基于知识图谱的 GitHub 层次化学习和检索服务

江惠珍[1]，孙艳春[1*]，黄罡[1,2]

（1. 北京大学计算机学院高可信软件技术教育部重点实验室，北京 100871；
2. 数据空间技术与系统全国重点实验室，北京 100091）

摘　要：GitHub 是全球最大的在线代码托管平台，为软件开发学习者提供了丰富的学习资源，然而面对丰富而繁杂的 GitHub 内容，软件开发初学者在使用 GitHub 的搜索功能搜索所需的学习资源时，由于需求不明确或缺乏相关知识和经验，常会遇到难以构建合适的搜索文本进行有效搜索的问题。针对该问题，利用 GitHub 主题潜在的层次结构，结合 Wikipedia 软件开发领域知识，设计 GitHub 软件开发领域知识图谱，在此基础上提出一种基于知识图谱的 GitHub 层次化学习和检索服务。通过对比实验和调查问卷的方式验证了提出的层次化学习和检索服务的可行性和有效性。

关键词：GitHub 平台；维基百科；知识图谱；层次化学习；检索服务

中图分类号：TP311.5　　**文献标志码**：A　　**DOI**：10.19678/j.issn.1000-3428.0068655

GitHub Hierarchical Learning and Retrieval Service Based on Knowledge Graphs

JIANG Huizhen[1], SUN Yanchun[1*], HUANG Gang[1,2]

(1. Key Laboratory of High Confidence Software Technologies, Ministry of Education, School of Computer Science, Peking University, Beijing 100871, China;
2. National Key Laboratory of Data Space Technology and System, Beijing 100091, China)

【Abstract】 As the largest online code hosting platform in the world, GitHub provides rich learning resources for software development learners. However, faced with such rich and complex GitHub content, beginners in software development often encounter difficulties in forming suitable search texts to search effectively when using the search function of GitHub to search for the learning resources they need because of their unclear requirements or lack of relevant knowledge and experience. To address this problem, this study designs a GitHub software development knowledge graph combining the potential hierarchical structure of GitHub topics with the domain knowledge of software development in Wikipedia and proposes a GitHub hierarchical learning and retrieval service based on the knowledge graphs. The feasibility and effectiveness of the proposed hierarchical learning and retrieval service are verified through comparative experiments and questionnaires.

【Key words】 GitHub platform; Wikipedia; knowledge graph; hierarchical learning; retrieval service

0　引言

GitHub 是世界上最大的在线代码托管平台，对软件开发学习者而言是一个学习资源的宝库，然而 GitHub 上的学习资源虽多，但对于软件开发初学者而言存在过于繁杂、难于检索的问题。在 GitHub 上检索仓库实际上是一个提出需求的过程。软件开发初学者欠缺软件开发方面的经验和背景知识，对他们而言明确自己的需求是一件具有门槛的事情。当搜索者的需求不明确时，往往只能给出一个模糊、范围较大的搜索文本，给搜索造成困难，同时 GitHub 的搜索是基于搜索文本与仓库标题、描述的文本匹配，并没有针对用户需求不明确的问题进行优化。因此，软件开发初学者使用 GitHub 存在一定的困难。

为了解决以上问题，本文构建 GitHub 仓库所涉及的软件开发相关概念之间内在的层次关系，基于此对 GitHub 上的仓库进行层次分类，从而对仓库展开层次化检索。相比于直接进行文本搜索，层次化检索每次提示用户下一步检索的仓库领域选项，从而帮助挖掘出用户内在的检索需求。此外，所构建的 GitHub 软件开发相关概念的层次关系也展示了软件开发知识的结构，借助该层次关系，用户可以系统性地探索学习软件开发知识，并结合 GitHub 仓库，在学习中做到理论结合实际。

本文使用 GitHub 从 2007 年 10 月到 2022 年

收稿日期：2023-10-20　　**修回日期**：2023-12-25

基金项目：北京高等学校卓越青年科学家计划项目（BJJWZYJH01201910001004）。

通信作者 E-mail：* sunyc@pku.edu.cn

10 月的公开仓库数据，不涉及任何用户隐私，对 GitHub 仓库所含有的 GitHub 主题(topic)进行层次构建。GitHub topic 是 GitHub 于 2017 年推出的功能，用户可以自主为仓库添加 topic，相当于为仓库打上标签，从而便于仓库的分类与检索。此外，为了构建一个完整的软件开发领域知识层次，使用维基百科(Wikipedia)软件开发领域下的知识条目。Wikipedia 提供了知识条目之间的层次关系，常用于各领域的概念层次的构建。基于 Wikipedia 和 GitHub 数据，建立 GitHub 软件开发领域知识图谱，设计并实现了基于知识图谱的 GitHub 层次化学习和检索服务。通过该服务，用户可以层次化地浏览学习软件开发领域的知识点和相关的 GitHub 仓库，也可以对 GitHub 仓库进行层次化检索。

本文的主要贡献如下：1)利用层次聚类构建 GitHub topic 的层级结构，解决了 GitHub topic 无序、难于浏览和检索的问题，使其可用于仓库的层次化检索；2)应用来自 Transformer 的双向编码器表示(BERT)对 GitHub topic 进行了层次分类，有效地为无 topic 的仓库预测 topic 类别；3)通过融合 Wikipedia 知识目录和 GitHub topic 层级结构构造软件开发领域知识图谱，在此基础上实现 GitHub 层次化学习和检索服务，解决了软件开发初学者由于需求不明确难以展开 GitHub 检索的问题。

1 相关工作

1.1 学习领域知识图谱的构建与应用

知识图谱是一种组织知识的方式，以实体-关系-实体三元组来描述真实世界中的各种实体和概念以及它们之间的关系。知识图谱最初应用在搜索领域，随着知识图谱技术的发展，在问答、推荐、大数据分析等领域也得到广泛的应用。在学习领域，知识图谱也具有很高的应用价值。WANG 等[1]提出一种从教科书中提取概念图的框架，其中概念图是一种特殊类型的知识图谱，在教育领域有着广泛应用。CHEN 等[2]提出了一种教育领域的知识图谱构建方法，其中使用神经序列标注算法从教育数据中抽取概念，从学习数据中识别有意义的关系。DANG 等[3]在慕课网上为慕课课程构建了知识图谱，将从慕课课程的文本中抽取的 24 188 个概念连接到相应的 Wikipedia 条目，从而为慕课课程提供了知识的规范化表示和更精确的描述。

