stronger generalization ability under low-order, high-order, and mixed-order complementary cloud API recommendation scenarios. Furthermore, the hyperparameter sensitivity experiments, case analysis and user study have verified the effectiveness, practicality and feasibility of PLR4HCCR, which makes the cloud API recommendation combined with high-order complementarity not only more likely to generate satisfactory results for developers, but also effectively improve the revenue of cloud API service providers.

Keywords service-oriented software development; cloud API recommendation; high-order complementarity; logical reasoning; Beta probabilistic embedding

1 引 言

过去五十年软件工程的发展历史已经证明，复用技术是解决软件危机，提高软件生产效率和质量，推进软件产业走上工程化和工业化的现实可行途径[1]. 为了适应动态开放多变网络环境下的软件复用，人们提出了面向服务的软件开发方法[2]，采用中立的访问协议和显式的服务契约实现服务使用者和提供者解耦，深刻改变了软件的形态、开发范式和运行模式. 随着面向服务软件开发理念的深入发展，云应用程序编程接口（Application Programming Interface，API）凭借松耦合、跨平台和轻量级等优势[3]，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. 这样开发人员便可通过调用云API来复用云端的各类服务，快速实现新技术和大数据的接入，响应用户多变的需求，有效降低软件的复杂性和开发成本.

进入万物互联的云时代，云API作为服务交付、数据交换和能力复制的最佳载体，可有效落地高新技术企业AI算法和释放传统行业数据价值，其已成长为当今数字经济建设不可或缺的基础设施[4]. 一方面，不论是科大讯飞通过云API提供语言翻译服务，阿里巴巴使用云API支持在线支付服务，还是OpenAI采用云API将ChatGPT聊天机器人等智能服务带入我们的生活，云API正在成为实现人工智能技术下沉，助力企业和组织实现数字化转型的重要管道. 另一方面，不论是中国铁路12306通过调用云API在其APP中集成天气、订餐和约车等功能，还是众多开发者利用云API提供疫情和防疫知识来共享传染病、健康和防护等信息，云API正在成为释放传统行业数据价值，实现跨界融合、价值重塑和数字经济发展的重要依托.

随着数字化时代的到来，云API已成为了不同软件应用、平台、设备或系统之间互操作、共享数据和功能的关键桥梁. API经济作为数字经济的重要组成部分，是一种基于开放、共享和协同的新型经济模式. 它打破了传统产业的界限，促进了不同领域之间的融合和协作，为推动整个数字生态系统的繁荣和数字经济的发展注入了新的活力和动力. 当前，越来越多企业和组织开始拥抱API经济，纷纷加入将其核心业务、AI算法和沉睡数据服务化的行列，由此极大地丰富了网络中的云API，并且云API的数量正在以每年30%以上的增速走入人们的视野[5]. 例如，开放云API平台ProgrammableWeb①中的云API数量在近十年时间里已经从3K+增加到20K+. 然而，网络中不断丰富的云API在给开发者提供了更多便利的同时，也将其淹没在海量的云API选择之中，由此阻碍了云API的普及应用以及其进一步的发展[6]. 因此，如何准确高效地为开发者推荐满足其个性化需求的云API已成为服务化软件开发中亟需解决的现实问题.

针对动态开放网络环境中持续增长的海量云API挑战和开发者对云API推荐的客观需要，研究者们从不同视角提出了多种云API推荐方法，其中已有典型的方法包括内容驱动的云API推荐[7-10]、服务质量感知的云API推荐[11-14]和基于偏好的云API推荐[15-18]. 内容驱动的云API推荐方法主要利用搜索关键词匹配或文本语义理解生成功能单一的云API推荐列表，不能有效满足开发者的服务质量需要和个性化偏好. 服务质量感知的云API推荐方法聚焦云API非功能侧需求，旨在为用户推荐高质量云API. 但是由于真实世界中云API的服务质量信息通常不完整且难以获得，继而使服务质量感知的云API推荐方法并不可靠，甚至在许多情况下不适用. 基于偏好的云API推荐方法，着眼于开发者和云API之间的历史交互关系，尝试通过预测开发者对云API的偏好以期实现个性化云API推荐. 综上可见，现有云API推荐研究主要围绕将搜索关键词、服务质量和开发者偏好等约束条件作为滤子生成质量高、功能单一的云API推荐列表，尚未注意到开发者在进行面向云API服务混搭的应用开发时，对多元化互补云API推荐的实际需要.

互补云API推荐旨在向开发者提供经常一起调用的云API，以满足开发者的联合需求. 例如，开发者在调用具有地图功能的“Google Map”云API时，常常会混搭具有天气功能的“Weather Channel”云API，以实现在地图上显示天气信息. 我们将在2.1节通过ProgrammableWeb和华为应用市场两个平台中的云API数据分析，论证多元互补云API推荐需求的客观性和必要性. 在本文中，我们提出面向服务化软件开发场景的两类云API互补推荐问题，包括低阶互补推荐和高阶互补推荐，如图1所示.

高阶互补云API推荐旨在为多个查询云API生成多元互补云API列表，要求推荐的云API与多个

① http://www.programmableweb.com/

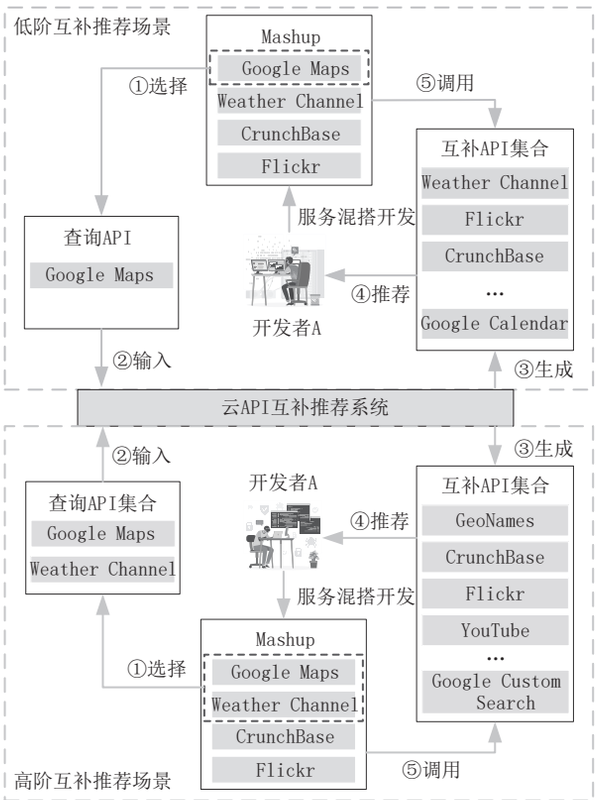

图1　面向服务化软件开发的云API互补推荐示意图

查询云API均互补，以满足应用迭代开发过程中对互补云API选择调用的实际需要．在如图1所示的高阶互补云API推荐场景中，云API互补推荐系统会将提供地理数据的“Geonames”、实现信息检索服务的“CrunchBase”等云API作为与开发者已调用的两个云API“Google Map”和“Weather Channel”均互补的云API推荐给开发者．这是因为“Google Maps”和“Weather Channel”二者常常混搭“Geonames”以实现旅游指南功能，组合“CrunchBase”实现基于地图的信息检索功能．特别地，当查询云API数量为1时，高阶互补云API推荐问题则简化为低阶(1阶)互补云API推荐问题．由此可见，结合高阶互补关系的云API推荐系统可在提高推荐结果多样性的同时，进一步提升面向云API服务混搭应用的用户体验．

综上，面向服务化软件开发的云API互补推荐系统应用可以实现云API使用者和提供者的双赢．从云API使用者来看，当用户检索云API时，不仅会关注与检索内容相匹配的云API，而且也会对潜在互补的云API感兴趣[19]．互补云API推荐则可有效满足服务化软件开发过程中开发者对多元化云API的实际需要，继而提升服务化软件开发效率和用户对所开发出来的应用软件的用户体验．从云API提供者来看，结合互补关系的云API推荐可以帮助云API提供者更好的制定定价策略实现组合销售，提升云API的收益．

然而，对面向服务软件开发中云API的互补关系建模，实现互补云API推荐面临如下挑战：1)如何刻画非对称的互补关系．例如，“地图API”是“疫情API”的互补云API．反之，“疫情API”是“地图API”互补云API的可能性很小，因为开发者不会因为先调用了地图API再使用疫情API．2)如何减少功能相同的替补关系噪声对互补推荐建模的影响．通常，云API之间的共同调用行为能够有效表征其互补关系，但是其中的相似功能等替补关系则会对互补关系的发掘带来噪声，继而影响互补推荐结果．3)如何感知高阶互补推荐需求．高阶互补要求推荐的云API与查询云API集合整体互补，即互补推荐具有高阶性．

针对上述挑战，本文提出基于概率逻辑推理的高阶互补云API推荐方法(Probabilistic Logic Reasoning for High-order Complementary Cloud API Recommendation，PLR4HCCR)．在PLR4HCCR中，我们首先采用Beta概率嵌入编码云API及其之间的关系约束，以刻画云API间互补关系的不确定性和支持进一步的互补关系逻辑推理．其次，引入一阶逻辑推理思想，定义投影、取反和交并三个基本逻辑算子，据此构建互补关系逻辑推理网络，使查询云API集合中的每一个云API能够获得非对称互补关系感知和替补关系消解约束下的互补云API表示．如果采用深度学习方法利用特征交互学习进行互补关系建模，则无法有效消除替补噪声的影响．为此，在互补逻辑推理网络中，我们设计并引入取反算子来消解替补噪声的影响．再次，引入注意力机制为查询云API的互补云API分配不同权重，增强高阶互补云API基向量的表征能力．最后，由于KL(Kullback Leibler)散度可有效衡量两个概率分布之间的差异，利用KL散度来计算高阶互补云API基向量(Beta概率分布表示)与候选云API(Beta概率分布表示)之间的距离，进一步以KL散度距离作为约束条件，排序生成高阶互补云API推荐列表．

本文的主要贡献总结如下：

(1)首次提出并解决了面向服务软件开发的云API互补推荐问题，包括低阶互补推荐和高阶互补推荐两类问题．以统一的方式给出了互补云API推荐的形式化定义，使其更具一般性，更贴近服务化软

件开发的客观实际.

(2)设计了一个刻画云API互补关系的逻辑推理网络.该逻辑推理网络由投影、取反和交并三个基本逻辑算子组合而成,能够有效刻画云API互补关系的非对称性,消解替补关系噪声的影响.

(3)提出了一个基于概率逻辑推理网络的高阶互补云API推荐方法.该方法利用注意力机制获得表征能力更强的高阶互补云API表示,提升满足互补推荐高阶性的推荐效果.

(4)在真实云API数据集上进行大量实验,结果表明所提出的方法在低阶、高阶和混合阶互补云API推荐场景下均展示出更优的推荐效果.参数敏感性实验、实例分析和用户调查分析进一步证明了所提推荐方法的有效性和实用性.

本文第2节介绍云API生态的数据分析和高阶互补推荐问题定义.第3节给出了基于概率逻辑推理的高阶互补云API推荐方法的总体框架、构建过程和算法实现.第4节是实验设计与结果分析.第5节分析了本文研究所面临的有效性威胁.第6节分析并讨论了与本文相关的研究工作.第7节总结了全文工作并介绍了未来的工作方向.

2 数据分析和问题定义

2.1 数据分析

在介绍具体的高阶互补云API推荐建模方法之前,先回答以下两个基本问题(Basic Question,BQ):

BQ1.在实际服务化软件开发过程中,是否存在互补云API推荐的客观需要?

BQ2.在基于共同调用行为的云API互补关系建模中,是否存在功能相同的替补关系噪声?

为了回答BQ1,我们设计爬虫和解析程序:(1)从全球最大的云API开放平台ProgrammableWeb获取了6,452个Mashup应用和22,663个云API数据;(2)从华为应用市场获取了11,339个移动APP与1,946个云API数据.利用这两组数据,对应用实际调用的云API数量进行统计分析,绘制了如图2所示的应用调用云API数量的分布图.

从图2(a)所示的Mashup应用调用云API数量分布来看,40.14%的Mashup应用调用了2个及以上的云API.特别地,有22.45%的Mashup应用调用了2个云API,这表明Mashup应用实际开发中具有低阶互补云API推荐的客观需要,即已知一个云API进行互补云API推荐.有17.70%的Mashup应用调用了超过2个以上的云API,这表明Mashup应用开发具有高阶互补云API推荐的客观需要,即推荐的云API需要和查询的多个云API整体互补.从图2(b)华为应用市场APP调用云API数量的分布来看,调用2个及以上云API的APP比例高达90.16%,有77.55%的APP调用了超过2个以上的云API,进一说明了面向服务的移动应用开发中高阶互补云API推荐的实际需要.基于上述观察,可得出以下结论:服务化软件开发有多元化互补云API的客观需要,继而有效回答BQ1.

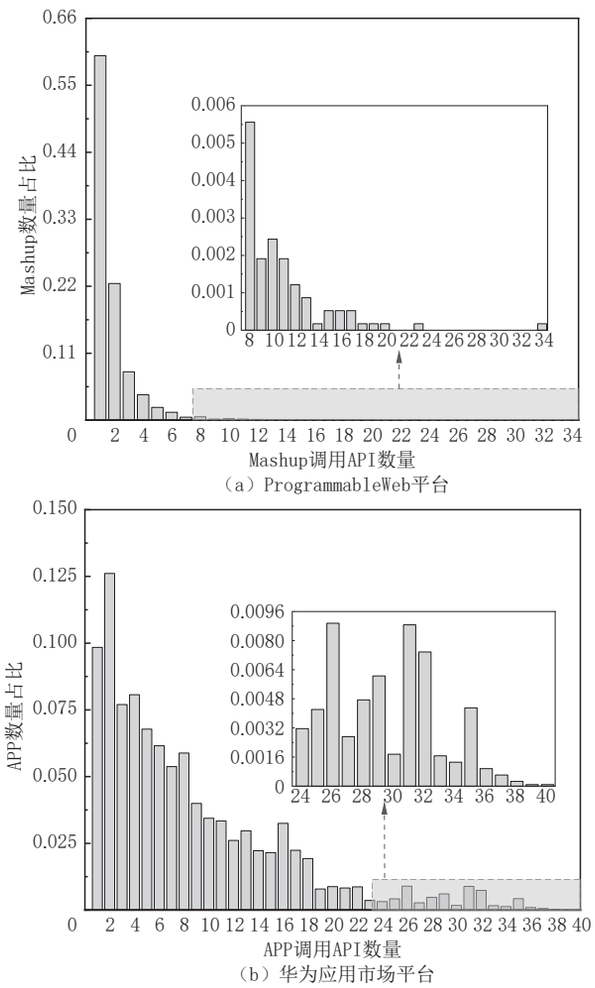

图2 应用调用云API数量分布图

为了回答BQ2,我们根据应用已调用的多个云API生成API对,统计API对之间的功能是否相同,进而分析基于共同调用行为的云API互补关系挖掘中是否存在功能相同的替补关系.具体来说,如若Mashup应用m调用了三个云API(a_1, a_2, a_3),那么可以生成三组API对:(a_1, a_2)、(a_1, a_3)和(a_2, a_3).如果其中(a_1, a_2)具有不同的功能,那么我们认为云API a_1, a_2之间存在潜在的互补关系;否则,是替补关

系. 特别地，对于云API对之间的功能是否相同问题的判定，我们可以通过ProgrammableWeb和华为应用市场中提供的云API的功能类别标签来进行判定. 图3(a)和图3(b)分别绘制了ProgrammableWeb平台和华为应用市场中调用了2个及以上云API数量的应用中具有不同功能和相同功能API对的分布情况.