Wikipedia 是一个综合的协作型的互联网百科全书，超过 100 万个条目，以分类索引的方式构建层级目录，便于用户检索。针对以 Wikipedia 为代表的知识社区，学者们进行了相关的知识图谱构建与应用研究。LIANG 等[4]提出一种概念之间先序关系的度量方法，基于 Wikipedia 提取出概念之间的先序关系，构建知识图谱。HUANG 等[5]提出一个利用学生问题日志、教科书、Wikipedia 等多源数据构建概念图的框架。YIN 等[6]结合 GitHub 仓库的多源信息，构建知识图谱，在此基础上结合知识图谱嵌入模型和路径推荐算法，为开发人员推荐开源项目的学习路径。

综合上述分析，知识图谱对于知识学习起到了积极的作用，以 Wikipedia 为代表的知识社区也被应用于知识图谱构建，惠及不同领域的学习者。但是，单纯利用 Wikipedia 学习某特定领域的知识对于学习者而言也有不便之处。以软件开发知识与技能的学习为例，Wikipedia 软件开发领域的知识图谱涵盖了软件开发领域各个方面的知识，但 Wikipedia 是一个综合性的百科全书，对于软件开发领域细分知识的收录不足，若要学习和掌握一个具体的知识点，则并不是仅仅通过阅读 Wikipedia 上的一个条目就可以达成的，其下许多细分的知识在 Wikipedia 上并没有提供，阻碍了初学者通过 Wikipedia 进行系统性的知识学习，这也是许多相关研究基于 Wikipedia 的概念与知识图谱，结合特定学习领域的知识进行扩充的主要原因。

1.2 开源软件仓库的分类、检索与推荐

GitHub 是世界上最大的开源软件社区，对于软件开发人员以及软件开发学习者具有重要的价值。然而，GitHub 仓库的巨大量级也给仓库资源的快速准确搜索和有效推荐提出了挑战。

随着 GitHub 仓库数量和规模的快速增加，一些学者对 GitHub 上的开源软件项目进行分类，以便于浏览、推荐和搜索。BORGES 等[7]将 GitHub 仓库根据应用领域分为 6 类，并给出一个人工标注的数据集。ZANARTU 等[8]基于分类标准和标注数据[7]设计一个自动分类器，使用自动化机器学习框架对 GitHub 仓库进行分类，并发现在用于分类的仓库特征中仓库的文本特征起到了重要作用。SOLL 等[9]提出一种基于集成学习的 GitHub 仓库分类方法，同样对仓库的应用领域进行分类，使用多个弱分类器分别根据仓库的文件类型、Readme 文件、语言、仓库结构等信息进行分类，并结合弱分类器的结果得出最终的分类结果。SHARMA 等[10]使用潜在狄利克雷分配(LDA)-遗传算法(GA)对 GitHub 仓库的描述文本进行主题建模，并人工为各个主题命名。ZHANG 等[11]提出一种弱监督的

GitHub 仓库层级分类方法，仅须提供标签层次和关键词，利用异构信息网络和主题建模，将仓库分类到给定的标签层级。

综合上述分析，随着自然语言处理技术的发展，从仓库文档中提取文本语义信息以进行分类逐渐成为主流。大部分仓库分类工作都需要事先给定分类类别，这难以适应如今开源软件社区的快速发展和开源软件项目数量和类别的迅速增长，因此自动生成仓库分类类别是研究的方向。

在开源软件仓库检索方面，学者们进行了一系列研究。ASYROFI 等[12]使用类型解析，帮助用户在 GitHub 仓库中精确搜索特定 API 的用法示例。CHEN 等[13]结合了流行的编程问答社区 Stack Overflow，基于深度学习模型，使用 Stack Overflow 中部分提及 GitHub 仓库作为问题回答的问题帖，学习相关 GitHub 仓库的表示，从而抽取仓库的语义特征，使得针对用户提问以检索相关仓库成了可能。LIU 等[14]提出一种与用户需求相关的多维度的开源软件度量模型，并实现了一种基于模糊综合评价法的排序算法，在搜索时允许用户提出对目标仓库的多种指标的个性化需求，从而使得搜索结果能够更加符合用户的需求。WU 等[15]基于命名实体识别和异构信息网络，为软件开发人员在 GitHub 等平台提供基于功能语义的开源仓库搜索服务。针对 GitHub 仓库杂乱无章且难于检索的问题，CAI 等[16]提出一种基于图谱的为 GitHub 仓库分配标签的方法，利用标签来扩充用户输入的搜索文本，并基于标签设计一种仓库排序算法，从而为用户提供最相关的搜索结果。

关于 GitHub 仓库推荐的相关研究主要基于仓库相似度和用户-仓库网络进行推荐。MATEK 等[17]利用 GitHub 用户对仓库的贡献关系建图进行网络分析，使用链接预测构建仓库推荐系统。GUENDOUZ 等[18]使用协同过滤方法，并将用户行为建模为用户-仓库矩阵，从而利用用户和仓库相似性计算等方法进行仓库推荐。KOSKELA 等[19]提出一种混合的开源软件推荐算法，将新仓库与用户过去感兴趣的仓库进行比较，通过计算 topic、编程语言和仓库 Readme 文件等方面的相似度为用户进行推荐。YANG 等[20]基于深度矩阵分解，利用深度神经网络从用户-仓库矩阵中学习用户和仓库的低维表示，捕获用户对每个仓库的潜在偏好，从而为用户推荐个性化的 GitHub 仓库。KIM 等[21]提出一个大型的 GitHub 交互数据集，在该数据集上比较了各种流行的基于深度学习的推荐算法，验证了通过建模用户以往的行为可以有效地为用户推荐仓库。ZHOU 等[22]利用 BERT 对 GitHub 仓库进行 topic 预测，结合用户的开发偏好，为软件开发人员推荐 GitHub 的趋势开源软件项目。

通过分析目前的相关工作可知，在 GitHub 上进行检索往往需要用户具有明确的搜索意图，而仓库推荐往往需要利用用户的历史行为数据，存在冷启动问题，且难以实时匹配用户对仓库的需求。目前的相关研究对于软件开发初学者搜索需求不明确及软件开发经验和背景知识不足且难以用合适的关键词表达搜索意图这一问题考虑较少。

2 GitHub 层次化学习和检索服务架构

本文提出的 GitHub 层次化学习和检索服务系统架构如图 1 所示。

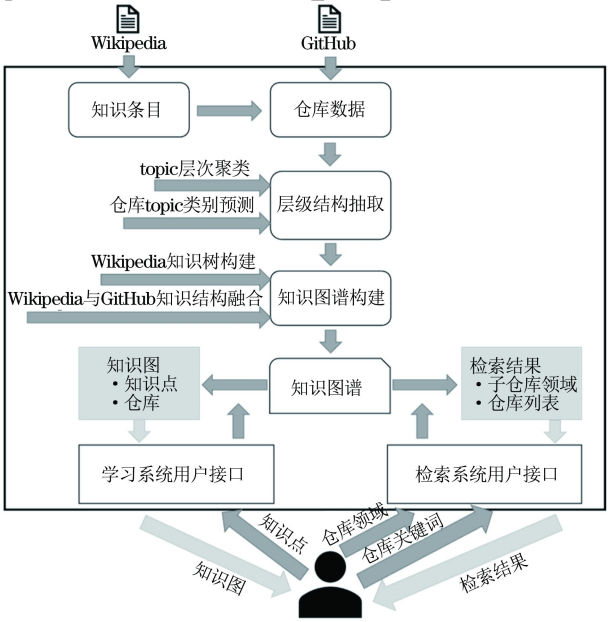

图 1 GitHub 层次化学习和检索服务系统架构

Fig. 1 System architecture of the GitHub hierarchical learning and retrieval service

2.1 GitHub topic 层级结构构建

2.1.1 GitHub topic 层次聚类

为构建软件开发领域知识图谱，获取 Wikipedia 软件开发领域下的各个知识条目和相关联的 2 281 个 GitHub 仓库，以它们为基础进行扩展：对于每个仓库，随机选取 1 名对该仓库标星(star)的用户作为该仓库的标星用户代表，爬取其标星仓库列表，由此获得超过 75 万个 GitHub 仓库。星标数是衡量仓库质量的一个重要指标[23]，为了提高仓库质量，筛选出这些仓库中星标数不少于 150 的超过 18 万个仓库，获取其中每个仓库的星标数、仓库描述、Readme 文件、topic 等信息，并将每个仓库的仓库描述与 Readme 文件中的文本内容连接，作为该仓库的描述文本。

在划定 GitHub 仓库的研究范围后，基于其中 94 617 个含有 topic 的仓库所含 topic 之间的共现关系，利用 Leiden 社区发现算法对 topic 进行层次聚类[24]，算法流程如图 2 所示，其中，topic t_a 和 t_b 的共现次数定义为同时含有 t_a 和 t_b 的仓库数量。

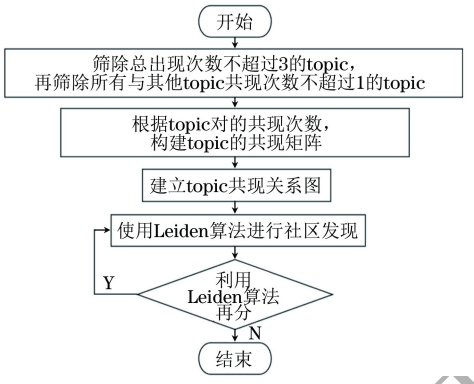

图 2 topic 层次聚类算法流程

Fig. 2 Procedure of topic hierarchical clustering algorithm

将一个 topic 社区称为一个 topic 簇。度中心性常用于衡量网络中节点的重要性[25]，在后文的讨论中，称每个 topic 簇中度中心性最大的 topic 为该 topic 簇的“标志性 topic”。对于一个 GitHub 仓库，若其含有某个 topic 簇内的一个 topic，则称该仓库与该 topic 簇具有关联关系。

2.1.2 GitHub 仓库的 topic 类别预测

由于 GitHub 仓库的 topic 由仓库作者自主选择标注，因此有一大部分仓库不含 topic，需要通过 topic 类别预测补上隐含的 topic，从而将它们连接到第 2.1.1 节中得到的 topic 簇，建立知识图谱。然而，topic 类别过多，给类别预测带来困难，常见的解决方案是利用类别之间的层次关系进行层次分类[26-28]。本文使用层次分类方法，结合第 2.1.1 节中构建的 topic 层次聚类，对于每个非叶子节点的 topic 簇构建一个基于 BERT 预训练模型的多标签分类器，从而预测仓库所连接的底层 topic 簇。

由于第 2.1.1 节中所得的底层 topic 簇数量过大，因此本文基于 GitHub 提供的特征主题(featured topic)进行仓库 topic 类别预测，包含 GitHub 官方提供的最为流行和最具有代表性的 180 个 topic，剔除其中在 GitHub 上出现次数不足 100 的 topic，得到 175 个 featured topic，后文所提到的 featured topic 均指这 175 个 featured topic。经统计，在所涉及的 94 617 个含有 topic 的仓库中，有 75.7%的仓库含有 featured topic，有 81.0%的仓库关联到含有 featured topic 的底层 topic 簇。此外，在第 2.1.1 节的层次聚类结果中，每个仓库从关联的底层 topic 簇到根节点的路径中，含 featured topic 节点的路径长度所占比例平均为 63.5%。因此，认为仓库含有的 featured topic 可以在一定程度上代表仓库含有的所有 topic，反映出仓库在 topic 层次聚类结果中关联到的底层 topic 簇，可以用于进行仓库 topic 类别的预测。在 159 个含 featured topic 的底层 topic 簇中为不含 topic 的仓库寻找相关联的 topic 簇。为了促进数据的平衡性，为每个 featured topic 采样 100 个含有该 topic 的仓库，并按 9∶1的比例划分为训练集和测试集。