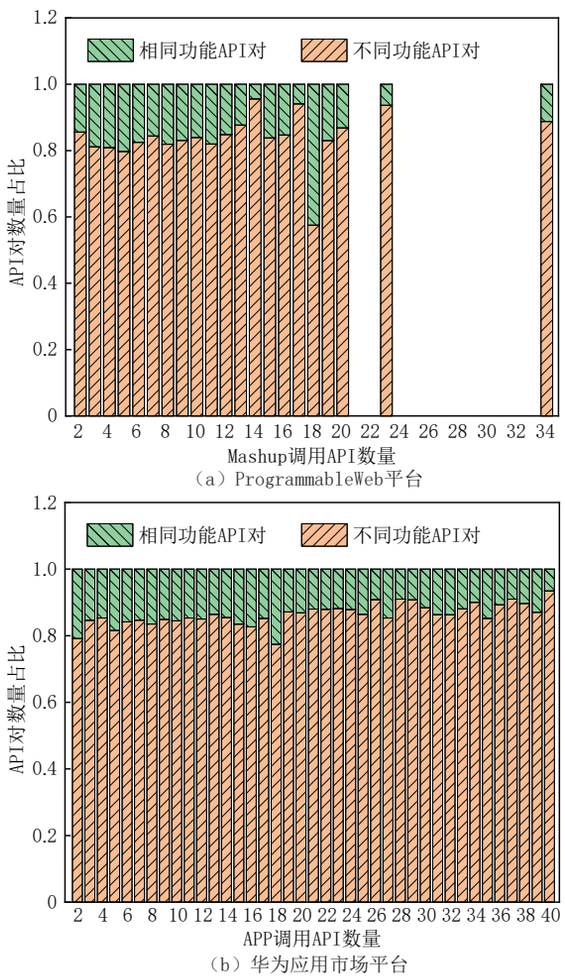

图3　应用调用不同功能和相同功能的云API对分布图

从图3(a)Mashup应用调用不同功能和相同功能的API对分布情况来看，除了调用了18个云API的Mashup相同功能云API对比例超过了20%，其余Mashup应用中具有相同功能云API对的比例均不超过20%，如图3(b)所示的APP中具有相同功能云API对的平均比例也不超过20%. 这些结果表明在实际面向云API的服务化应用开发时，开发者倾向于调用具有不同功能的互补云API进行混搭来构建Mashup/APP应用，这也进一步回答了BQ1：调用不同功能的多元化互补API是服务化软件开发的一个客观需要. 此外，我们注意到，在少部分的Mashup应用开发过程中，例如调用云API数量为18的Mashup占比只有0.0174%，开发者会调用具有相同功能的云API，即替补云API，这是因为开发者有时会采用冗余策略调用替补云API来提升Mashup应用的鲁棒性和用户体验. 然而，对于替补云API，开发者可以采用现有内容驱动的和基于偏好的云API推荐系统来获取. 基于上述观察，可以得出以下结论：在基于共同调用行为的云API互补关系挖掘中，存在功能相同的替补关系噪声，继而有效回答BQ2. 这也启示我们在利用共同调用行为挖掘云API之间的互补关系时，需要考虑其中客观存在的替补关系带来的噪声对互补关系建模的影响.

2.2　问题定义

为了使读者更好的了解面向服务软件开发中云API生态的交互关系和后续高阶互补推荐问题的定义，首先建立应用-云API交互图$G_{M\times A}$.

定义1.　应用-云API交互图$G_{M\times A}$是一个四元组：

$$G_{M\times A}=(V, E, \xi, T) \tag{1}$$

其中节点集合$V=M\cup A$，$M=\{m_1, m_2, \cdots, m_{|M|}\}$是应用的有穷集合，$A=\{a_1, a_2, \cdots, a_{|A|}\}$是云API的有穷集合. 由于应用集合$M$和云API集合$A$代表云API生态中两类不同实体的集合，故$M\cap A=\varnothing$. $E=\{0, 1\}$是关系集合，0表示应用和云API之间不存在调用关系，1表示应用和云API之间存在调用关系. ξ是转换函数，是$V\times V\rightarrow E$上的映射. 例如，应用$m\in M$与云API $a\in A$之间存在调用关系，那么$\xi(m, a)=1$，否则$\xi(m, a)=0$，也即节点m和a之间存在一条边$e=<m, a>\in E$. $T=\{t_1, t_2, \cdots, t_k\}$是功能属性等标签的有穷集合. 特别的，互补关系逻辑推理网络中的替补关系约束SC需要利用功能属性标签构建. 综上，将V中的元素作为节点，E中的元素作为边，T中的元素作为节点的属性，可生成应用-云API生态的交互图$G_{M\times A}$.

云API之间的共同调用等行为为分析云API的互补性提供了客观依据. 为了更好地表征和利用云API之间的共同行为，我们在$G_{M\times A}$的基础上，建立云API-云API共同行为图$CBG_{A\times A}$.

定义2.　云API-云API共同行为图$CBG_{A\times A}$是一个四元组：

$$CBG_{A\times A}=(A, W, \zeta, CB) \tag{2}$$

其中$A=\{a_1, a_2, \cdots, a_{|A|}\}$是云API的有穷集合. W是共同行为权重关系的集合. ζ是转换函数，是$A\times A\rightarrow W$上的映射. 因为云API之间的共同调用行为

可有效刻画云API间的互补关系，本文主要基于共同调用行为进行互补分析. 例如，云API $a_1 \in A$ 与 $a_2 \in A$ 被同一应用共同调用，则有 $\zeta(a_1, a_2)=1$，即 a_1 与 a_2 的共同调用行为权重 $w=1 \in W$. CB 是共同行为的有穷集合，如共同调用、共同关注等行为. 特别的，互补关系逻辑推理网络中的共同调用关系约束CC需要利用共同调用行为构建. 根据应用-云API交互图 $G_{M\times A}$，采用如下过程可构建 $CBG_{A\times A}$：遍历云API集合 A 中的不同云API对 (a_1, a_2)，基于 $G_{M\times A}$ 中应用和云API的历史调用行为，可判断 a_1 和 a_2 之间是否存在共同调用关系. 如果 a_1 和 a_2 之间存在共同调用行为，即 $\zeta(a_1, a_2)=1$，则 a_1 和 a_2 之间存在一条边，边上的权重为1，行为属性为共同调用. 据此过程可构建云API-云API共同行为图 $CBG_{A\times A}$，继而为云API互补推荐提供数据依托.

定义3. 高阶互补云API推荐. 将开发者在面向服务应用开发过程中已调用的多个云API构成的查询集合 $Q=\{a_{q_1}, a_{q_2}, \cdots, a_{q_n}\}$ 作为输入，利用云API-云API共同行为图 $CBG_{A\times A}$，互补云API推荐系统 R 为开发者生成互补云API推荐列表 ANS：

$$\begin{aligned} &R(Q)=\{a_1, a_2, \cdots\}=ANS \\ &s.t.\ \ Q\cap ANS=\varnothing,\ \ P(a_1|Q)\geqslant P(a_2|Q)\cdots \end{aligned} \tag{3}$$

其中 $P(a_i|Q, a_i\notin Q)$ 表示推荐云API $a_i \in ANS$ 在互补性和高阶性感知约束下的互补条件概率. 高阶性要求 a_i 与 Q 中的云API整体互补. 因此，互补云API推荐列表 ANS 中不包括查询集的云API，即 $a_i \notin Q$，并且推荐的云API按照与云API查询集合互补程度的大小排序. 特别地，当查询集 Q 中云API数量为1时，即 $|Q|=1$，高阶互补云API推荐问题则简化为低阶互补云API推荐问题.

3 方法设计

本文所提出的基于概率逻辑推理的高阶互补云API推荐方法PLR4HCCR整体框架如图4所示，共包括三个模块：将查询集中的云API嵌入到可用于支持逻辑推理的云API Beta概率嵌入编码模块，分析单一查询云API的互补云API表示的互补关系逻辑推理模块以及高阶互补关系约束感知的高阶互补云API生成模块.

PLR4HCCR方法整体工作流程如下：采用Beta概率嵌入编码技术，得到每个查询云API的Beta嵌入表示，并将其输入概率逻辑推理网络中. 利用概率逻辑推理网络分析查询集中的云API在共同调用关系和替补关系约束下的映射，包括利用投影算子在共同调用关系下进行非对称互补感知，利用取反算子对替补关系噪声进行消解，继而应用交并算子推理出与每个查询云API的具有潜在互补关系的Beta概率表示. 然后将所有查询云API的互补嵌入通过注意力网络进行处理，得到与查询集 Q 整体相互补的云API高阶互补基向量. 进一步，计算候选云API嵌入与高阶互补云API基向量之间的KL散度. 最后根据KL散度排序生成高阶互补云API推荐列表. 通过上述流程，可以使推荐的云API满足对推荐结果互补性和高阶性要求的同时，消解替补关系噪声对互补推荐结果的影响. 在接下来的章节中，我们将详细介绍PLR4HCCR方法的每个环节.

3.1 Beta概率嵌入编码

为了刻画开发者查询云API和推荐云API之间互补关系的不确定性，支持互补关系逻辑推理建模，受文献[20]的启发，本文采用Beta概率嵌入技术对云API实体及其之间的关系约束进行嵌入编码.

Beta概率分布由两个形状参数 α 和 β 来决定，不同于现有特征嵌入技术，Beta概率嵌入需要 $2d$ 个参数对云API的 d 个隐特征进行编码表示. 具体而言，利用Beta概率嵌入对云API a_i 嵌入编码时，我们采用两个长度为 d 的参数向量 $\boldsymbol{\alpha}_i=[e_{\alpha i\_1}, \cdots, e_{\alpha i\_d}]$ 和 $\boldsymbol{\beta}_i=[e_{\beta i\_1}, \cdots, e_{\beta i\_d}]$ 共同表示云API a_i 的 d 个隐藏特征，即：

$$E(a_i)=([e_{\alpha i\_1}, \cdots, e_{\alpha i\_d}], [e_{\beta i\_1}, \cdots, e_{\beta i\_d}])=B(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) \tag{4}$$

其中，云API a_i 的隐藏特征 j 由两个参数 $e_{\alpha i\_j}$ 和 $e_{\beta i\_j}$ 所定义Beta分布 $\mathrm{Beta}(e_{\alpha i\_j}, e_{\beta i\_j})$ 表示.

令云API a_i 的Beta概率嵌入表示为：$E(a_i)=B(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)$，那么云API查询集合 $Q=\{a_{q_1}, a_{q_2}, \cdots, a_{q_n}\}$ 经过Beta概率嵌入编码可表示为：

$$E(Q)=\{B(\boldsymbol{\alpha}_{q_1}, \boldsymbol{\beta}_{q_1}), \cdots, B(\boldsymbol{\alpha}_{q_n}, \boldsymbol{\beta}_{q_n})\} \tag{5}$$

类似地，云API之间的关系约束 r 采用Beta概率嵌入可表示为：

$$E(r)=([e_{\alpha r\_1}, \cdots, e_{\alpha r\_d}], [e_{\beta r\_1}, \cdots, e_{\beta r\_d}])=B(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) \tag{6}$$

本文在挖掘云API间互补关系时，使用到的关系约束（Relation Constraints，RC）主要包括，RC={共同调用关系约束（Co-Invoke Constraint，CC），替补关系约束（Substitution Constraint，SC）}.

采用Beta概率嵌入技术进行特征编码具有以

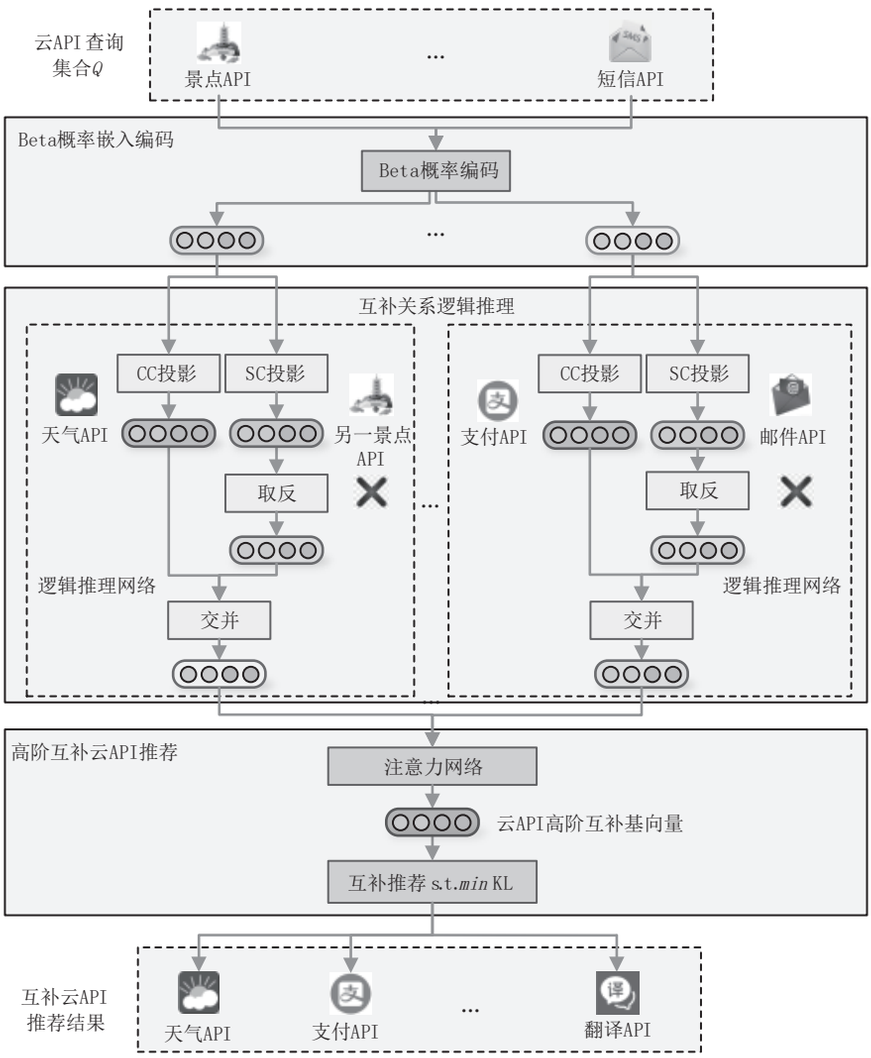

图4　PLR4HCCR方法整体框架

下优势：(1)Beta分布包含两个形状参数α和β，其中Beta分布的不确定性可通过微分熵反映[20]. 我们注意到，现有*One-Hot*和*Dense*嵌入技术也可以实现云API的特征嵌入编码. 但是*One-Hot*嵌入采用一位有效的状态码来进行特征表示，数量庞大的云API会使采用*One-Hot*嵌入得到的特征表示十分稀疏，并且特征空间会变得非常大. 尽管*Dense*嵌入技术通过引入权重矩阵可以将高维稀疏特征表示压缩成稠密嵌入向量，即采用d个参数实现云API的d维隐藏特征表示，但其难以有效支持云API间互补关系的不确定性表示. 通常，两个云API之间互补的原因是不确定的. 例如，云API a_1与a_2互补，可能是因为二者在功能上存在互补关系，而a_1与a_3互补可能是因为它们在云API响应数据类型方面互补. Beta概率嵌入采用d个Beta概率分布实现云API的隐藏特征的嵌入表示. 在模型参数更新完成后，不同云API的Beta嵌入表示不同，故云API隐藏特征间的Beta概率分布差异可刻画出云API间互补关系的不确定性，继而使本文所提出的方法能够生成更好的多元互补云API推荐结果. (2)基于Beta概率嵌入的云API编码表示可以利用Beta概率分布，有效实现逻辑“取反”和“交并”等逻辑关系，继而可以实现互补云API推荐中的替补噪声消解处理. (3)Beta概率分布支持闭环逻辑操作，可将一或多个Beta概率嵌入转换为一个新的Beta概率嵌入表示，从而有效支持我们构建互补关系逻辑推理网络.

3.2　互补关系逻辑推理

3.2.1　逻辑关系算子

为了对云API的互补关系进行逻辑推理建模，首先定义投影(*Projection*，*PJ*)、取反(*Negation*，*NG*)和交并(*Intersection*，*IS*)三个基本算子.

定义4.　投影算子*PJ*. 给定一个云API $q\in Q$及其关系约束$r\in R$，投影算子输出一个新的云API

a，其中q和a通过关系约束r连接，即：

$$PJ_r(q)=PJ_{B(\boldsymbol{\alpha}_r,\boldsymbol{\beta}_r)}(B(\boldsymbol{\alpha}_q,\boldsymbol{\beta}_q))=B(\boldsymbol{\alpha}_a,\boldsymbol{\beta}_a) \quad (7)$$

其中$B(\boldsymbol{\alpha}_r,\boldsymbol{\beta}_r)$，$B(\boldsymbol{\alpha}_q,\boldsymbol{\beta}_q)$和$B(\boldsymbol{\alpha}_a,\boldsymbol{\beta}_a)$分别表示关系$r$、云API q和a的Beta概率嵌入编码. 云API q和a通过关系约束r连接.