第 2.1.1 节中的 topic 层次聚类最终得到一棵 topic 簇树，抽取出其中含有 featured topic 的子树部分，保留每个 topic 簇中的 featured topic，从而得到 featured topic 的层次聚类，作为仓库 topic 类别预测的类别层次。利用该类别层次，将 1 个含有 159 个标签的多标签分类器拆分为 74 个标签数量较少的多标签分类器。对于类别层次中每一个非叶子节点的 topic 簇，假设当前 topic 簇被分为 T_1，T_2，…，T_k 这 k 个子 topic 簇，选取训练集内含有簇中 topic 的仓库作为训练样本，在其上训练一个 k 标签的多标签分类器，输入样本仓库的描述文本，输出该仓库与各子 topic 簇存在关联的概率。每个多标签分类器的结构如图 3 所示，仓库描述文本通过 BERT 编码层获得包含语义特征的文档向量，再通过全连接层和 Sigmoid 层得到输出。

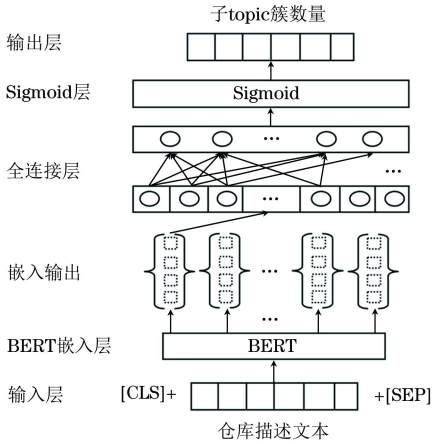

图 3 单个仓库 topic 类别预测分类器模型结构

Fig. 3 Model structure of single repository topic category prediction classifier

在训练完成后，对于每个不含 topic 的仓库，将仓库的描述文本输入类别层次中每个 topic 簇的分类器，得到仓库与该 topic 簇相关联时分别与各个子 topic 簇相关联的概率。对于任意 topic 簇 C，设仓库与之相关联的概率为 p_C。在综合结果时，已知对于类别层次中根节点 C_R，$p_{C_R}=1$，对于任意 topic 簇节点 C_N，若 C_N 有 k 个子 topic 簇 C_{N_1}，

$C_{N_2},\cdots,C_{N_k}$，分类器对于 C_{N_i} 输出的概率为 p_i，则 $p_{C_{N_i}}=p_{C_N}\times p_i(i=1,2,\cdots,k)$，递归可得仓库与每个 topic 簇相关联的概率。取底层 topic 簇中与仓库关联概率最大的前 3 个 topic 簇，将仓库与它们相关联。

2.2 知识图谱构建

在第 2.1.1 节中，从 Wikipedia 软件开发领域页面出发，递归遍历其下的各个子分类和子页面，获取软件开发领域的超过 27 000 个知识条目，根据条目间的上下级关系连接，组织为 Wikipedia 软件开发领域层级知识树。

在构建 Wikipedia 软件开发领域层级知识树后，需要与 GitHub 层级结构进行连接，从而对 Wikipedia 软件开发领域层级知识树进行扩展，构成完整的 GitHub 软件开发领域知识图谱。

Wikipedia 软件开发领域层级知识树与 GitHub 层级结构的连接方法如下：将层级知识树的叶子节点与本文所涉及的 GitHub topic 进行文本匹配，假如一个叶子节点 E 在 GitHub topic 中匹配 topic T，说明该叶子节点的知识在 GitHub 上有继续延伸的可能，在 topic 层次聚类结果中寻找以 topic T 为标志性 topic 的 topic 簇。假如 topic T 是 topic 簇 $C_1,C_2,\cdots,C_n$ 的标志性 topic，$C_1,C_2,\cdots,C_n$ 之间应当存在层次关系。选择在 $C_1,C_2,\cdots,C_n$ 中层次最高的 topic 簇 C_k，将 E 与 C_k 连接。

通过上述连接，仅有 9 个 topic 簇无法连接到 Wikipedia 知识树，均为 topic 层次聚类中第 1 次对 topic 运用社区发现所获得的 topic 社区，且规模小，无子 topic 簇。将它们连接到 Wikipedia 知识树的根节点，从而使所有 topic 簇都与 Wikipedia 软件开发领域层级知识树建立联系。同时，递归删除 Wikipedia 知识树中无法连接到 GitHub topic 簇的叶子节点，直到 Wikipedia 软件开发领域层级知识树中的每个知识条目都可以通过在图中向下延伸到达 topic 簇。至此，本文构建了完整的 GitHub 软件开发领域知识图谱，结构如表 1 和表 2 所示。

表 1 GitHub 软件开发领域知识图谱中的实体类型与描述

Table 1 Entity types and descriptions in the knowledge graphs of GitHub software development domain

实体类型	实体类型描述
Wikipedia-entry	Wikipedia 知识条目实体，代表一个软件开发领域的 Wikipedia 知识条目
Topic-cluster	topic 簇实体，代表第 2.1.1 节中 topic 层次聚类所得的一个 topic 簇
Repository	GitHub 仓库实体，代表一个 GitHub 仓库

表 2 GitHub 软件开发领域知识图谱中的关系

Table 2 Relationships in the knowledge graphs of GitHub software development domain

关系	头实体	尾实体
has sub Wikipedia-entry	Wikipedia-entry	Wikipedia-entry
match	Wikipedia-entry	Topic-cluster
has sub Topic-cluster	Topic-cluster	Topic-cluster
related	Repository	Topic-cluster

2.3 GitHub 层次化学习和检索服务

软件开发初学者在需求不明确时难以在 GitHub 上开展有效的学习和检索。为此，实现了 GitHub 层次化学习和检索服务（https://github.com/JHZ-D/GitHub-HierSearch），该服务有 2 种使用方式：第 1 种方式是查看软件开发领域知识点构成的层级知识图谱，从而层次化地对 GitHub 平台所包含的知识点进行学习；第 2 种方式是对 GitHub 进行层次化检索，每次根据知识图谱提供下一步仓库范围选项，用户选择与希望检索的主题最接近的选项，从而逐步明确检索需求。

图 4 为服务的层次化学习系统界面，为用户提供与知识点相关联的层级知识树和知识点介绍。图 5 为服务的层次化检索系统界面，用户可以逐层选择仓库检索范围选项，输入关键词进行检索，服务将返回符合条件的仓库，向用户展示仓库名、仓库描述和仓库链接。

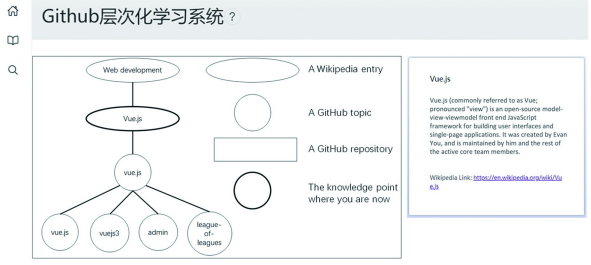

图 4 GitHub 层次化学习系统界面

Fig. 4 GitHub hierarchical learning system interface

图 5 GitHub 层次化检索系统界面

Fig. 5 GitHub hierarchical retrieval system interface

3 GitHub 层次化学习和检索服务验证

3.1 topic 类别预测算法验证

topic 类别预测算法利用 BERT 和层次分类实现了 159 个标签的多标签分类，为了验证 BERT 和层次分类的效果，分别与 2 个基线(Baseline)模型进行对比：基于长短时记忆(LSTM)网络的含 159 个标签的分类模型和未使用层次分类的基于 BERT 的含 159 个标签的分类模型，使用 LSTM 层和 BERT 编码层对仓库描述文本进行编码，再通过全连接层和 Sigmoid 层输出仓库属于每个分类的概率。

采用在该领域内广泛使用的预测 top-k 的 topic 的成功率、精确率和召回率作为评价指标[29]。对于每个 k 值，模型将所有分类中概率最大的 k 个分类作为仓库的分类；对于每个仓库的每个预测类别，若其在仓库的真实 topic 类别中，则记为一次真阳性，否则记为一次假阳性；若仓库的真实 topic 类别中的一个类别不在预测类别中，则记为一次假阴性。模型通过在测试集上的预测，统计其中真阳性数量记作 N_{TP}，假阳性数量记作 N_{FP}，假阴性数量记作 N_{FN}，则模型的成功率、精确率和召回率分别计算如下：

成功率是测试集 R 中至少有一个 topic 类别预测正确的仓库数占所有仓库数的比例，计算公式如下：

$$R_{\text{Success\_rate}}=\frac{\text{count}_{r\subseteq R}(N_{TP}>0)}{|R|} \tag{1}$$

精确率是正确预测的类别数占预测数的比例，计算公式如下：

$$P=\frac{N_{TP}}{N_{TP}+N_{FP}} \tag{2}$$

召回率是正确预测的类别数占真实数的比例，计算公式如下：

$$R=\frac{N_{TP}}{N_{TP}+N_{FN}} \tag{3}$$

当 k 分别取 2、5、8 时，对比实验结果如表 3 所示，其中，LSTM 代表基于 LSTM 的分类模型，BERT-159 代表未使用层次分类的基于 BERT 的分类模型，BERT 代表本文使用的基于 BERT 和层次分类的分类模型，最优指标值用加粗字体标示。

由表 3 中数据可以看出，基于 BERT 的分类模型优于基于 LSTM 的分类模型，而本文使用的基于 BERT 和层次分类的分类模型又明显优于未使用层次分类的基于 BERT 的分类模型，由此可知本文使用 BERT 和层次分类进行 topic 类别预测的合理性和有效性。

表 3 预测 k 个 topic 时的 3 种模型比较

Table 3 Comparison of three models for predicting k topics

k	评价指标	LSTM	BERT-159	BERT
2	成功率	0.294 2	0.601 5	**0.907 4**
	精确率	0.058 0	0.217 4	**0.693 3**
	召回率	0.109 1	0.239 1	**0.454 8**
5	成功率	0.479 2	0.713 7	**0.957 8**
	精确率	0.077 8	0.156 8	**0.466 7**
	召回率	0.201 6	0.353 0	**0.699 1**
8	成功率	0.577 9	0.758 8	**0.973 8**
	精确率	0.064 2	0.144 4	**0.350 0**
	召回率	0.268 4	0.417 0	**0.790 0**

为了验证 top-k 中 $k=3$ 时性能最优，统计了 k 取 1～5 时的成功率、精确率、召回率和 F1 值，其中 F1 值的定义如下：

$$F_1=2\times\frac{P\times R}{P+R} \tag{4}$$

由表 4 可知，随着 k 的增大，成功率和召回率升高，精确率降低，F1 值先升高后降低，$k=3$ 时 F1 值最高，因此每个仓库与预测概率最高的 3 个 topic 类别相关联。

表 4 k 取 1～5 时的成功率、精确率、召回率和 F1 值

Table 4 Success rate, precision, recall and F1 value when k takes 1 to 5

评价指标	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
成功率	0.816 5	0.907 4	0.934 5	0.950 8	**0.957 8**
精确率	**0.778 0**	0.693 3	0.596 4	0.526 4	0.466 7
召回率	0.266 2	0.454 8	0.571 2	0.649 0	**0.699 1**
F1 值	0.373 3	0.528 4	**0.567 9**	0.566 1	0.544 0

3.2 Wikipedia 与 GitHub 链接准确性验证

使用文本匹配对 Wikipedia 知识条目和 GitHub topic 进行链接。为了验证链接的准确性，利用 GitHub Explore 仓库(https://github.com/github/explore)中官方公布的常用 topic 信息进行验证。该仓库有 654 个 topic 在第 2.1.1 节所划定的 topic 研究范围内，在 topic 层次聚类中所得的 topic 簇中有 21.9% 含有该仓库中的 topic，有 15.9%以该仓库中的 topic 为标志性 topic。这 654 个 topic 大部分有对应的 Wikipedia 链接，链接到 Wikipedia 上 514 个不同的知识条目，这 514 个 Wikipedia 知识条目构成集合 W。根据该仓库中的 topic 信息，W 中每个知识条目 w 链接到 topic t_w，以此为标准答案；使用文本匹配方法将 w 链接到 topic y_w，计算匹配方法的准确率，计算公式如下：

$$A=\frac{\text{count}_{w\in W}y_w=t_w}{|W|} \tag{5}$$

为了验证第 2.2 节中采用的 Wikipedia 知识条目与 GitHub topic 的链接方法的准确性，将其准确率与随机匹配方法进行对比，随机匹配方法具体做法如下：集合 W 中的 Wikipedia 知识条目对应的 GitHub topic 构成集合 G，对于集合 W 中的每个知识条目 w，随机抽取集合 G 中的一个 topic 进行匹配，记为 r_w，最后计算准确率。准确率计算公式如下：

$$A=\frac{\text{count}_{w\in W}y_w=r_w}{|W|} \tag{6}$$

将随机匹配方法重复 10 次，取准确率的平均值作为随机匹配方法的准确率。

本文方法的准确率为 0.819 1，而随机匹配方法的准确率为 0.002 2，由此可知本文采用的 Wikipedia 知识条目与 GitHub topic 的链接方法具有较高的准确性，且明显优于随机匹配方法。