投影算子PJ在本质上是对一个云API Beta嵌入在关系约束下生成新的云API Beta概率嵌入，这一过程可通过如图5所示的多层感知机来实现. 具体而言，投影算子PJ的实现流程如下：首先将查询云API和关系的Beta嵌入向量进行拼接，然后将拼接后的向量经过多层感知机处理，继而输出与查询云API满足嵌入关系约束下的新的云API Beta概率嵌入表示. 例如，云API a_i与a_j被应用共同调用过，即$\zeta(a_i, a_j)=1$，参与模型训练时，我们给出$PJ_{CC}(B(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i))$的互补结果为$B(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j)$，供模型投影算子中共同调用关系嵌入$B(\boldsymbol{\alpha}_{CC}, \boldsymbol{\beta}_{CC})$参数及多层感知机中神经元参数的学习. 因此，在模型训练完成后，对于新的云API嵌入，则可利用共同调用关系嵌入$B(\boldsymbol{\alpha}_{CC}, \boldsymbol{\beta}_{CC})$和图5中的多层感知机进行非线性变化得到具有共同调用关系的云API嵌入.

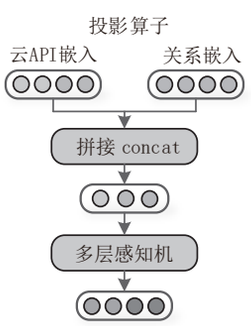

图5 投影算子实现流程

定义5. 取反算子NG. 给定一个云API $q\in Q$，取反算子输出一个新的云API a，即：

$$NG(q)=NG(B(\boldsymbol{\alpha}_q,\boldsymbol{\beta}_q))=B(\boldsymbol{\alpha}_a,\boldsymbol{\beta}_a) \quad (8)$$

其中$\boldsymbol{\alpha}_a=1/\boldsymbol{\alpha}_q$，$\boldsymbol{\beta}_a=1/\boldsymbol{\beta}_q$. 取反算子$\boldsymbol{NG}$是在Beta分布形状参数向量的计算上对两个形状参数取倒数. 采用这种方式进行取反的原因在于：(1)满足取反后的参数集合仍然是原集的要求，同时这种方式可同时满足取反的一个特性，即任意特征经过两次取反运算后得到的特征仍与原特征相同.(2)对于Beta分布而言，对其形状参数取倒数，可以最大程度地改变原分布与新分布之间的差异(体现在概率密度函数的改变)，进而可以最大程度地改变隐特征之间的差异，这也符合我们对取反的理解.

定义6. 交并算子IS. 给定一个云API集合$Q=\{q_1,q_2,\cdots\}$，交并算子输出一个新的云API a，即：

$$IS(Q)=IS(B(\boldsymbol{\alpha}_{q_i},\boldsymbol{\beta}_{q_i})|i=1,2,...)=B(\boldsymbol{\alpha}_a,\boldsymbol{\beta}_a) \quad (9)$$

其中$\boldsymbol{\alpha}_a=\sum w_i\alpha_{qi}$，$\boldsymbol{\beta}_a=\sum w_i\beta_{qi}$，权重系数$w_i$计算如下：

$$w_i=e^{\mathrm{Beta}(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)}/\sum_{j=1,2,...}e^{\mathrm{Beta}(\boldsymbol{\alpha}_j,\boldsymbol{\beta}_j)} \quad (10)$$

交并算子IS是把交并操作建模为多个Beta嵌入的概率密度分布的加权积，进而使得交并算子输出的新的云API有更强的综合表现能力.

3.2.2 互补关系逻辑推理网络

互补关系逻辑推理网络由投影、取反和交并三个基本逻辑算子按云API的互补关系刻画方法组合而成. 首先，云API之间的共同调用关系能够有效表征其互补关系，故基于云API-云API共同行为图$CBG_{A\times A}$，利用投影算子PJ可实现对查询云API $q\in Q$在共同调用关系约束CC下的投影映射$PJ_{CC}(q)$. 例如，在$CBG_{A\times A}$中，云API a_i和a_j之间存在共同调用关系，即$\zeta(a_i, a_j)=1$. 如果a_i和a_j作为训练集中的一条数据参与训练时，令$PJ_{CC}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)$投影结果为$B(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j)$，然后将其作为输入供投影算子中的共同调用关系嵌入$B(\boldsymbol{\alpha}_{CC}, \boldsymbol{\beta}_{CC})$参数及多层感知机中神经元参数的学习. 可见，投影算子$PJ_{CC}(q)$中的参数是由$CBG_{A\times A}$中的云API共同调用记录学习到的. 其次，考虑到在基于共同调用关系的云API互补关系建模中存在功能相同的替补关系，而这将会对互补云API的挖掘带来噪声. 因此对查询云API $q\in Q$在替补关系约束SC下的映射$PJ_{SC}(q)$应用取反算子进行映射得到$NG(PJ_{SC}(q))$，从而可以消解替补关系对互补云API挖掘的影响. 在此基础上，将共同调用关系下的互补映射和替补噪声消解后的结果应用交并算子IS，可进一步得到查询云API $q\in Q$非对称互补云API a的Beta概率表示如下：

$$IS(PJ_{CC}(q), NG(PJ_{SC}(q)))=B(\boldsymbol{\alpha}_a,\boldsymbol{\beta}_a) \quad (11)$$

根据式(11)中的逻辑运算关系，则可以构建非对称、消解替补关系影响的云API互补关系逻辑推理网络. 最后，依据该逻辑推理网络，通过$CBG_{A\times A}$上的图计算可实现查询云API集合中每一个云API的互补云API Beta概率表示.

3.3 高阶互补云API推荐

3.3.1 高阶互补基向量学习

对于低阶互补推荐，利用由式(11)构建的互补关系逻辑推理网络可直接得出查询云API q的互补云API Beta概率表示. 而高阶互补云API推荐

则要求推荐的云API与云API查询集$Q=\{a_{q_1},a_{q_2},\cdots,a_{q_n}\}$整体具有互补性. 为此，引入注意力机制为查询集$Q$中每个云API的互补逻辑推理结果分配不同权重，继而增强高阶互补云API基向量的表征能力.

令查询集Q中云API的Beta概率嵌入记为：$E(Q)=\{B(\boldsymbol{\alpha}_{q_1},\boldsymbol{\beta}_{q_1}),\cdots,B(\boldsymbol{\alpha}_{q_n},\boldsymbol{\beta}_{q_n})\}$，将$Q$中每一个云API经过逻辑推理网络之后得到的互补云API记为：$E(Q)^c=\{B(\boldsymbol{\alpha}^c_{q_1},\boldsymbol{\beta}^c_{q_1}),\cdots,B(\boldsymbol{\alpha}^c_{q_n},\boldsymbol{\beta}^c_{q_n})\}$. 然后将$E(Q)^c$输入到如图6所示的基于注意力网络的高阶互补基向量学习模块，则可学习到云API查询集Q的高阶互补云API的基向量.

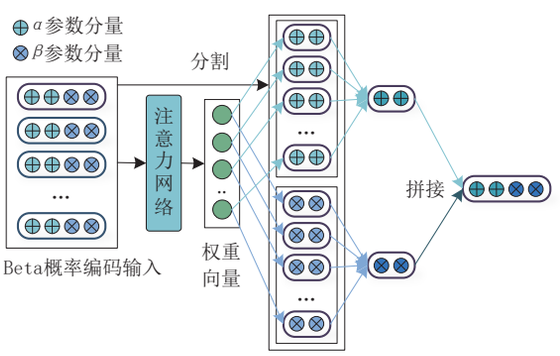

图6　基于注意力网络的高阶互补基向量学习

具体来说，对于$E(Q)^c$中的n个云API的Beta概率嵌入输入，先经过注意力网络计算得到对应云API的权重$w_i=e^{\mathrm{Beta}(\boldsymbol{\alpha}^c_{q_i},\boldsymbol{\beta}^c_{q_i})}\Big/\sum_{j=1}^{n}e^{Beat(\boldsymbol{\alpha}^c_{q_j},\boldsymbol{\beta}^c_{q_j})}$. 然后将所有互补云API Beta嵌入中的$\boldsymbol{\alpha}$和$\boldsymbol{\beta}$两组形状参数加权求和得到云API查询集$Q$的高阶互补基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$：

$$\begin{aligned}&Att(E(Q)^c)\\&=Att(\{B(\boldsymbol{\alpha}^c_{q_1},\boldsymbol{\beta}^c_{q_1}),\cdots,B(\boldsymbol{\alpha}^c_{q_n},\boldsymbol{\beta}^c_{q_n})\})\\&=B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})\end{aligned}\tag{12}$$

其中$\boldsymbol{\alpha}^c_{base}=\sum_{i=1}^{n}w_i\boldsymbol{\alpha}^c_{q_i},\boldsymbol{\beta}^c_{base}=\sum_{i=1}^{n}w_i\boldsymbol{\beta}^c_{q_i}$.

3.3.2　基于KL散度的互补推荐

KL散度可以很好地度量两个概率分布函数之间的距离. 这里我们根据学习到的高阶互补云API基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$与候选云API的Beta概率嵌入$B(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)$计算云API隐特征间的KL散度和. 隐特征间的KL散度距离的和越小，则表示候选云API与高阶互补基向量越接近，也即候选云API与查询云API集Q整体越互补. 在此基础上，取候选云API中与高阶互补云API基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$隐特征间的KL散度和前$K$小的云API，生成最终的互补云API推荐结果. 具体而言，对于Beta分布$\mathrm{Beta}(\alpha,\beta)$，其概率密度函数表示如下：

$$f(x;\alpha,\beta)=\frac{1}{\mathrm{Beta}(\alpha,\beta)}\times x^{(\alpha-1)}\times(1-x)^{(\beta-1)}\tag{13}$$

对于两个Beta分布$\mathrm{Beta}(\alpha_p,\beta_p)$与$\mathrm{Beta}(\alpha_q,\beta_q)$，利用公式(13)计算出它们的概率密度函数$p(x)$和$q(x)$后，它们之间的KL散度计算如下：

$$KL(p\|q)=\int p(x)*\log(p(x)/q(x))\mathrm{d}x\tag{14}$$

在高阶云API互补推荐应用中，假设Beta概率嵌入维度为3，最终学习到的高阶互补云API基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$中$\boldsymbol{\alpha}^c_{base}=[1,3,8]$，$\boldsymbol{\beta}^c_{base}=[3,5,4]$；候选云API a_i的Beta概率嵌入$B(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)$中$\boldsymbol{\alpha}_i=[11,8,9]$，$\boldsymbol{\beta}_i=[8,13,7]$. 也即$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$的三个隐特征分别用分布$\mathrm{Beta}(1,3)$，$\mathrm{Beta}(3,5)$和$\mathrm{Beta}(8,4)$表示；$B(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)$中的三个隐特征分别用分布$\mathrm{Beta}(11,8)$，$\mathrm{Beta}(8,13)$和$\mathrm{Beta}(9,7)$表示. 那么根据公式14，高阶互补云API基向量与候选云API隐特征间的KL散度为$\mathrm{KL}(f(x;1,3)\|f(x;11,8))+\mathrm{KL}(f(x;3,5)\|f(x;8,13))+\mathrm{KL}(f(x;8,4)\|f(x;9,7))$. 综上，根据此过程可计算出高阶互补云API基向量与所有候选云API之间KL散度. 最终，取候选云API中与$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$的KL散度和前$K$小的云API则可生成高阶互补云API推荐结果.

3.4　方法实现

3.4.1　损失函数设计

优化PLR4HCCR的目标是最小化高阶互补基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$与真实互补云API $ans\in ANS$（正样本）的Beta嵌入$B(\boldsymbol{\alpha}_{ans},\boldsymbol{\beta}_{ans})$之间的KL距离. 此外，为了提升模型的学习效果，在训练过程中引入负样本，用来最大化高阶互补基向量$B(\boldsymbol{\alpha}^c_{base},\boldsymbol{\beta}^c_{base})$与负样本云API *neg* Beta嵌入$B(\boldsymbol{\alpha}_{neg},\boldsymbol{\beta}_{neg})$之间的KL距离. 据此作为优化目标，设计损失函数如下：

$$\begin{aligned}Loss=&-\log\sigma(\gamma-KL(B(\boldsymbol{\alpha}_{ans},\boldsymbol{\beta}_{ans}),\\&B(\boldsymbol{\alpha}_{res},\boldsymbol{\beta}_{res})))-\log\sigma(KL(B(\boldsymbol{\alpha}_{neg},\boldsymbol{\beta}_{neg}),\\&B(\boldsymbol{\alpha}_{res},\boldsymbol{\beta}_{res}))-\gamma)\end{aligned}\tag{15}$$

其中γ表示期望的KL距离，σ表示*sigmod*激活函数.

3.4.2　高阶互补推荐算法设计

算法1给出了PLR4HCCR方法的设计与实现.

算法1. 高阶互补云API推荐算法.

输入:数据集D;
数据集划分比例r;
训练步数$steps$;
训练批次大小$bsize$;
推荐云API数量K;
输出:互补云API推荐列表$List$

1. 初始化参数
2. D_{train}, D_{test}, D_{valid}←按比例r划分数据集D
3. FOR $step$ IN range($steps$): //模型训练
4. 按$bsize$分批处理D_{train}
5. FOR (Q, ans) IN $D_{train-bsize}$:
6. neg ← 为查询集Q生成负样本
7. FOR a_q IN Q:
8. B_q^{cc}←$PJ_{CC}(a_q)$//共同调用关系投影
9. B_q^{sc}←$PJ_{SC}(a_q)$//替补关系投影
10. B_q^{u-sc}←$NG(B_q^{sc})$//替补云API取反
11. B_q^c←$IS(B_q^{cc}, B_q^{u-sc})$//互补云API表示
12. B_{base}^c←$Att(B_1^c, B_2^c, \ldots, B_n^c)$//计算$Q$互补基向量
13. 根据损失函数$Loss$计算损失值
14. 根据损失值进行梯度下降更新参数
15. FOR Q IN D_{test}: //模型测试
16. FOR a_{qi} IN Q:
17. B_q^c←按步骤8-11计算q的互补云API表示
18. B_{base}^c←$Att(B_1^c, B_2^c, \ldots, B_n^c)$
19. l.append($KL(B_{base}^c, B_{cand})$)//KL由小到大排序
20. $List.append(Top-K(l))$//生成K个互补云API
21. RETURN $List$

算法1中,首先根据数据集划分比例r将数据集划分训练集、测试集和验证集.3-14行是模型的训练阶段.模型训练的计算复杂度主要依赖于训练步长$steps$和训练数据集D_{train}的大小.15-20行是模型的测试阶段,对测试集中的查询云API集合Q推荐K个互补的云API.

4 实验与分析

为了验证方法PLR4HCCR的有效性,本文利用两个真实世界云API生态数据集进行广泛的实验,以回答以下7个研究问题(Research Question,RQ):

RQ1. 所提出的方法是否优于现有主流的基线方法?

RQ2. 利用混合阶交互数据训练的模型在任意阶互补推荐场景下的迁移效果如何?

RQ3. 所提出的方法在不同云API推荐数量K下的推荐性能表现如何?

RQ4. 嵌入维数对方法PLR4HCCR的推荐性能影响如何?

RQ5. 方法PLR4HCCR的收敛性与效率如何?

RQ6. 方法PLR4HCCR在实际应用中的推荐效果如何?

RQ7. 方法PLR4HCCR在实际云API互补推荐系统应用中的可行性和用户体验如何?

4.1 准备

4.1.1 数据集

本文使用的实验数据集包括面向Mashup开发的PWA(ProgrammableWeb API)和面向移动应用开发的HGA(HUAWEI AppGallery API)两个真实云API数据集.PWA源自于从ProgrammableWeb获取的数据,其中包括6,452个Mashup应用和22,663个云API的基础信息,如类别标签、描述信息等,以及Mashup和云API之间的历史调用关系.HGA源自于从华为应用市场获取的数据,包括11,339个移动APP与1,946个云API的类别,文档描述等信息,以及它们之间的真实调用关系数据.

PWA和HGA均是面向服务混搭开发的真实开放云API生态数据集,其中PWA是面向Mashup开发的云API数据集,HGA是面向移动应用开发的云API数据集,二者是云API在Web端应用开发和移动端应用开发中的典型代表.此外,PWA和HGA在应用功能类别、云API功能类别、应用平均调用云API数量和云API交互密度等方面也有所不同.本文所采用的PWA和HGA数据集统计特征如表1所示.

表1 PWA和HGA数据集统计特征

数据集	PWA	HGA
应用类型	Mashup应用	移动应用
应用数量	6,452	11,339
云API数量	22,663	1,946
应用功能类别数量	315	78
云API功能类别数量	160	85
平均调用云API数/应用	3.11	9.40
云API交互矩阵密度	1.432%	2.433%

为了研究所提出的高阶互补云API方法对不同阶互补推荐问题的求解效果,我们使用调用云API数量大于1的应用构建仅用于1阶互补推荐的数据集$D_{1\text{-}order}$,使用调用云API数量大于2的应用构建仅用于二阶互补推荐的数据集$D_{2\text{-}order}$,使用调用API数

量大于3的应用构建仅用于三阶互补推荐的数据集$D_{3\text{-}order}$. 具体地，指定阶的云API互补数据集可以通过算法2来生成.

算法2.　指定阶云API互补数据集生成算法.

输入：指定数据集阶数*order*

　　应用调用云API历史*M_APIList*

输出：*order*阶互补云API数据集*Dictionary*

1. FOR *APIList* IN *M_APIList*：
2. 　*len* ← *Len*(*APIList*)
3. 　IF *len* <= *order*：
4. 　　Continue；
5. 　//生成查询集的索引组合列表*QIdx_List*
6. 　*QIdx_List*←[0，…，*len*-1]中长度为*order*的子集，共C_{len}^{order}种组合方式
7. 　FOR *idxs* IN *QIdxs_List*
8. 　　*queryList*←[*APIList*[*i*] for *i* in *idxs*]
9. 　　*ansList*←[*APIList*[*i*] for *i* not in *idxs*]
10. 　　*Dictionary*[*queryList*]←*ansList*
11. RETURN *Dictionary*

算法2中首先遍历应用已调用的云API列表*M_APIList*(1行)，如果其调用的云API数量小于等于指定的阶数*order*，则该应用不参与数据集的生成(2-4行). 接着生成查询集索引组合列表*QIdx_List*(6行). 然后遍历*QIdx_List*为每一个查询云API组合*queryList*生成互补云API(7-9行)，并将结果存放到数据字典*Dictionary*[*queryList*](10行). 综上，利用数据集PWA和HGA作为算法2的输入，则可生成其相应的一阶、二阶和三阶互补云API数据集，相关数据集的统计特征如表2所示.

表2　实验数据集统计信息

数据集名称	数据集类型	数据量	格式	云API占比
PWA	$PWA_{1\text{-}order}$	12,772	$(a_1)\rightarrow a_2$	100%
	$PWA_{2\text{-}order}$	74,469	$(a_1, a_2)\rightarrow a_3$	80.42%
	$PWA_{3\text{-}order}$	379,008	$(a_1, a_2, a_3)\rightarrow a_4$	67.72%
HGA	$HGA_{1\text{-}order}$	38,588	$(a_1)\rightarrow a_2$	100%
	$HGA_{2\text{-}order}$	363,150	$(a_1, a_2)\rightarrow a_3$	98.09%
	$HGA_{3\text{-}order}$	2,061,330	$(a_1, a_2, a_3)\rightarrow a_4$	96.50%

注：云API占比指相应阶数据集中包含的云API占总云API的比例.

此外，我们将一阶，二阶和三阶互补云API数据集进行混合，可构建用于分析任意阶互补推荐场景下的数据集. 在利用上述数据集进行互补推荐实验时，将数据集按比例7∶2∶1划分为训练集、验证集和测试集. 实验数据和PLR4HCCR的实现代码已开源到了Github①，以供其他相关人员进一步深入研究.

4.1.2　评价指标

为了评价高阶互补云API推荐方法的效果，采用以下指标评价推荐结果的准确性和替补程度.

(1)命中率(Hit Rate，HR). 衡量推荐列表中是否出现真实结果. HR越高表明推荐命中准确性效果越好，计算如式(16)所示.

(2)归一化折损累计增益(Normalize Discounted Cumulative Gain，NDCG). 用于排序评估，考虑推荐列表中结果的顺序. NDCG越高表明推荐结果的排序质量越好，计算如式(17)所示.

(3)平均倒数排名(Mean Reciprocal Rank，MRR). 考虑真实结果在排序列表中的排名. MRR越高表明真实结果在推荐列表中的排名越靠前，计算如式(18)所示.

(4)替补程度(Substitution Degree，SD)，衡量推荐列表中云API与查询云API的替补程度，SD越低表明推荐云API越符合互补要求，计算如式(19)所示.

$$\mathrm{HR@K}=\frac{1}{N}\sum_{i=1}^{N}hits(i) \tag{16}$$

$$\mathrm{NDCG@K}==\frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{K}\frac{pre_{i,j}}{\log_2(j+1)}}{\sum_{j=1}^{T_i}\frac{1}{\log_2(j+1)}} \tag{17}$$

$$\mathrm{MRR}=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{p_i} \tag{18}$$

$$\mathrm{SD@K}=\frac{1}{N\times M\times K}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{j=1}^{K}s_{i,m,j} \tag{19}$$

其中N表示测试数据的个数. K表示推荐结果列表中云API个数. $hits(i)$表示第i条测试数据的结果是否在推荐结果列表中，是则为1，否则为0. $pre_{i,j}$表示第i条测试数据的推荐结果j是否是真实项，是则为1，否则为0. T_i表示第i条测试数据的推荐结果中真实项的个数. p_i表示第i条测试数据的推荐结果中第一个真实项在推荐结果中的位置，若推荐列表不存在真实项，则$p_i\rightarrow\infty$. M表示查询集中云API个数，$s_{i,m,j}$表示第i条测试数据的推荐结果j是否与查询集中第m个查询项有替补关系，是则为1，否则为0.

4.1.3　实验环境与参数

本文所有相关实验运行的硬件环境：英特尔至强处理器，内存128G. 软件环境：编程语言Python3.9，编程环境VSCode. 方法PLR4HCCR的超参数设置见表3所示.

① https://github.com/hey-mem/PLR4HCCR

表3 超参数设置

参数	参数值(PWA/HGA)
嵌入维数d	64/96
推荐云API数量K	20/20
学习率lr	0.0002/0.0002
训练批次大小$bsize$	512/512
训练步数$steps$	15 000/80 000

4.2 方法比较(RQ1)

由于目前还没有关于高阶互补云API推荐的相关研究工作，我们采用8种有代表性的可用于高阶互补推荐的方法作为基线方法(1-3为启发式推荐方法，4-8为深度学习方法)进行比较分析. 此外还使用所提方法的变体和在不同阶互补推荐场景下评估我们的方法.

(1)Random，从所有云API列表中随机选择K个云API作为推荐结果[19].

(2)Popular-N，考虑云API调用的头部效应，将云API被Mashup调用次数作为流行度，把流行度较高的K个云API作为推荐结果[19].

(3)Apriori(AP)，利用频繁项集挖掘云API之间的关联规则，利用云API之间的关联规则来推荐与查询目标集中云API相关联的云API[21].

(4)FM，因子分解机(Factorization Machine，FM)是一种经典的推荐模型，构造显式的低阶特征交互进行云API推荐[22].

(5)AFM，在FM特征交互层与输出层之间增加了注意力网络，为不同的特征交互分配适当的权重进行云API推荐[23].

(6)DeepFM，利用一种新的神经网络架构结合FM的特征交互和多层神经网络来学习更复杂的特征表示，提升模型的表达能力[24].

(7)NAFM，在特征交互层与输出层之间添加注意力网络和多层神经网络，在捕获高阶特征交互关系的同时也学习每个特征交互之间的重要性[25].

(8)DynamicRec，使用卷积神经网络对云API交互调用序列进行建模. 基于当前查询目标集输入的序列，动态预测下一个云API的调用概率来生成推荐结果[26].

(9)PLR4HCCR-NG，是本文所提方法的变体，在逻辑推理网络中去掉了替补关系约束投影、取反和交并算子操作.

为了回答RQ1，证明所提出方法的优越性. 我们分别利用PWA和HGA的混合阶数据集对模型进行训练，测试两个不同数据集训练后的模型在不同阶互补云API推荐场景下的性能表现. 表4和表5分别给出了互补云API推荐数量为20时，所提方法PLR4HCCR在PWA和HGA数据集下与对比方法在HR、NDCG、MRR和SD指标下的实验结果.

观察表4和表5，可以得出以下结论.

(1)HR、NDCG和MRR分别从命中率、推荐顺序和推荐结果排名三个方面评价推荐列表与查询云API之间的互补性，从这三个指标的实验结果可以看出，PLR4HCCR在所有推荐场景下的表现均优于八种基线方法. 这是因为Random完全采用随机策略，推荐结果的效果最差. Popular-N和AP基于流行度和关联规则挖掘互补关系的能力远弱于DynamicRec. 而对于FM、AFM、DeepFM和NAFM四种方法，它们擅长处理云API实体的特征交互问题，而对于高阶推荐问题，需要同时考虑多个输入云API所拥有的共同特征，它们在处理这一类型问题时会失去优势，导致推荐效果变差，DynamicRec是一种基于多目标的序列推荐的方法，故相较于其它七种基线方法，该方法的推荐结果较好. 从P vs. D行HR、NDCG和MRR三种指标的提升效果来看，我们的方法相较于DynamicRec方法有着显著的提升.

(2)SD指标从推荐结果的替补程度角度来评价推荐方法对于替补关系噪声的消解能力. 从八种基线方法中可以看出，在没有去除替补关系噪声模块的情况下，由于替补关系噪声的影响，SD会随着HR、NDCG和MRR指标值的增加而有所增加，这说明这些基线方法均无法同时满足高互补推荐效果和低替补程度两种需求. 而PLR4HCCR在逻辑推理模块中增加了去除替补噪声的模块，故使得PLR4HCCR可以在推荐出较为准确的互补推荐结果同时，大大降低推荐结果与查询集的替补程度.

(3)为验证PLR4HCCR中利用替补关系约束SC投影和取反算子对替补噪声处理的有效性，构建PLR4HCCR-NG去掉SC投影和取反算子操作. 从表4和表5中的P vs. P-NG行可以看出，引入SC投影和取反算子，尽管在不同互补推荐场景下HR、NDCG和MRR指标的结果略有下降，但SD指标在所有互补推荐场景下均有大幅度的下降. 这说明PLR4HCCR对替补噪声处理是有效的，可以在很大程度上降低推荐结果的替补程度，继而有效提升互补云API推荐结果的质量.

(4)在PWA数据集下，除Random和Popular-N

表4　基于PWA数据集的方法比较结果

方法	1阶互补推荐场景				2阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0461	0.0148	0.0126	0.0263	0.0441	0.0157	0.0138	0.0316
Popular-N	0.4418	0.2138	0.1600	0.0413	0.4687	0.2145	0.1508	0.0486
AP	0.4229	0.1823	0.1129	0.0795	0.4005	0.1583	0.0883	0.0735
FM	0.1677	0.0781	0.0554	0.0166	0.2267	0.1027	0.0704	0.0164
AFM	0.1828	0.0640	0.0307	0.0333	0.2247	0.0783	0.0376	0.0327
DeepFM	0.1914	0.0677	0.0383	0.0333	0.2380	0.0930	0.0572	0.0325
NAFM	0.2114	0.0889	0.0605	0.0499	0.2692	0.1048	0.0626	0.0501
DynamicRec	0.6570*	0.2828*	0.1859*	0.0721*	0.8105*	0.3694*	0.2537*	0.1042*
PLR4HCCR-NG	0.8884**	0.4112**	0.2744**	0.1202**	0.9466**	0.4420**	0.3005**	0.1218**
PLR4HCCR	**0.8453**	**0.3723**	**0.2420**	**0.0654**	**0.9172**	**0.4369**	**0.3016**	**0.0668**
P vs. D	28.66%	31.66%	30.12%	9.26%	13.16%	18.30%	18.90%	35.84%
P vs. P-NG	−4.85%	−9.46%	−11.83%	45.58%	−3.11%	−1.15%	0.36%	45.12%
方法	3阶互补推荐场景				混合阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0465	0.0164	0.0145	0.0328	0.0456	0.0156	0.0136	0.0303
Popular-N	0.4918	0.2121	0.1402	0.0535	0.4674	0.2135	0.1503	0.0478
AP	0.3805	0.1451	0.0779	0.0621	0.4013	0.1619	0.0931	0.0717
FM	0.3433	0.1562	0.1090	0.0161	0.2459	0.1123	0.0783	0.0164
AFM	0.3085	0.1085	0.0526	0.0323	0.2387	0.0836	0.0403	0.0328
DeepFM	0.3012	0.1239	0.0806	0.0319	0.2436	0.0949	0.0587	0.0325
NAFM	0.3458	0.1345	0.0818	0.0513	0.2754	0.1094	0.0683	0.0504
DynamicRec	0.8189*	0.3593*	0.2384*	0.1176*	0.7622*	0.3371*	0.2260*	0.0979*
PLR4HCCR-NG	0.9541**	0.4335**	0.2894**	0.1226**	0.9297**	0.4289**	0.2881**	0.1215**
PLR4HCCR	**0.9358**	**0.4367**	**0.2977**	**0.0674**	**0.8995**	**0.4153**	**0.2804**	**0.0665**
P vs. D	14.28%	21.55%	24.88%	42.69%	18.02%	23.19%	24.08%	32.06%
P vs. P-NG	−1.92%	0.74%	2.89%	45.05%	−3.26%	−3.17%	−2.66%	45.25%

两种方法外，其余方法都出现一阶推荐效果小于二阶推荐效果，二阶推荐效果小于三阶推荐效果的现象. 出现这种现象的原因是$PWA_{3\text{-}order}$和$PWA_{2\text{-}order}$中包含的云API分别仅占总云API的67.72%和80.42%，这导致在PWA中，高阶推荐部分在拟合的过程中会偏向$PWA_{3\text{-}order}$和$PWA_{2\text{-}order}$中的云API，继而出现高阶推荐的效果优于低阶推荐的现象.

(5)在一阶互补推荐场景下，HGA数据集下的推荐效果优于PWA数据集下的推荐效果. 这是因为$PWA_{1\text{-}order}$数据集的API交互矩阵密度为1.432%，而$HGA_{1\text{-}order}$数据集API交互矩阵密度为2.433%. HGA数据集下的云API交互更为密集，一阶推荐的拟合程度更好，从而使HGA下的一阶推荐效果更好.

(6)从表5观察到，在1阶推荐场景下，即只有一个查询云API时，PLR4HCCR只需通过逻辑推理网络的共同调用投影算子即可获得互补云API的基向量. 然而随着互补推荐问题阶数的增加，PLR4HCCR需要对多个云API进行逻辑推理，并在推荐时要同时考虑多互补推理结果之间的关系. 这增加了问题的复杂程度，使得HGA数据集在方法PLR4HCCR下出现一阶推荐效果优于二阶推荐效果，二阶推荐效果优于三阶推荐效果的现象.

为进一步验证PLR4HCCR的泛化能力及其优越性，我们混合PWA和HGA两个数据集进行实验. 特别地，在数据集混合时，云API使用原有标签，以确保云API类别归属的正确性，进而实现有效的替补关系噪声消解. 此外，混合后的训练集、验证集和测试集分别由HGA和PWA两个数据集各自的训练集、验证集和测试集混合组成，分属两个数据集的云API之间没有交互信息. 表6给出了互补云API推荐数量为20时，所提方法PLR4HCCR在混合数据集下与对比方法在HR、NDCG、MRR和SD指标下的实验结果.