3.3 topic 层次分类效果验证

在本文的数据范围内，GitHub Explore 仓库中共有 469 对 Wikipedia 知识条目-GitHub topic 对，利用该信息对 topic 层次分类效果进行验证。

经过本文的层次分类，GitHub topic 形成了树状的层级结构。在 topic 簇树中通过与其距离较近的 topic 来验证 topic 层次分类效果。使用 topic 链接到的 Wikipedia 知识条目表示其表达的实际概念，2 个 Wikipedia 知识条目的实际概念越接近，知识条目在 Wikipedia 知识网络中也越接近。为此，需要验证：若 topic 在 topic 簇树中距离越近，则它们所链接的 Wikipedia 知识条目在 Wikipedia 知识网络中也越接近。

概念 a 和 b 分别对应 topic t_a 和 t_b，与 Wikipedia 知识条目 w_a 和 w_b，定义 topic 反向距离 $d_{\text{rt}}(a,b)$ 为 t_a 与 t_b 在 topic 簇树中的最近公共祖先的深度（根节点深度为 0），定义 Wikipedia 知识条目距离 $d_{\text{w}}(a,b)$ 为 w_a 和 w_b 在 Wikipedia 知识网络中最短路径的长度，需要验证：对于每对概念 a 和 b，若 $d_{\text{rt}}(a,b)$ 越大，则 $d_{\text{w}}(a,b)$ 越小。使用以下 2 个方法进行验证：

方法 1 对于每对概念 a 和 b，计算 $d_{\text{rt}}(a,b)$ 和 $d_{\text{w}}(a,b)$，统计所有的 d_{rt} 和 d_{w} 中，当 d_{rt} 一定时对应的 d_{w} 的平均值，制作折线图如图 6 所示。由图 6 可知，随着 topic 反向距离的上升，链接的 Wikipedia 知识条目距离呈下降趋势，说明本文的层次分类可以体现 topic 概念之间的关系。

方法 2 对于每个概念 a，计算其他所有概念与之的 topic 反向距离与 Wikipedia 知识条目距离，求出所得结果的 Kendall 相关系数。Kendall 相关系数根据 2 个随机变量排名的一致程度评估 2 个随机变量之间的相关关系，相关系数小于 0 表示 2 个变量呈负相关。对于每个概念 a，若其他所有概念与之的 topic 反向距离与 Wikipedia 知识条目距离的 Kendall 相关系数小于 0，说明对于 a，其他概念与之的 topic 反向距离越大，Wikipedia 知识条目距离越小。根据计算，87.8%的概念所得 Kendall 相关系数小于 0，说明从整体上看，当 a 确定时，对于其他概念 b，$d_{\text{rt}}(a,b)$ 越大，$d_{\text{w}}(a,b)$ 越小。

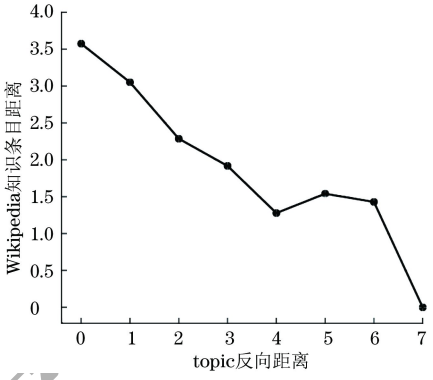

图 6 topic 反向距离与链接的 Wikipedia 知识条目平均距离的关系

Fig.6 Relationship between topic reverse distance and average distance of linked Wikipedia knowledge items

这 2 个验证方法存在一定的局限性：首先，由于 topic 簇树的叶子节点深度并不统一，方法 1 有可能存在偏差，特别是 topic 反向距离大于等于 4 时，样本数量较少，存在较大的偶然性，导致 topic 反向距离为 4 时结果不符合趋势；其次，topic 簇树与 Wikipedia 知识网络的结构不同，计算距离会存在偏差；最后，在 Wikipedia 知识网络中的距离并不能精确地表示 2 个概念实际语义间的距离，只能获得一个大概的趋势，因此验证结果无法完全符合预期。

此外，该验证也说明了本文对 topic 层次分类的方法并不完美。利用 topic 之间的共现关系，采用无监督的层次聚类方法获得 topic 的层次分类，难以获得完全精确的分类结果。但从整体上看，本文对 topic 的层次分类可以体现 topic 之间的语义关系。

3.4 topic 层次化检索服务效果验证

对于 topic 层次化检索服务效果的验证主要以调查问卷的形式进行，问卷面向高校具有计算机或软件开发相关背景的学生，共收集 48 份。问卷主要分为 2 个部分：

第 1 个部分调查了用户的相关背景信息，具体如下：

问题 1 你的软件开发水平处于哪个层级？【单选题】选项：

A. 没有接触过软件开发；

B. 正在学习软件开发知识，参与过 0 或 1 个软件项目的开发；

C. 拥有较为丰富的软件开发经验，参与过 2 个及以上的软件项目的开发，熟悉 1 种或多种软件开发框架。

问题 2 你对于前端开发的熟悉程度如何？【评分题】评分选项为 0～5 分，评分越高表示越熟悉。

问题 3 你对于安卓开发的熟悉程度如何？【评分题】评分选项为 0～5 分，评分越高表示越熟悉。

在问题 1 结果中，6 名用户选择 A 项，33 名用户选择 B 项，这 2 类用户可被视为软件开发初学者，对本文服务效果的验证具有较大意义，还有 9 名用户选择 C 项。问题 2 和问题 3 调查了用户对接下来的 2 个案例主题(前端开发和安卓开发)的熟悉程度，结果如图 7 所示。由图 7 可知，大部分用户对 2 个案例的主题并不精通，可以看成前端开发和安卓开发的初学者。