从表6的推荐结果来看，由于PWA和HGA混合数据集中一部分云API之间缺乏交互信息，从而导致FM、AFM、DeepFM和NAFM等方法在混合数据集下的推荐效果较差. 虽然基于多目标序列推荐方法DynamicRec的推荐性能优于其它七种基线方法，但在混合数据集下，其推荐性能相较于在如

表5 基于HGA数据集的方法比较结果

方法	1阶互补推荐场景				2阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0496	0.0171	0.0083	0.0441	0.0428	0.0154	0.0081	0.0447
Popular-N	0.6933	0.3076	0.2047	0.0450	0.6903	0.3118	0.2098	0.0856
AP	0.4270	0.2032	0.1423	0.1112	0.2740	0.1124	0.0652	0.1091
FM	0.4785	0.2477	0.1836	0.0499	0.4529	0.2463	0.1887	0.0492
AFM	0.5509	0.2700	0.1893	0.0999	0.5702	0.2779	0.1976	0.0974
DeepFM	0.5226	0.2409	0.1645	0.0832	0.5305	0.2419	0.1630	0.0817
NAFM	0.5567	0.2546	0.1765	0.0803	0.5461	0.2456	0.1674	0.0810
DynamicRec	0.8005*	0.3651*	0.2452*	0.0802*	0.7920*	0.3597*	0.2428*	0.1209*
PLR4HCCR-NG	0.9598**	0.4536**	0.3084**	0.0886**	0.9168**	0.4233**	0.2864**	0.1006**
PLR4HCCR	**0.9547**	**0.4469**	**0.3037**	**0.0309**	**0.8773**	**0.4076**	**0.2778**	**0.0318**
P vs. D	19.27%	22.43%	23.85%	61.49%	10.77%	13.33%	14.43%	73.73%
P vs. P-NG	−0.53%	−1.46%	−1.54%	65.14%	−4.31%	−3.71%	−2.99%	68.45%
方法	3阶互补推荐场景				混合阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0324	0.0147	0.0075	0.0449	0.0416	0.0157	0.0080	0.0446
Popular-N	0.6895	0.3074	0.2039	0.1049	0.6910	0.3089	0.2061	0.0785
AP	0.2630	0.1060	0.0598	0.0961	0.3213	0.1406	0.0891	0.1055
FM	0.4312	0.2433	0.1906	0.0484	0.4542	0.2457	0.1876	0.0492
AFM	0.5560	0.2765	0.2000	0.0943	0.5590	0.2748	0.1956	0.0972
DeepFM	0.5179	0.2406	0.1643	0.0798	0.5237	0.2411	0.1639	0.0815
NAFM	0.5476	0.2510	0.1741	0.0816	0.5501	0.2504	0.1727	0.0810
DynamicRec	0.7802*	0.3491*	0.2323*	0.1406*	0.7909*	0.3579*	0.2401*	0.1139*
PLR4HCCR-NG	0.8925**	0.4123**	0.2801**	0.1084	0.9231**	0.4297**	0.2916**	0.0992**
PLR4HCCR	**0.8136**	**0.3802**	**0.2618**	**0.0299**	**0.8819**	**0.4116**	**0.2811**	**0.0309**
P vs. D	4.28%	8.92%	12.72%	78.71%	11.50%	14.99%	17.08%	72.91%
P vs. P-NG	−8.85%	−7.77%	−6.53%	72.40%	−4.46%	−4.22%	−3.61%	68.90%

注：*表示基线方法最好的实验结果，**表示消融变体方法PLR4HCCR-NG的实验结果．P vs. D表示PLR4HCCR相对于DynamicRec的提升，P vs. P-NG表示PLR4HCCR相对于PLR4HCCR-NG的提升．

图5和图6所示单数据集PWA或HGA中的性能有所下降，这表明方法DynamicRec的泛化能力较差．从PWA和HGA混合数据集下评价指标的提升效果来看，本文所提方法PLR4HCCR相较于DynamicRec以及其他七种基线方法均有着显著的提升，进一步说明了PLR4HCCR具有较好的泛化能力和推荐性能优势．

4.3 迁移效果分析(RQ2)

为了分析使用混合阶数据集$PWA_{hybrid\text{-}order}$和$HGA_{hybrid\text{-}order}$训练出的方法PLR4HCCR在不同阶互补推荐场景下的迁移效果，我们与$PWA_{1\text{-}order}$，$PWA_{2\text{-}order}$，$PWA_{3\text{-}order}$和$HGA_{1\text{-}order}$，$HGA_{2\text{-}order}$，$HGA_{3\text{-}order}$数据集训练的PLR4HCCR进行对比，分析他们在1-3阶互补推荐场景下的推荐结果．表7和表8分别给出了在PWA和HGA数据集下PLR4HCCR的迁移效果．

从表7和表8可以观察到：

(1)无论在PWA还是在HGA数据集下，使用混合阶数据集所训练的PLR4HCCR模型能很好地应用到不同阶互补推荐场景中，并且推荐效果没有显著的下降．这是因为使用混合阶数据集进行训练时，可使PLR4HCCR很好的捕捉到云API之间的互补关系．可见，使用$PWA_{hybrid\text{-}order}$和$HGA_{hybrid\text{-}orde}$数据集训练的模型可有效进行迁移，解决任意阶场景下云API互补推荐问题．

(2)在PWA和HGA数据集的1阶互补推荐场景下，使用混合阶数据集所训练模型的互补推荐表现远优于仅使用一阶数据集所训练模型，这是因为一阶数据集的数据量较少，使模型无法学习到所有潜在的互补关系，从而导致模型拟合过度，推荐效果变差．

4.4 互补云API推荐数量影响分析(RQ3)

合理的互补云API推荐数量对推荐方法的性能和用户体验均有一定影响．图7和图8分别给出了在PWA和HGA数据集下，PLR4HCCR在不同互补推荐场景中，互补云API的推荐数量K分别为3、

表6　基于PWA和HGA混合数据集的方法比较结果

方法	1阶互补推荐场景				2阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0214	0.0073	0.0034	0.0208	0.0211	0.0074	0.0038	0.0230
Popular-N	0.4583	0.2014	0.1313	0.0316	0.5496	0.2410	0.1555	0.0716
AP	0.3971	0.1690	0.1102	0.0849	0.4846	0.1927	0.1039	0.1105
FM	0.2617	0.1260	0.0908	0.0228	0.2701	0.1406	0.1084	0.0292
AFM	0.1353	0.0842	0.0715	0.0203	0.1498	0.0949	0.0808	0.0257
DeepFM	0.2580	0.1236	0.0900	0.0278	0.2946	0.1463	0.1091	0.0428
NAFM	0.1524	0.0461	0.0192	0.0210	0.1500	0.0429	0.0161	0.0277
DynamicRec	0.5852*	0.2597*	0.1678*	0.0704*	0.6649*	0.2936*	0.1918*	0.1017*
PLR4HCCR-NG	0.9449**	0.4554**	0.3147**	0.1008**	0.9338**	0.4343**	0.2948**	0.1059**
PLR4HCCR	**0.9284**	**0.4376**	**0.2971**	**0.0497**	**0.8979**	**0.4192**	**0.2866**	**0.0395**
P vs. D	58.64%	68.49%	77.06%	29.31%	35.04%	42.77%	49.42%	61.18%
P vs. P-NG	−1.75%	−3.90%	−5.61%	50.66%	−3.84%	−3.47%	−2.80%	62.72%
方法	3阶互补推荐场景				混合阶互补推荐场景			
	HR↑	NDCG↑	MRR↑	SD↓	HR↑	NDCG↑	MRR↑	SD↓
Random	0.0187	0.0066	0.0035	0.0240	0.0204	0.0071	0.0036	0.0226
Popular-N	0.5705	0.2491	0.1603	0.0919	0.5261	0.2305	0.1490	0.0650
AP	0.4765	0.1891	0.1033	0.1073	0.4527	0.1836	0.1058	0.1009
FM	0.2690	0.1428	0.1108	0.0325	0.2670	0.1365	0.1033	0.0282
AFM	0.1535	0.0977	0.0830	0.0282	0.1462	0.0923	0.0784	0.0248
DeepFM	0.2923	0.1445	0.1072	0.0494	0.2816	0.1382	0.1021	0.0400
NAFM	0.1569	0.0457	0.0181	0.0313	0.1531	0.0449	0.0178	0.0266
DynamicRec	0.6606*	0.2860*	0.1839*	0.1160*	0.6369*	0.2798*	0.1811*	0.0960*
PLR4HCCR-NG	0.9170**	0.4240**	0.2873**	0.1107**	0.9319**	0.4379**	0.2989**	0.1058**
PLR4HCCR	**0.8451**	**0.3917**	**0.2672**	**0.0350**	**0.8905**	**0.4162**	**0.2836**	**0.0414**
P vs. D	27.92%	36.97%	45.35%	69.84%	39.80%	48.75%	56.58%	56.88%
P vs. P-NG	−7.85%	−7.61%	−6.97%	68.40%	−4.45%	−4.96%	−5.13%	60.87%

注：*表示基线方法最好的实验结果，**表示消融变体方法PLR4HCCR-NG的实验结果．P vs. D表示PLR4HCCR相对于DynamicRec的提升，P vs. P-NG表示PLR4HCCR相对于PLR4HCCR-NG的提升．

表7　基于PWA数据集的模型迁移结果

互补推荐场景	数据集	HR↑	NDCG↑	MRR↑	SD↓
1阶	$PWA_{1\text{-}order}$	0.3548	0.1949	0.1057	*0.0522*
	$PWA_{hybrid\text{-}order}$	0.8453	0.3723	0.2420	*0.0654*
2阶	$PWA_{2\text{-}order}$	0.9034	0.4152	0.2771	*0.0722*
	$PWA_{hybrid\text{-}order}$	0.9172	0.4369	0.3016	*0.0668*
3阶	$PWA_{3\text{-}order}$	*0.9279*	0.4321	0.2928	*0.0610*
	$PWA_{hybrid\text{-}order}$	*0.9358*	0.4367	0.2977	*0.0674*

表8　基于HGA数据集的模型迁移结果

互补推荐场景	数据集	HR↑	NDCG↑	MRR↑	SD↓
1阶	$HGA_{1\text{-}order}$	0.4645	0.1431	0.0560	0.0601
	$HGA_{hybrid\text{-}order}$	0.9436	0.4469	0.3037	0.0309
2阶	$HGA_{2\text{-}order}$	0.8479	0.3827	0.2528	*0.0252*
	$HGA_{hybrid\text{-}order}$	0.8773	0.4076	0.2778	*0.0318*
3阶	$HGA_{3\text{-}order}$	*0.8157*	*0.3813*	*0.2629*	*0.0279*
	$HGA_{hybrid\text{-}order}$	*0.8136*	*0.3802*	*0.2618*	*0.0299*

5、10、20、30、40、50时，HR、NDCG、MRR和SD的变化情况．从图7和图8可以观察到：

（1）当互补云API推荐数量K从3增加到20时，不同阶数推荐场景下的HR、NDCG和MRR的结果均呈现较大幅度的增加，这表明增加云API推荐数量可有效提升互补云API的推荐准确性，同时PLR4HCCR也可适应不同阶互补推荐场景的需要．然而，当推荐数量K从20增加到50时，HR、NDCG和MRR提升的幅度逐渐变缓，出现这种现象是由于随着K越来越大，大部分互补云API已经出现在推荐列表中．在互补云API推荐数量K从3增加到50时，我们看到SD呈现增速下降甚至负增长的变化过程，这是因为替补噪声的存在，随着K的增加，越多具有替补关系的云API出现在推荐列表中的概率逐渐增大，但当K增加到一定程度时，PLR4HCCR消除替补关系噪声的能力得到了充分发挥．随着K的进一步增加，PWA数据集下SD开始逐渐下降，HGA数据集下SD增速减缓．此外，较

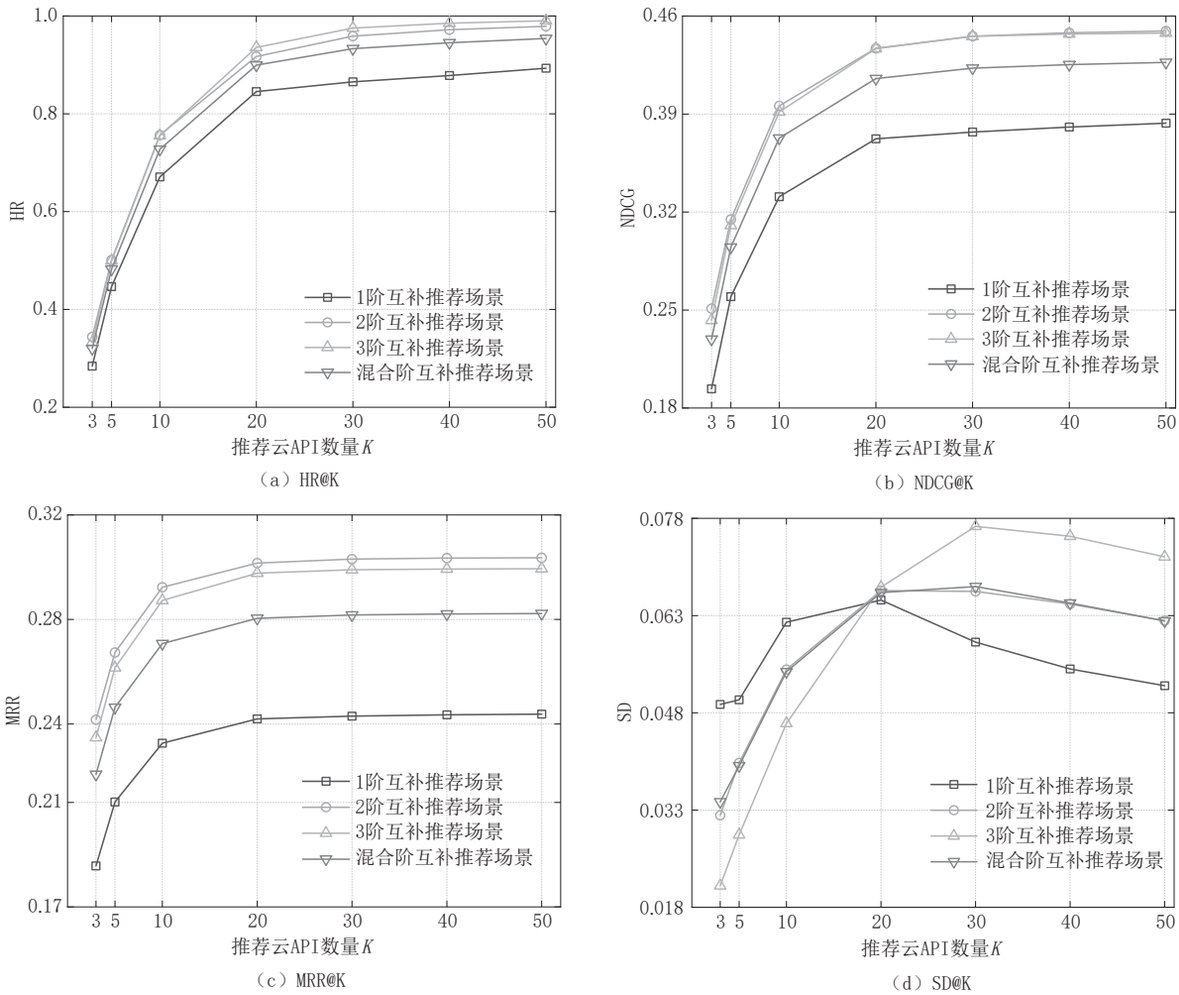

图7 PWA数据集下不同推荐云API数量K的影响

大数量的推荐列表反而会给开发者带来二次选择的困难. 因此，在本文其它实验中我们设置互补云API推荐数量K为20，可获得理想的推荐效果.

(2)在图7中，PWA数据集在不同推荐场景的指标HR、NDCG和MRR出现高阶互补推荐场景的推荐效果大于低阶互补推荐场景的推荐效果，这是由于PWA数据集下高阶数据中云API占比较低所导致的，这与在4.2节结论(4)中的观察一致. 在图8中，HGA数据集在不同推荐场景的指标HR、NDCG和MRR出现高阶互补推荐场景的推荐效果小于低阶互补推荐场景的推荐效果，这是由于随着互补推荐问题阶数的增加，模型需要考虑的交互因素也在增加，这与在4.2节观察(6)中的现象一致.

4.5 嵌入维数影响分析(RQ4)

嵌入维数d用来决定采用多少隐特征来表征云API，即Beta概率嵌入编码中Beta分布的数量，d的值越小，意味着使用较少的隐特征来表征云API，反之亦然. 图9和图10分别给出了PWA和HGA数据集下嵌入维数d从32增加到512时，PLR4HCCR在不同推荐指标下的变化情况. 我们发现过于小或者过于大的d值均不能获得较好的推荐效果. 当d为过小时，由于嵌入维数太小，没有足够多的特征来表征云API，从而使PLR4HCCR提取特征的能力较弱，出现欠拟合现象. 当d过大后，模型参数的空间复杂度变大，逐渐出现过拟合现象，模型的推荐效果也逐渐变差，综上，由于两个数据集的数据量、数据分布等方面存在差异，综合图9和图10的实验结果，在PWA中取d为64；在HGA中取d为96. 此外，PWA中每个功能标签下平均拥有7.23个云API；HGA中每个功能标签下平均拥有16.57个云API，基于数据集的上述差异，我们有以下观察：(1)当模型拟合效果不佳时，HGA数据集更容易推荐出与查询集功能相似的结果，相比于PWA，HGA下的SD指标会更大；(2)当模型拟合效果良好时，由于HGA中的云API功能标签数少于PWA数据集中的云API功能标签数，这使得在HGA数据集下，

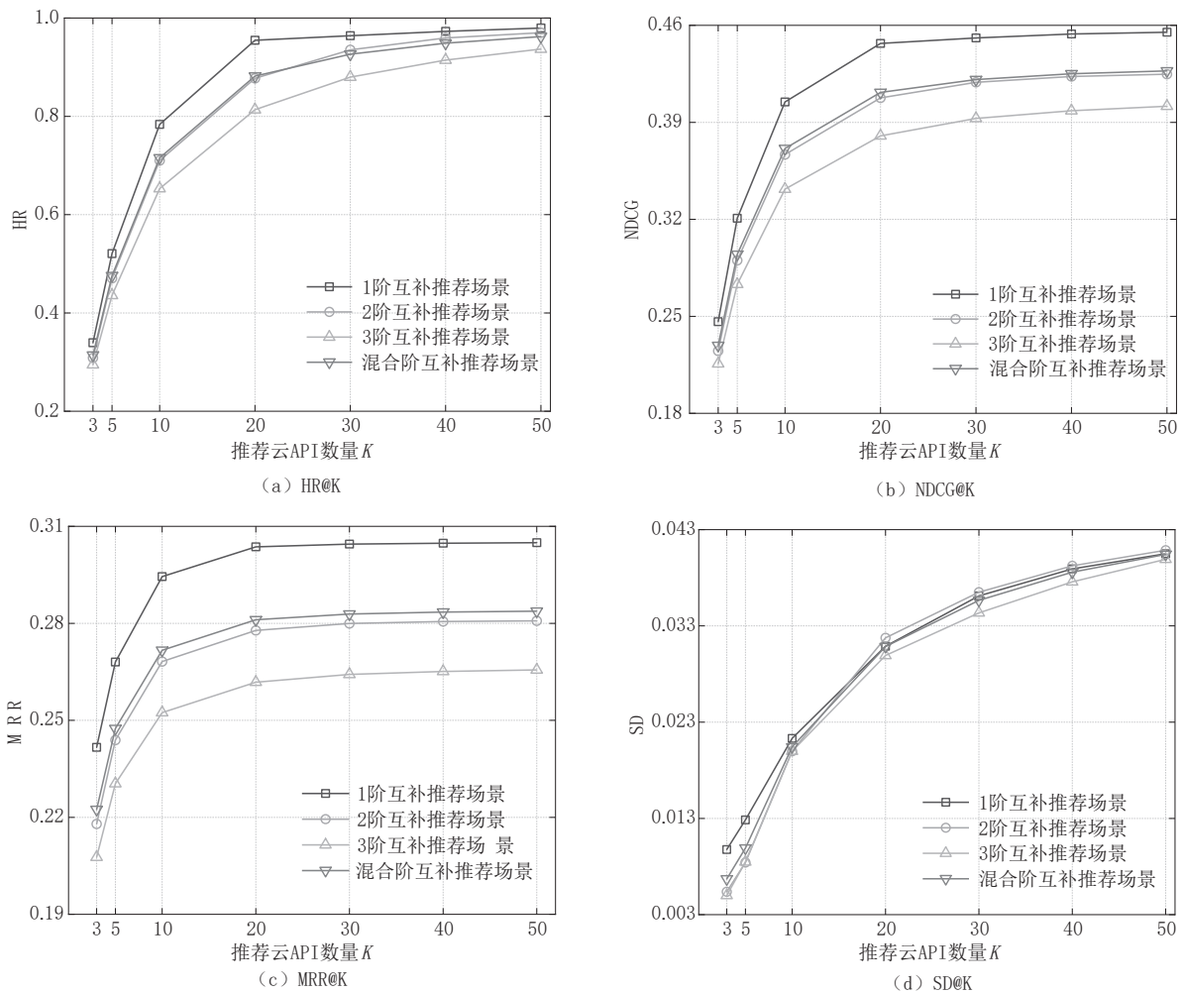

图8　HGA数据集下不同推荐云API数量K的影响

PLR4HCCR方法的逻辑推理部分的去噪模块可以更加充分地学习到噪声特征，进而在去除替补关系噪声时更加有效，即在模型拟合效果良好时，PLR4HCCR的去噪能力在HGA下更强. 以上两点的观察和分析也解释了图9(d)和图10(d)之间会出现变化的原因.

4.6　收敛性与效率分析(RQ5)

为了分析PLR4HCCR的收敛情况，图11和图12分别记录了PLR4HCCR在利用PWA和HGA进行训练过程中，不同互补推荐场景下HR、NDCG、MRR、SD、LOSS及模型训练时间的变化情况.

(1)从图11(a)～(c)和图12(a)～(c)的结果看，随着训练次数的增加，PLR4HCCR的推荐结果指标出现一个从较大幅度提升到逐渐变缓的过程，其原因是PLR4HCCR刚开始尚未找到局部最优解，由此出现了随着训练步长增加，推荐指标出现大幅提升. 但是当训练到一定的步长之后，PLR4HCCR已获得局部最优解，推荐指标和损失函数趋于稳定，这充分表明所提方法PLR4HCCR具有较好的收敛性.

(2)从图11(d)和图12(d)的SD指标看，由于PWA和HGA两个数据集中云API功能标签数量及云API数量本身的差异，使得图11(d)和图12(d)中SD的变化趋势有所不同，这是由于在模型拟合效果不佳时PWA数据集下的SD小于HGA数据集下的SD，而当模型拟合后，PWA数据集的SD大于HGA数据集的SD.

(3)从图11(e)、(f)和图12(e)、(f)结果看，PLR4HCCR的LOSS随着训练步长呈现一个从快速下降到逐渐稳定波动的变化过程，这表明PLR4HCCR可以通过一定次数的训练后快速收敛获得局部最优解. 而且PLR4HCCR训练所需的时间和训练步长呈现一个线性增长的关系，这一结果也与我们针对算法1高阶互补云API推荐算法训练所需的时间在理论分析方面的结果一致.

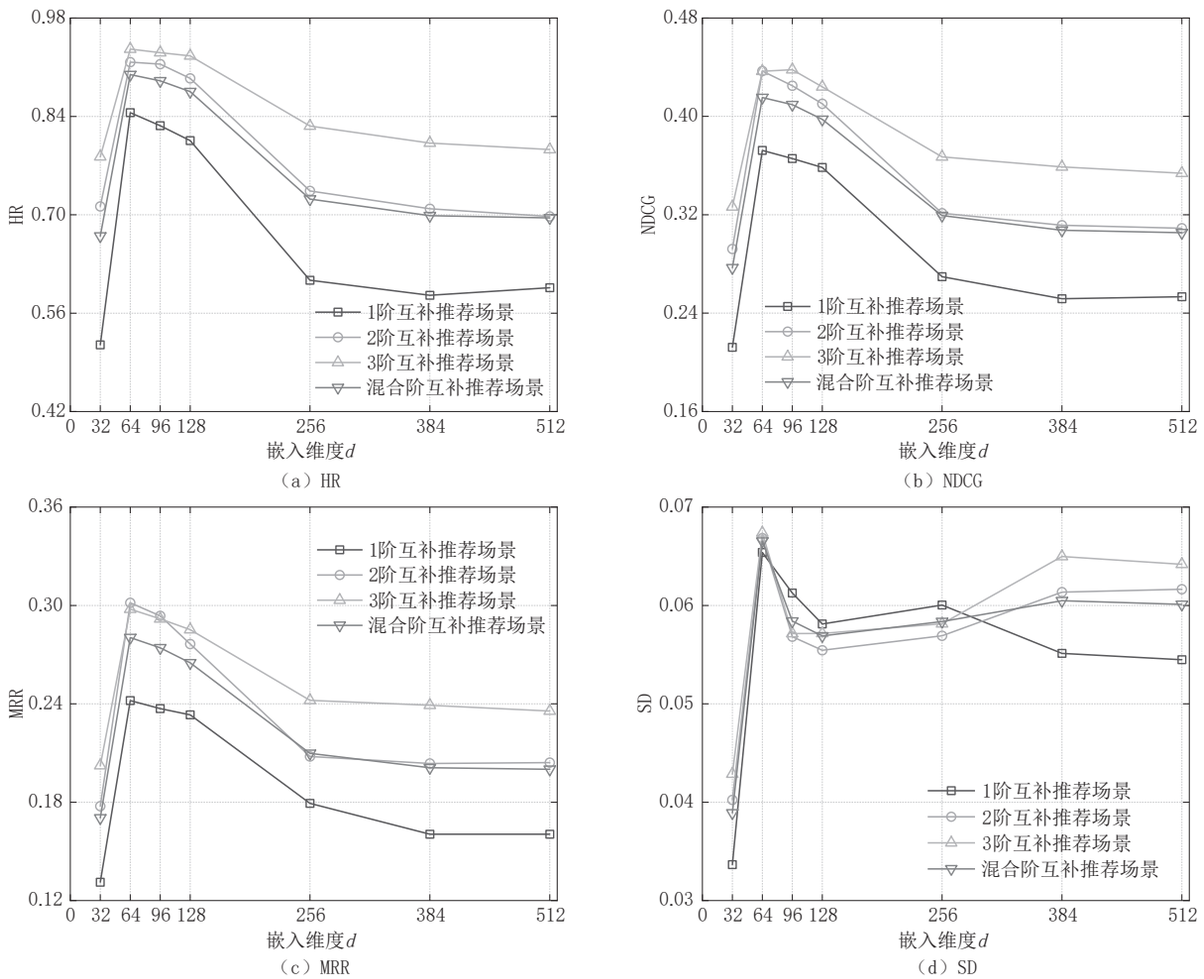

图9 PWA数据集下不同嵌入维数下的推荐指标变化

4.7 推荐实例分析(RQ6)

为了分析PLR4HCCR在实际云API互补推荐系统中的应用效果，表9给出了PLR4HCCR与对比基线方法基于PWA数据集在1-3阶推荐场景下的三个真实推荐实例结果. 为便于观察，利用编号表示云API，编号所表示的具体云API可从本文在Github2共享的数据集中获取，表9中推荐结果加粗的编号表示推荐正确的互补云API.

从表9可以直观地看出，PLR4HCCR推荐结果在大部分推荐场景下的HR、NDCG、MRR和SD指标优于其他八种基线方法. 特别是Random不采用任何推荐策略，随机为查询云API生成推荐结果，使得Random方法在不同性能指标下的结果均是最差的. 这也启示我们针对云API互补推荐问题需设计有效的互补推荐方法. 而我们提出的PLR4HCCR方法不仅可以应用到实际低阶/高阶互补推荐场景中，而且能够生成更优的推荐结果. 例如，对于低阶互补推荐应用，在表9中我们将单一查询云API (120)作为目标云API后，基于PLR4HCCR的互补推荐系统会根据互补关系推理推荐761、340等云API. 其原因是在ProgrammableWeb平台中Amazon Marketplace Web Service(120)，Facebook Graph(340)共同组建过Ewelike应用，Amazon Marketplace Web Service (120)，Amazon SimpleDB (761)共同组建过应用50 Shops. 对于高阶互补推荐应用，我们将查询云API集合(130,290,304)作为查询云API集合后，基于PLR4HCCR的互补推荐系统会根据整体互补关系推荐41、62、77、79、354、578等云API. 其可能的原因是由于WordPress (304)、SoundCloud (130)、Pocket (290)、Yelp Fusion (41)、Vimeo(62)、Bit. ly(77)、Foursquare (79)、Twilio SMS(354)、Gravatar(578)曾共同构建过Mashup应用We-Wired Web.