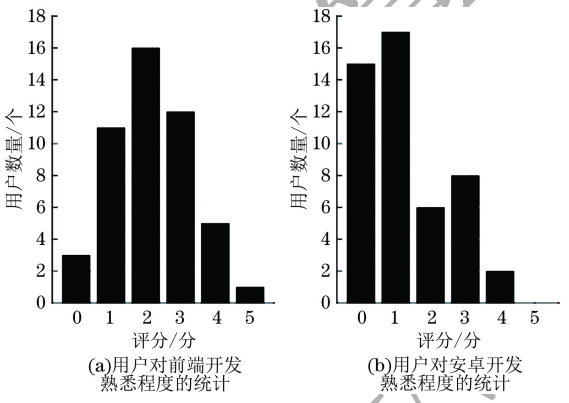

图 7 用户对前端和安卓开发熟悉程度的统计

Fig. 7 Statistics of user familiarity with front-end and Android development

第 2 个部分使用 2 个案例对 topic 层次化检索服务效果进行验证，将用户置于一个正在学习软件开发某方面知识且具有搜索与学习的大方向但对接下来的搜索方向并不明确的场景中，模拟用户在学习前端或安卓开发知识的情况下对于层次化检索服务的使用，根据用户的选择提出下一个问题。案例问题结构如图 8 所示。

针对案例主题的选择，前端和安卓在本文所得的层次分类中分属 2 个不同的大分类，且前端开发和安卓开发都是软件开发者中的热门技术，根据 Stack Overflow 发布的《2023 Developer Survey》调查报告，前端开发和移动端开发者数量均排在前 5，而安卓开发又是移动端开发的重要部分，由此可知这 2 个主题具有较强的代表性，因此采用这 2 个案

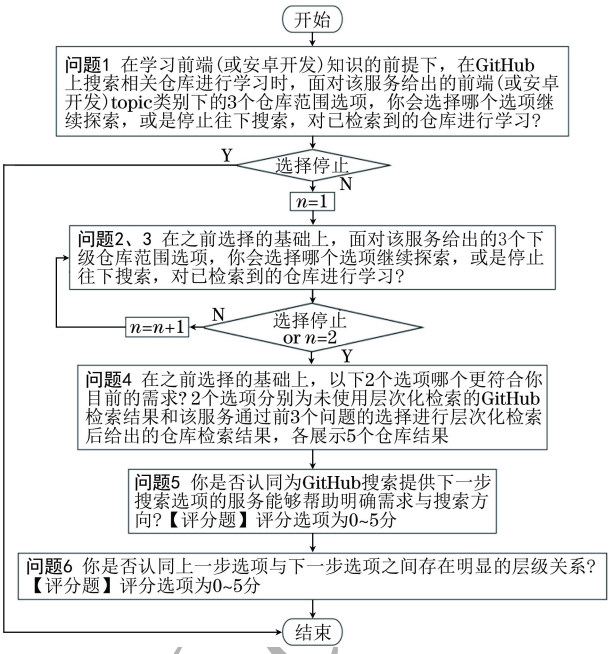

图 8 案例问题结构

Fig. 8 Case question structure

例主题。

若用户在前 3 个问题中选择了层次化检索服务所提供的仓库领域选项，则将用户选择该选项的次数记为该服务帮助用户延长的检索路径长度，最长为 3，延长的检索路径长度越长，说明层次化检索帮助用户细化了越多搜索条件，也说明用户对于层次化检索的需求程度越大。问题 1～3 调查了用户使用该服务的意愿和该服务为用户延长的检索路径长度，从而验证了该服务帮助用户细化搜索条件的程度；问题 4 验证了该服务的检索结果是否比 GitHub 的检索结果更符合用户的需求；问题 5 调查了用户模拟使用该服务后对服务效果的评价；问题 6 调查了案例中 topic 层次分类的效果，即上级与下级之间是否存在层次关系。

分别统计了软件开发初学者用户对于各问题的回答。若用户在问题 1 中选择了该服务给出的选项，说明用户有使用该层次化检索服务进行辅助搜索的意愿，在案例 1 中这一比例为 69.2%，在案例 2 中这一比例为 79.5%，说明大部分用户会选择使用该层次化检索服务进行辅助搜索。层次化检索服务为用户延长的检索路径长度的统计结果如表 5 所示，由此可知层次化检索服务可以有效延长用户的检索路径，明确用户的搜索方向。由表 6 可知，层次化检索服务的检索结果比不使用层次化检索而直接在 GitHub 上搜索的结果仓库更符合用户的需求。表 7 显示了大部分用户对于提供下一步搜索选项能够帮助明确需求与搜索方向持不同的认同度。表 8

显示了用户对父子仓库领域之间存在的层级关系的平均认同度较高，说明了 topic 层次分类能够体现 topic 的层次结构。

表 5 层次化检索服务为用户延长检索路径长度的调查结果

Table 5 Survey results of length of retrieval path extended for users through hierarchical retrieval service

案例	延长的路径长度为 1 的用户比例/%	延长的路径长度为 2 的用户比例/%	延长的路径长度为 3 的用户比例/%	为用户延长的平均检索路径长度
案例 1	37.0	25.9	37.0	2.00
案例 2	6.5	12.9	80.6	2.74

表 6 层次化检索服务检索结果符合用户需求的情况统计

Table 6 Situation statistics of compliance of the retrieval results of the hierarchical retrieval service with users' needs %

案例	GitHub 搜索结果更符合需求的比例	层次化检索服务搜索结果更符合需求的比例
案例 1	40.7	59.3
案例 2	35.5	64.5

表 7 用户对提供下一步搜索选项帮助明确需求与搜索方向的认同度统计

Table 7 Statistics of user recognition of providing next search options to help clarify their needs and search direction

案例	1 分占比/%	2 分占比/%	3 分占比/%	4 分占比/%	5 分占比/%	平均得分/分
案例 1	0	12.8	41.0	43.6	2.6	3.36
案例 2	0	15.4	25.6	53.8	5.1	3.49