综上可见，本文提出的高阶互补云API推荐方法PLR4HCCR可有效应用到实际云API互补推荐系统中，生成开发者满意的推荐结果，同时能够便于

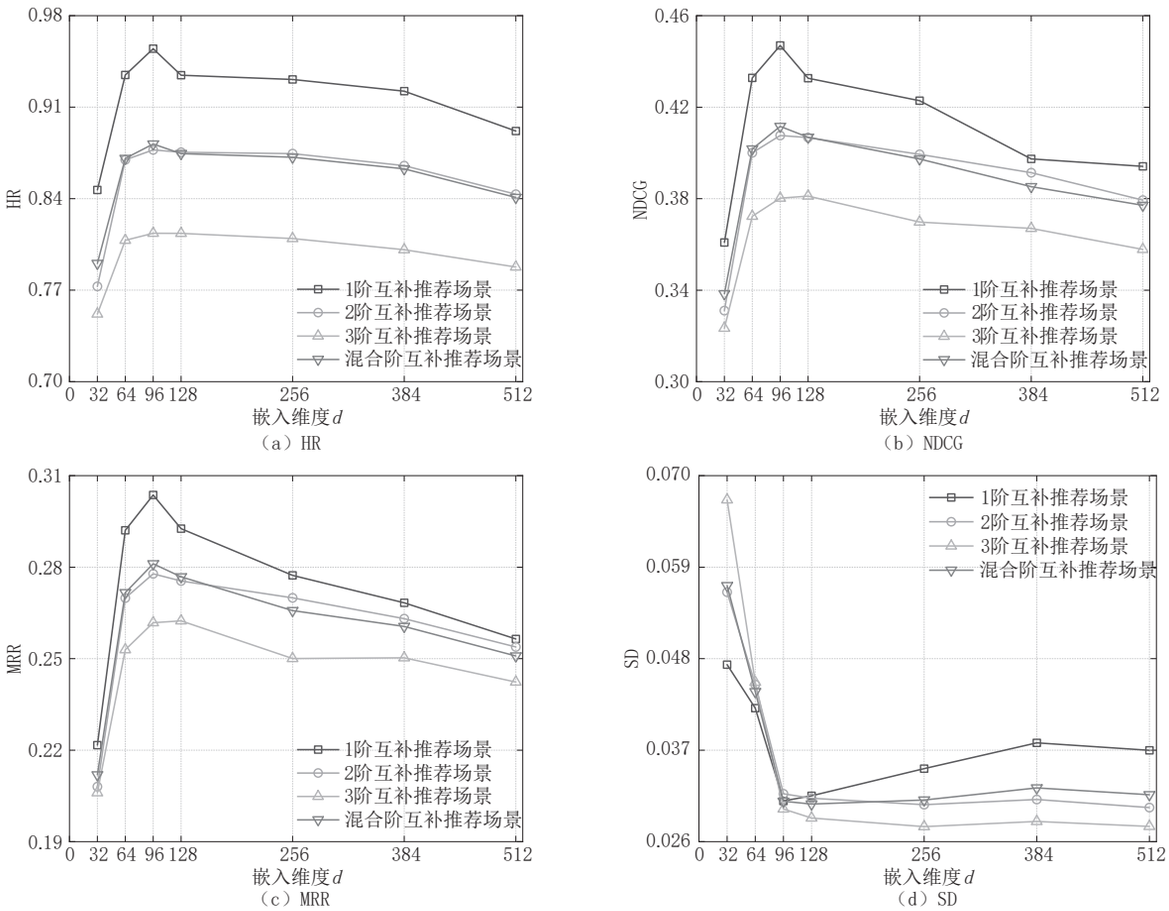

图10　HGA数据集下不同嵌入维数下的推荐指标变化

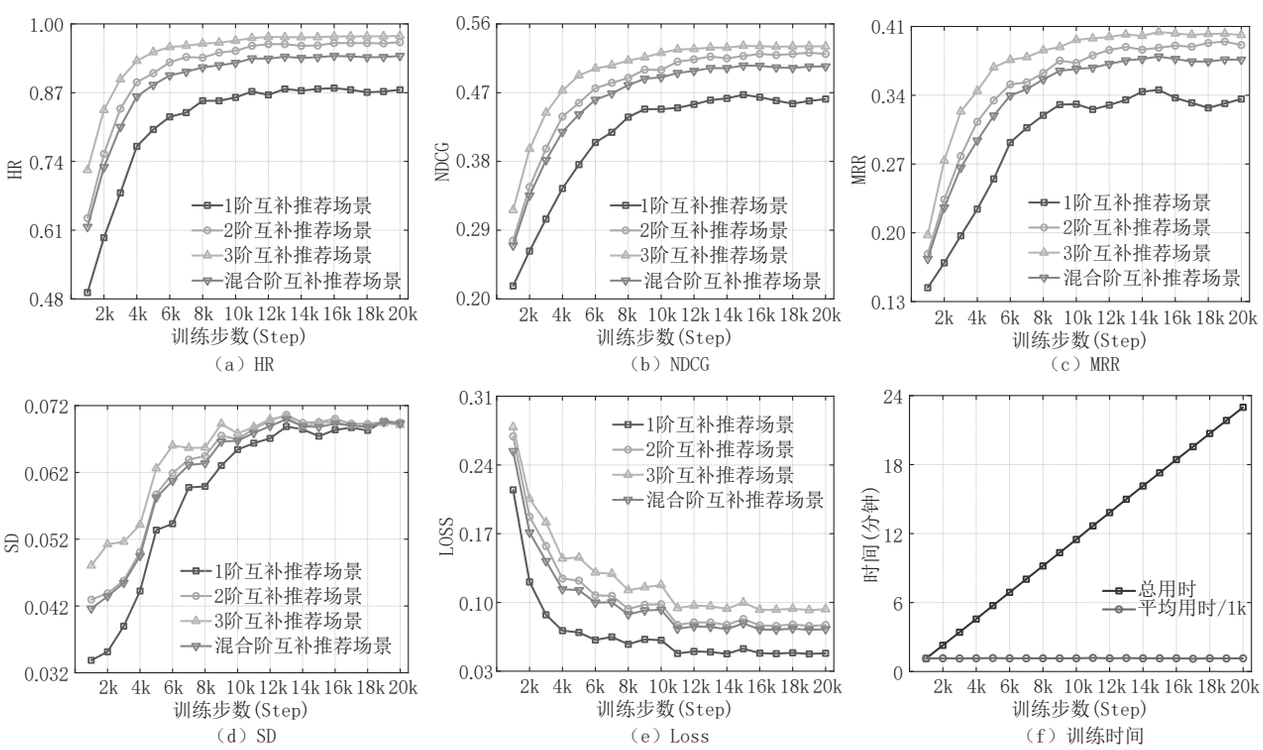

图11　PWA数据集下模型训练效果

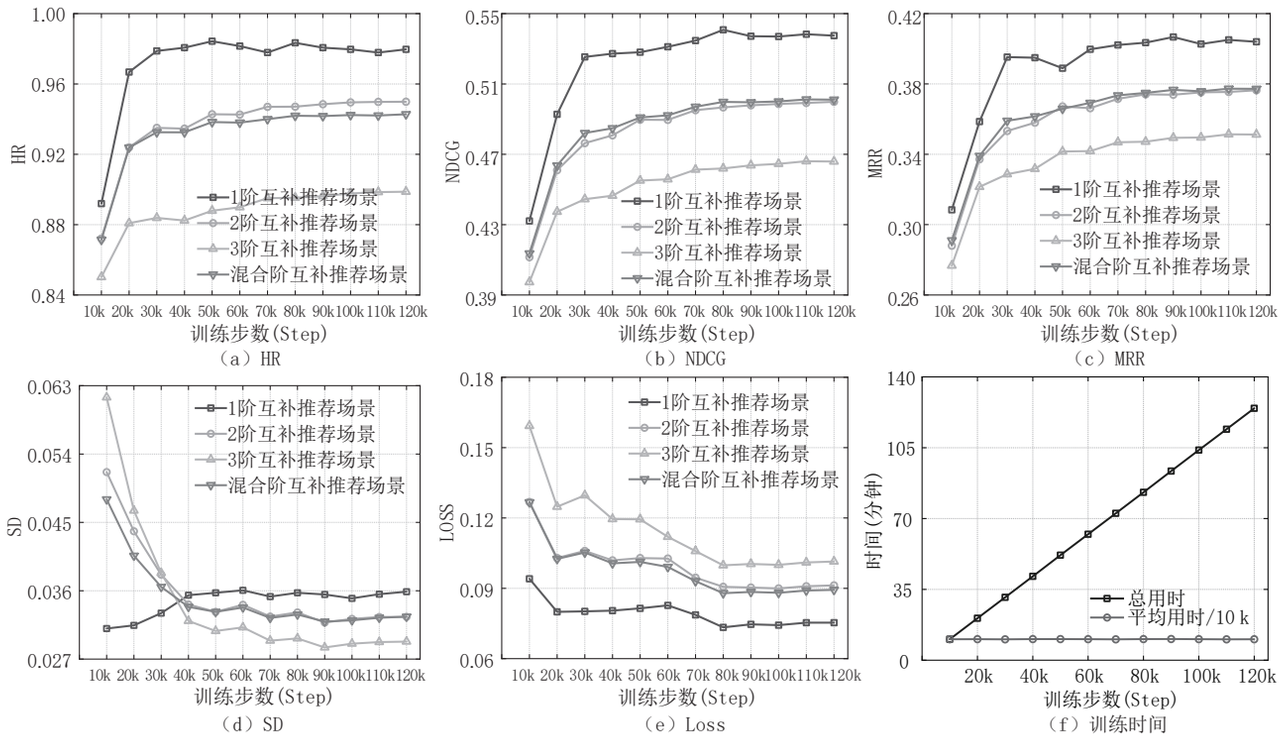

图12 HGA数据集下模型训练效果

云API服务提供者制定组合策略，提升对云API的收益.

4.8 云API互补推荐系统用户调查分析(RQ7)

为进一步探究本文所提云API互补推荐方法应用到实际推荐系统的可行性和用户体验，我们基于自研的互补云API推荐系统ReAPI，面向开发人员设计并发放了“关于云API互补推荐系统的问卷调查”，共计收集到76份问卷反馈. 表10统计了云API互补推荐系统的调查反馈结果.

从表10中调查项1和2的统计结果来看，超过80%参与问卷调查的开发人员能够通过举例，理解互补推荐的基本概念，这在一定程度上保证了面向云API互补推荐系统问卷调查项3-6反馈结果的可靠性. 调查项3的统计结果表明，绝大多数开发人员在进行软件开发时会考虑调用云API用于辅助开发，提升开发效率. 但是从调查项4的统计结果来看，开发人员检索云API的渠道各有不同，侧面反映出目前互联网中缺乏云API互补推荐系统. 进一步，基于调查项5对比传统云API推荐系统和云API互补推荐系统的用户体验，93.4%的开发人员认为结合互补推荐算法的云API互补推荐系统有助于软件开发. 调查项6采用用户打分的方式，定量了解开发人员使用云API互补推荐系统的意向. 调查统计结果显示，开发人员使用云API互补推荐系统的意向为8.1，具有很高的使用意向，进一步验证了本文所提云API互补推荐方法PLR4HCCR在实际云API互补推荐系统实际应用中的可行性.

5 有效性威胁分析

本文方法的有效性主要面临以下两个威胁.

(1)内部有效性威胁. 内部有效性威胁与实验数据集和实验对比的基线方法有关. 在实验数据集方面，除了在研究者们广泛采用的ProgrammableWeb平台构建云API数据集PWA验证本文所提的方法之外，我们还基于华为应用市场HUAWEI AppGallery获取了11,339个移动APP与1,946个云API之间的调用关系数据构建数据集HGA，并对所提出的高阶互补云API推荐方法有效性进行了验证. 由于从ProgrammableWeb平台和华为应用市场获取的数据均为真实应用与云API之间的实际调用关系，因此基于这些真实云API数据验证本文提出的云API高阶互补推荐方法是可行和可靠的. 此外，云API作为当前服务化软件开发与运行的必需核心要素，新的云API应用平台如RapidAPI，APIlist和ShowAPI等也逐渐涌现，因此在未来的工作中，拟采用更多的云API数据集进行实验，以进一步增强本文所提出方法的泛化能力. 在对比实验基

表9　云API互补推荐实例结果

查询云API	推荐方法	推荐结果(K=20)	HR	NDCG	MRR	SD
(120)	Random	[453, 835, 722, 134, 415, 341, 850, 566, 409, 719, 157, 276, 325, 700, 221, 149, 222, 475, 854, 275]	0	0	0	0.05
	popular-N	[18, 6, 60, 58, 4, 22, 8, 354, 110, 395, 79, 468, 51, 257, 199, 107, 480, 115, 243, 41]	0	0	0	0.05
	AP	[199, 765, 480, 395, 91, 90, 11, 149, 92, 97, 303, 5, 932, 107, 18, 597, 658, 256, 481, 734]	0	0	0	0.05
	FM	[928, 46, 459, 90, 409, 623, 73, 667, 820, 816, 431, 105, 9, 354, 838, 334, 161, 866, 794, 23]	0	0	0	0.1
	AFM	[683, 637, 779, 673, 122, 41, 780, 543, 816, 34, 378, 411, 273, 230, 119, 710, **340**, 137, 294, 864]	1	0.2398	0.0588	0.05
	DeepFM	[194, 515, 117, 568, 117, 752, 225, 53, 14, 387, 178, 158, 83, 300, 841, 66, 537, 940, 359, 563]	0	0	0	0.05
	NAFM	[418, 380, 6, 408, 445, 33, **348**, 44, 210, 699, 23, 596, 620, 753, 482, 748, 91, 522, 641, 312]	1	0.3333	0.1429	0.1
	DynamicRec	[6, 58, 18, 4, 110, 60, 51, 90, 468, 199, 8, **340**, 91, 79, 395, 257, 211, 107, 115, 159]	1	0.2702	0.0833	**0.05**
	PLR4HCCR	[888, **761**, 676, 18, 27, 90, **348**, 250, **340**, 4, 775, 211, **26**, 51, 60, 116, 692, 6, 315, 135]	**1**	**0.5964**	**0.5**	0.15
(116, 413)	Random	[569, 821, 900, 656, 736, 438, 771, 503, 571, 762, 120, 884, 424, 177, 15, 188, 567, 343, 241, 53]	0	0	0	0.075
	popular-N	[18, 6, 60, 58, 4, 22, 8, 354, 110, 395, 79, 468, 51, 257, 199, 107, 480, 115, 243, **41**]	1	0.3482	0.0909	0.025
	AP	[189, 626, 715, 701, 457, 714, 634, 385, **12**, 713, 94, 346, 8, 18, 4, 58, 60, 110, **41**, **79**]	1	0.3566	0.1111	0.025
	FM	[297, 402, 154, 502, 464, 678, 548, 461, 891, 492, 744, 410, 806, 431, 342, 642, 170, 911, 251, 483]	0	0	0	0.075
	AFM	[533, 297, 451, 6, 560, 476, 349, 9, 415, 915, 668, 434, 89, 273, 71, 620, 35, 22, 3, 568]	0	0	0	0.05
	DeepFM	[363, 23, 746, 96, 899, 191, 536, 447, 520, 283, 257, 519, 801, 890, 704, 796, **79**, 426, 701, 907]	1	0.2398	0.0588	0.075
	NAFM	[777, 622, 901, 527, 587, 147, 123, 734, 872, 508, **79**, 850, 564, 834, 904, 638, 157, 858, 824, 725]	1	0.2789	0.0909	0.1
	DynamicRec	[4, 110, 6, 60, 468, 58, 18, 8, **79**, 102, 51, **115**, 480, 90, 113, 340, 353, 182, 578, 91]	1	0.3502	0.1111	0.05
	PLR4HCCR	[459, **79**, 114, **12**, 348, 110, 715, 716, 386, 4, 18, 717, 703, **41**, 62, 60, 58, 130, **115**, 638]	**1**	**0.6046**	**0.5**	**0.025**
(130, 290, 304)	Random	[518, 139, 782, 714, 736, 875, 153, 397, 647, 174, 141, 241, 721, 266, 351, 24, 565, 457, 517, 512]	0	0	0	0.1667
	popular-N	[18, 6, 60, 58, 4, 22, 8, **354**, 110, 395, **79**, 468, 51, 257, 199, 107, 480, 115, 243, 41]	1	0.3857	0.125	0.0333
	AP	[217, 215, 459, 480, **574**, 158, **77**, 245, 577, 572, 576, **575**, 569, 570, 287, 291, 22, 44, 573, 94]	1	0.4647	0.2	0.1
	FM	[933, 473, 876, 606, 62, 324, 639, 500, 667, 218, 116, 29, 937, 906, **574**, 383, 303, 376, 311, 774]	1	0.3905	0.2	0.0166
	AFM	[30, 853, 753, 274, 451, 252, 694, 797, 340, **574**, 760, 35, 518, 369, 839, 324, 97, 494, 437, 755]	1	0.2890	0.1	0.0333
	DeepFM	[606, 95, 720, 27, 276, 20, 305, 689, 861, 410, 225, 241, 733, 365, 885, 123, 679, 401, 918, 466]	0	0	0	0.05
	NAFM	[461, 849, 573, 735, 460, 837, 261, 612, 203, 292, **578**, 351, 743, 614, 788, 438, 173, 676, 213, 932]	1	0.3556	0.1111	0.05
	DynamicRec	[6, 4, 115, 5, 18, **79**, **77**, 58, 60, 110, 303, 302, 340, 353, 468, 22, 149, 289, 291, 293]	1	0.4227	0.1667	0.0667
	PLR4HCCR	[58, 580, **77**, **354**, 44, 22, 110, 579, **62**, 6, 159, 114, **203**, **41**, 60, **79**, 246, 115, 353, **578**]	**1**	**0.6109**	**0.3333**	**0.0167**

表10 云API互补推荐系统用户调查反馈结果统计

序号	调查项问题描述	选择项	结果
1	在电商领域,商品互补推荐旨在向用户推荐可能会一起购买的商品.例如,用户购物车里已有“鼠标”和“显示器”,据此互补推荐系统推断该用户可能正在组装一台电脑,进而推荐与已选择商品互补的“键盘”、“耳机”等商品,继而提升用户体验。经过上述举例,对于电商领域的互补推荐,您是否已经有了基本的了解?	是	89.5%
		否	10.5%
2	在面向服务软件开发领域,云API互补推荐旨在向开发者提供可能会一起调用的云API.例如,开发者当前所开发的应用中已调用“墨迹天气”云API,据此互补推荐系统推荐与已经调用云API所互补的云API,如“百度地图”云API,实现在地图上可视化显示天气功能,继而提升软件开发效率和用户体验.经过上述举例,对于面向服务软件开发的云API互补推荐,您是否已经有了基本的了解?	是	80.3%
		否	19.7%
3	作为开发人员,如果您想开发出一款轻量级应用,您是否会考虑调用第三方云API辅助开发?	是	94.7%
		否	5.3%
4	您一般通过什么渠道检索并选择云API?(多选)	开发经验	50.7%
		搜索引擎	77.3%
		好友推荐	32.0%
		ChatGPT	44.0%
		其他	2.7%
5	下面两张图(图14(a)和14(b))分别展示了传统基于内容的云API推荐系统结果呈现和结合云API互补推荐算法的结果呈现,您认为在推荐结果页面中,提供互补云API对您在软件开发过程中是否有帮助?	有帮助	93.4%
		没有帮助	6.6%
6	目前,我们设计开发了一款云API互补推荐系统ReAPI,演示视频抖音链接:互补推荐系统dhxie.cn/4O TnTV,请您观看.如果有一个这样的在线云API推荐系统,您在实际应用软件开发时需要调用一些云API,是否会考虑使用?	意向评分(0-10)分	8.1

线方法方面,我们采用了启发式推荐方法(Random,Popular-N和AP)和深度学习方法(FM,AFM,DeepFM,NAFM和DynamicRec)设计实验来验证其求解高阶互补云API推荐问题的性能.对于这些基线方法,所有的模型和算法都是参考权威出版物设计实现的.因此,对比基线方法实现中的错误威胁是最小的.

(2)外部有效性威胁.外部有效性威胁与互补云API推荐应用是否与现有主流云API推荐系统冲突有关.高阶互补云API推荐问题提出的理念源自基于云API服务混搭的应用软件开发中,开发者对互补云API持续集成的客观需要.图13给出了互补云API推荐系统驱动的服务化软件开发过程.

如图13所示,开发者通常在应用开发前,尚未

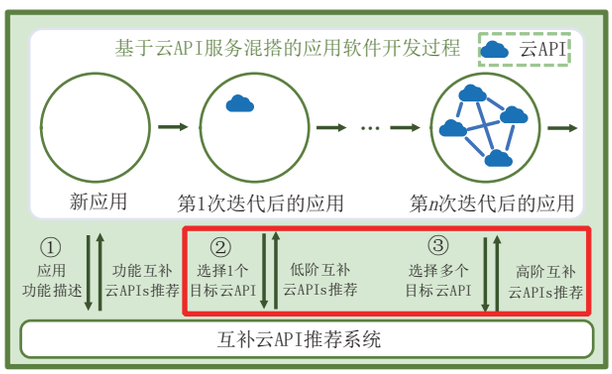

图13 互补云API推荐系统驱动的服务化软件开发

调用任何云API,但具有应用的功能类别等描述信息,故互补云API推荐系统可基于应用的功能描述,为开发者推荐功能互补的云API[27].

我们注意到,基于云API服务混搭的应用软件开发本身是一个对云API持续迭代集成的过程,这就要求推荐系统向开发者提供经常一起调用的云API,以满足开发者的联合兴趣.例如,在图14所示互补推荐应用比较中,传统内容驱动的云API推荐方法只提供功能单一的云API推荐结果(见图14(a)),而开发者在浏览云API的同时应用云API互补推荐算法后,推荐平台可以在页面右侧为开发者提供多元互补的云API列表(见图14b).这一功能与在淘宝,京东等主流电商平台已得到广泛应用的提供购买了“A物品”的买家同时还购买了“B、C和D等物品”的设计思路一致.由此可见,互补云API推荐与现有主流推荐系统不冲突,是对现有基于内容驱动、服务质量感知和用户偏好的云API推荐系统的有益补充,可有效提升用户对推荐平台的用户体验.

6 相关工作

6.1 云API推荐

现有云API推荐方法大致上可以分为三类:内容驱动的云API推荐、服务质量感知的云API推荐和基于偏好的云API推荐.

（a）传统内容驱动的云API推荐（无互补推荐算法应用）

（b）云API浏览过程中的互补推荐（有互补推荐算法应用）

图14　云API浏览过程中的互补推荐应用比较

（1）内容驱动的云API推荐. Thung等[7]建立了应用与云API关联模型，通过计算应用配置与云API需求描述信息之间的语义相似性进行推荐. Bianchini等[8]提出面向开发者的云API统一描述模型，实现了基于类别、标签和数据传输协议等条件检索的云API排序与推荐. Liu等[9]建立面向云API的句子描述模型，实现了基于自然语言理解的云API查询与推荐. Wang等[10]提出了面向开发者自然语言需求理解驱动的云API推荐. 上述方法主要利用应用配置、功能类别和文本描述等信息与云API检索关键词的相关性进行推荐. 这类方法虽然可以快速根据用户的显式需求给出高相关性的云API，但难以主动感知用户潜在的兴趣偏好，不能支持个性化云API推荐.

（2）服务质量感知云API推荐. Wu等[11]聚焦非功能侧需求，协同相似邻居预测云API服务质量，为用户推荐高质量云API. 针对稀疏性问题，Yin等[12]采用卷积神经网络学习邻居隐特征表示，并采用矩阵分解技术进行协同预测. Zhang等[13]提出地理位置感知的协同推荐方法. Chen[14]基于上下文感知和特征自动交互实现了端到端的云API服务质量预测. 尽管服务质量感知云API推荐研究在非功能侧感知方面取得了长足发展，但真实世界中用户侧的云API的服务质量信息通常不完整且难以获得，这使得服务质量感知的云API推荐方法并不可靠，甚至在许多情况下不适用. 此外，由于服务质量感知的云API推荐方法没有考虑云API与目标应用的功能相关性，对服务质量的关注常常作为其他推荐方法补充.

（3）基于偏好的云API推荐. Rahman等[15]设计正则项刻画云API的历史调用关系，利用矩阵分解模型预测用户对云API的偏好. Yao等[16]采用相似的正则化策略进行偏好预测并依据预测的偏好排序生成多个云API. Yin等[17]挖掘用户间、云API间相似性关系作为约束，并建立联合矩阵分解模型学习用户和云API的潜在特征表示. 进一步，Ma等[18]提出了面向多重交互的云API推荐方法，将应用于云API间的内容交互和邻居交互关系嵌入到深度神经网络预测用户的偏好. 上述方法主要利用用户和云API历史交互信息，通过建立邻居正则项约束，或者利用神经网络通过特征交互技术来实现偏好预测. 但用户和云API的特征表示仍然比较单一，未能有效挖掘用户多元兴趣偏好和云API多元特征表示.

已有云API推荐方法研究聚焦功能侧的替补云API推荐、非功能侧高质量云API推荐和满足用户偏好的个性化云API推荐，未能有效满足实际面向服务软件开发场景中开发者对多元互补推荐云API推荐的客观需要. Chen等在文献[27]指出基于应用功能描述信息驱动的互补云API推荐方法可有效提升推荐结果的多样性. 不同于已有工作，本文针对高阶互补云API这一问题，建立互补关系概率逻辑推理网络，实现了非对称性和高阶性可感知条件下的互补云API推荐.

6.2　互补推荐

在经济学中，互补商品定义为经常与另一种商品一起被消费的商品[28]. 如果两个商品之间的需求交叉弹性系数为负值，则一种商品价格的下降将增加其互补商品的需求. 因此，根据共同购买频率来识别两个商品是否为互补商品成为了互补性研究的第一经验法则.

近年来，研究者们在电商领域应用互补推荐的思想，给出互补且多元的结果，提升推荐准确性和用户体验. 在服饰互补推荐方面，Liu等[29]提出一种多自动编码器神经网络，对服装之间互补性进行建模的同时探索服饰间多种模式实现互补的服装匹配. 进一步，Liu等[30]提出一种基于端到端神经网络框架，利用产品文本和图像信息进行服装配搭推荐.

Li等[31]提出一种属性感知时尚推荐器解决服饰推荐任务，提取服饰的视觉特征并利用这些特征评估服饰间互补性. 在物品互补推荐方面，Wu等[32]引入互补感知图卷积模块捕捉商品之间互补关系，通过注意力机制区分互补性在目标商品中的重要程度. Yan等[33]通过使用图注意网络和顺序行为转换器对产品关系和用户偏好建模，捕捉用户行为序列信号，实现个性化的互补产品推荐. Rai等[34]使用长短期记忆网络的卵生神经网络制作一个基于内容的推荐系统，用于挖掘互补商品.

当前，针对电商领域的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，互补云API正是开发者在Mashup开发过程中的重要选择. 然而我们注意到，已有大部分研究使用了服饰或物品的视觉特征进行互补关系建模，而云API不存在视觉特征，从而导致已有方法不适用于解决云API互补推荐问题. 此外，已有研究将互补推荐简化建模成一个低阶互补推荐问题[18]，即仅为一个查询目标进行互补推荐. 而服务化软件开发中多元化云API的客观需要决定了云API的互补推荐具有高阶性，故本文对将云API推荐定义成一个高阶互补推荐问题，使其更具一般性、更贴近客观需要，而这也增加了问题求解的难度.

7 总结与展望

通过对真实云API生态数据分析，发现了开发者在构建应用时会调用多个不同功能和具有潜在互补关系的云API，以此作为本文的研究动机，提出并定义了高阶互补云API推荐问题. 本文设计了一种概率逻辑推理网络来应对高阶互补云API推荐问题求解中的挑战，采用Beta概率嵌入对云API及其关系进行编码，设计由投影、取反和交并三个基本逻辑算子构成的互补关系逻辑推理网络，挖掘查询云API在共同调用关系感知和替补关系噪声消解约束下的互补云API表示. 在此基础上，采用注意力网络获得与云API查询集整体互补的云API基向量. 最后，计算候选云API与互补云API基向量之间的KL散度距离，据此排序生成互补云API推荐列表. 实验表明本文所提方法能够有效地解决云API高阶互补推荐问题，并且具有很好的互补推荐场景迁移效果，可以满足与不同阶数的互补推荐需要.

本文所提方法PLR4HCCR虽然可以实现高阶互补云API推荐，但是PLR4HCCR目前只考虑了云API单侧的互补需求，尚未考虑应用侧的互补需求. 而在实际中电子商务类应用软件经常会调用支付类云API服务，即云API推荐还需考虑应用侧的互补性. 因此，在今后的工作中我们将探究应用侧的互补关系刻画方法，设计兼顾云API侧高阶互补性和应用侧互补关系的云API推荐方法，将更有可能生成令人满意的结果. 此外，我还将设计新的互补逻辑推理网络，进一步消解替补关系噪声对互补关系推理的影响；获取其他开放云API平台如RapidAPI数据，进一步验证本文所提方法的泛化能力.

致 谢 衷心感谢所有评审专家和编辑的建议和帮助!

参 考 文 献

[1] Ma XX, Liu XZ, Xie B, et al. Software development methods: review and outlook. Journal of Software, 2019, 30(1): 3-21 (in Chinese)
(马晓星, 刘譞哲, 谢冰, 等. 软件开发方法发展回顾与展望. 软件学报, 2019, 30(1): 3-21)

[2] Bouguettaya A, Singh M, Huhns M, et al. A service computing manifesto: the next 10 years. Communications of the ACM, 2017, 60(4): 64-72

[3] Chen Z, Qi WC, He PF, et al. A survey for cloud application programming interface security: threats and protection. Journal of Electronics & Information Technology, 2023, 45(1): 371-382
(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. 电子与信息学报, 2023, 45(1): 371-382)

[4] Airui Consulting Co., Ltd. China artificial intelligence API economic white paper. IResearch Consulting Series Research Report, 2020, 180-236
(艾瑞咨询有限公司. 中国人工智能API经济白皮书. 艾瑞咨询系列研究报告, 2020, 180-236)

[5] Qi L, Lin W, Zhang X, et al. A correlation graph based approach for personalized and compatible Web APIs recommendation in mobile APP development. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(6): 5444-5457

[6] Chen Z, Li Y, Wang Y, et al. Knowledge graph enhanced Web API recommendation via neighbor information propagation for multi-service application development//Proceedings of 2022 IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing. Hangzhou, China, 2022:20-40

[7] Thung F, Oentaryo R, Lo D, et al. WebAPIRec:

recommending web APIs to software projects via personalized ranking. IEEE Transactions on Emerging Topics in Computational Intelligence, 2017, 1(3): 145-156

[8] Bianchini D, Antonellis V, Melchiori M. WISeR: a multi-dimensional framework for searching and ranking web APIs. ACM Transactions on the Web, 2017, 11(3): 1-32

[9] Liu L, Bahrami M, Park J. Web API search: discover web API and its endpoint with natural language queries//IEEE International Conference on Web Services. Beijing, China, 2020: 96-113

[10] Wang Y, Chen J, Huang Q, et al. Deep learning-based open API recommendation for Mashup development. Science China Information Sciences, 2023, 66(7): 172102

[11] Wu J, Chen L, Feng Y, et al. Predicting quality of service for selection by neighborhood-based collaborative filtering. IEEE Transactions on Systems Man Cybernetics-Systems, 2013, 43(2): 428-439

[12] Yin Y, Chen L, Xu Y, et al. QoS prediction for service recommendation with deep feature learning in edge computing environment. Mobile Networks and Applications, 2020, 25(2): 391-401

[13] Zhang Y, Yin C, Wu Q, et al. Location-aware deep collaborative filtering for service recommendation. IEEE Transactions on Systems, Man, and Cybernetics-Systems, 2021, 51(6): 3796-3807

[14] Chen Z, Pan Z, He P, et al. Context and auto-interaction are all you need: towards context embedding based QoS prediction via automatic feature interaction for high quality cloud API delivery. Future Generation Computer Systems, 2022, 128: 265-281

[15] Rahman M, Liu X. Integrated topic modeling and user interaction enhanced web API recommendation using regularized matrix factorization for mashup application development//Proceedings of 2020 IEEE International Conference on Services Computing. Beijing, China, 2020: 124-131

[16] Yao L, Wang X, Sheng Q, et al. Mashup recommendation by regularizing matrix factorization with API co-invocations. IEEE Transactions on Services Computing, 2021, 14(2): 502-515

[17] Yin Y, Huang Q, Gao H, et al. Personalized APIs recommendation with cognitive knowledge mining for industrial systems. IEEE Transactions on Industrial Informatics, 2021, 17(9): 6153-6161

[18] Ma Y, Geng X, Wang J. A deep neural network with multiplex interactions for cold-start service recommendation. IEEE Transactions on Engineering Management, 2020, 68(1):105-119

[19] He P, Qi W, Liu X, et al. Seizing the long tail: neural complementary recommendation for cloud API delivery// Proceedings of 2022 IEEE Ubiquitous Intelligence & Computing. Haikou, China, 2022: 1954-1961

[20] Ren H, Leskovec J. Beta embeddings for multi-hop logical reasoning in knowledge graphs. Advances in Neural Information Processing Systems, 2020, 33: 19716-19726

[21] He P, Qi W, Liu X, et al. Association rule guided Web API complementary function recommendation for mashup creation: an explainable perspective//Proceedings of 2022 Chinese Conference on Computer Supported Cooperative Work and Social Computing. Taiyuan, China, 2022: 73-83

[22] Rendle S, Factorization machines with LibFM. ACM Transactions on Intelligent Systems and Technology, 2012, 3: 1-22

[23] Cao Y, Liu J, Shi M, et al. Service recommendation based on attentional factorization machine//2019 IEEE International Conference on Services Computing. Milan, Italy, 2019: 189-196

[24] Deng L, Guo B, Zheng W. A service recommendation algorithm based on self-attention mechanism and DeepFM. International Journal of Web Services Research, 2023, 20(1): 1-18

[25] Kang G, Liu J, Xiao Y, et al. Neural and attentional factorization machine-based Web API recommendation for Mashup development. IEEE Transactions on Network and Service Management, 2021, 18(4): 4183-4196

[26] Tanjim M, Ayuubi H, Cottrell G. DynamicRec: a dynamic convolutional network for next item recommendation// Proceedings of the 29th ACM International Conference on Information & Knowledge Management. Virtual Event, Ireland, 2020: 2237-2240

[27] Chen Z, Chen W, Liu X, et al. Functional complementarity relationship enhanced cloud API recommendation method. Journal on Communications, 2023, 44(6): 125-137
(陈真,陈文辉,刘啸威等.功能互补关系增强的云API推荐方法.通信学报, 2023, 44(6): 125-137)

[28] Zhang M, BockstedT J. Complements and substitutes in product recommendations: the differential effects on consumers' willingness to pay//2016 Joint Workshop on Interfaces and Human Decision Making for Recommender Systems. Boston, USA, 2016: 36-43

[29] Liu J, Song X, Chen Z, et al. Neural fashion experts: I know how to make the complementary clothing matching. Neurocomputing, 2019, 359: 249-263

[30] Liu J, Song X, Nie L, et al. An end-to-end attention-based neural model for complementary clothing matching. ACM Transactions on Multimedia Computing, Communications, and Applications, 2019, 15(4): 1-16

[31] Li Y, Chen T, Huang Z. Attribute-aware explainable complementary clothing recommendation. World Wide Web, 2021, 24: 1885-1901

[32] Wu B, Zhong L, Li H, et al. Efficient complementary graph convolutional network without negative sampling for item recommendation. Knowledge-Based Systems, 2022, 256: 1-15

[33] Yan A, Dong C, Gao Y, et al. Personalized complementary product recommendation//WWW'22: Companion Proceedings of the Web Conference. Lyon, France, 2022: 146-151

[34] Rai R, Patel M, Varma P, et al. Complementary product recommendation using siamese neural network//Proceedings of 2023 International Conference on Communication System, Computing and IT Applications. Sydney, Australia, 2023: 121-125

CHEN Zhen, Ph. D., associate professor. His main research interests include service computing, recommendation systems, and software engineering.

XIE Deng-Hui, M. S. candidate. His main research interests include cloud API recommendation and complementary recommender systems.

WANG Xiao-Long, M. S. candidate. His main research interests include semantic analysis and cloud API complementary recommendation.

SUN Meng-Meng, Ph. D. candidate. Her main research interests include deep learning and cloud API complementary recommendation.

LIU Xiao-Wei, M. S. candidate. His main research interests include cloud API data mining and complementary recommender system.

SHEN Li-Min, Ph. D., professor. His main research interests include service computing, flexible software, collaborative computing, and information security.

Background

The history of software engineering over the past 50 years has demonstrated that reusing technology is a viable way to solve the software crisis, increase the efficiency of software production, and promote the engineering and industrialization of the software industry. To accommodate software reuse in a networked environment, service-oriented software development methods have been proposed, advocating software development through networked service mashups. As a new technology to enable service-oriented software development, cloud API has attracted many developers to access various cloud services and respond to the changing needs of users through cloud API, effectively reducing the complexity and development cost of software. However, the massive and continuous growth of cloud APIs makes it extremely difficult to quickly identify cloud APIs that meet user needs, which limits the healthy development of the API economy and the popularity of cloud API application.

Cloud API recommendation is an effective way to solve the above problems. In recent years, top international journals and conferences such as TSE and ICWS have published research reports on the latest achievements, including content based, QoS aware and personalized cloud API recommendation methods, but little work has been done to understand and characterize the complementary needs between service oriented software development and cloud API recommendation. For example, in service oriented software development, considering the high-order complementarity between the recommended cloud API and the multiple cloud APIs that have been invoked will be more likely to yield satisfactory results.

To address the complementary recommendation problem of cloud API in service oriented software development, it remains to establish a probabilistic logic reasoning approach based high-order complementary cloud API recommendation method on the basis of our existing research works, which is of great importance for scientific research and industrial application of service oriented software development. This work was supported by the National Natural Science Foundation of China (No. 62102348, 62276226), the Natural Science Foundation of Hebei Province (No. F2022203012), Science and Technology Program of Hebei (No. 236Z0103G), Hebei Provincial Department of Higher Education Science and Technology Program (No. QN2020183), Hebei Provincial Innovation Capability Improvement Program (No. 22567626H).