表 8 用户对父仓库领域与子仓库领域之间存在层级关系的认同度统计

Table 8 Statistics of user recognition of the hierarchical relationship between the parent and child repository domains

案例	1 分占比/%	2 分占比/%	3 分占比/%	4 分占比/%	5 分占比/%	平均得分/分
案例 1	0	15.4	33.3	43.6	7.7	3.44
案例 2	0	7.7	41.0	46.2	5.1	3.49

4 结束语

本文建立一种 GitHub 层次化学习和检索服务系统架构，融合 Wikipedia 知识目录和 GitHub topic 层级结构构造软件开发领域知识图谱，并在此基础上实现层次化学习和检索服务，从而使用户可以层次化浏览软件开发领域知识，同时结合 GitHub 仓库进行层次化学习，并通过逐层在候选项中选择感兴趣的仓库领域，层次化检索 GitHub 仓库功能。后续将进一步挖掘 GitHub topic 之间的关系，由于 GitHub 仓库的 topic 设置具有随意性，因此将对 topic 之间的层次、复制关系等进行研究，以获得 GitHub 社区内部更精确的层次关系，同时进一步拓宽应用场景，将层次化学习和检索扩展到更大的数据集上，为软件开发初学者提供更加丰富的仓库资源，提升学习与检索效果。

参考文献

[1] WANG S T, ORORBIA A, WU Z H, et al. Using prerequisites to extract concept maps from textbooks[C]// Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. New York, USA: ACM Press, 2016: 317-326.

[2] CHEN P H, LU Y, ZHENG V W, et al. KnowEdu: a system to construct knowledge graph for education[J]. IEEE Access, 2018, 6: 31553-31563.

[3] DANG F R, TANG J T, PANG K Y, et al. Constructing an educational knowledge graph with concepts linked to Wikipedia[J]. Journal of Computer Science and Technology, 2021, 36(5): 1200-1211.

[4] LIANG C, WU Z H, HUANG W Y, et al. Measuring prerequisite relations among concepts[C]// Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, USA: Association for Computational Linguistics, 2015: 1668-1674.

[5] HUANG X Q, LIU Q, WANG C, et al. Constructing educational concept maps with multiple relationships from multi-source data[C]// Proceedings of IEEE International Conference on Data Mining (ICDM). Washington D. C., USA: IEEE Press, 2019: 1108-1113.

[6] YIN H, SUN Z Y, SUN Y C, et al. Automatic learning path recommendation for open source projects using deep learning on knowledge graphs[C]// Proceedings of the 45th Annual Computers, Software, and Applications Conference (COMPSAC). Washington D. C., USA: IEEE Press, 2021: 824-833.

[7] BORGES H, HORA A, VALENTE M T. Understanding the factors that impact the popularity of GitHub repositories [C]// Proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME). Washington D. C., USA: IEEE Press, 2016: 334-344.

[8] ZANARTU F, TREUDE C, CARTAXO B, et al. Automatically categorising GitHub repositories by application domain[EB/OL]. [2023-09-19]. https://arxiv.org/abs/2208.00269.

[9] SOLL M, VOSGERAU M. ClassifyHub: an algorithm to classify GitHub repositories [C] // Proceedings of Joint German/Austrian Conference on Artificial Intelligence. Berlin, Germany: Springer, 2017: 373-379.

[10] SHARMA A, THUNG F, KOCHHAR P S, et al. Cataloging GitHub repositories[C]//Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. New York, USA: ACM Press, 2017: 314-319.

[11] ZHANG Y, XU F F, LI S, et al. HiGitClass: keyword-driven hierarchical classification of GitHub repositories[C]//Proceedings of IEEE International Conference on Data Mining (ICDM). Washington D. C., USA: IEEE Press, 2019: 876-885.

[12] ASYROFI M H, THUNG F, LO D, et al. AUSearch: accurate API usage search in GitHub repositories with type resolution [C] // Proceedings of the 27th International Conference on Software Analysis, Evolution and Reengineering. Washington D. C., USA: IEEE Press, 2020: 637-641.

[13] CHEN M, LI G, MA C, et al. Repo4QA: answering coding questions via dense retrieval on GitHub repositories[C]// Proceedings of the 29th International Conference on Computational Linguistics. Stroudsburg, USA: ACL Press, 2022: 1580-1592.
[14] LIU J Z, LI Z X, WANG T, et al. Adaptive software search toward users' customized requirements in GitHub[C]// Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering. [S. l.]: KSI Research Inc. and Knowledge Systems Institute Graduate School, 2018: 143-152.
[15] WU J W, SUN Y C, ZHANG J Q. An open-source repository retrieval service using functional semantics for software developers[C]// Proceedings of International Conference on Service Science(ICSS). Washington D. C., USA: IEEE Press, 2022: 12-20.
[16] CAI X Y, ZHU J G, SHEN B J, et al. GRETA: graph-based tag assignment for GitHub repositories[C]// Proceedings of the 40th Annual Computer Software and Applications Conference(COMPSAC). Washington D. C., USA: IEEE Press, 2016: 63-72.
[17] MATEK T, TIMEJ Z S. GitHub open source project recommendation system[EB/OL]. [2023-09-19]. https://arxiv.org/abs/1602.02594.
[18] GUENDOUZ M, AMINE A, HAMOU R M. Recommending relevant open source projects on GitHub using a collaborative-filtering technique[J]. International Journal of Open Source Software and Processes, 2015, 6(1): 1-16.
[19] KOSKELA M, SIMOLA I, STEFANIDIS K. Open source software recommendations using GitHub[M]. Berlin, Germany: Springer International Publishing, 2018.
[20] YANG H, SUN S, WEN J H, et al. Improving personalized project recommendation on GitHub based on deep matrix factorization[C]//Proceedings of International Conference on Collaborative Computing: Networking, Applications and Worksharing. Berlin, Germany: Springer, 2021: 318-332.
[21] KIM J, WI J, KIM Y. Sequential recommendations on GitHub repository[J]. Applied Sciences, 2021, 11(4): 1585.
[22] ZHOU Y Q, WU J W, SUN Y C. GHTRec: a personalized service to recommend GitHub trending repositories for developers[C]//Proceedings of IEEE International Conference on Web Services(ICWS). Washington D. C., USA: IEEE Press, 2021: 314-323.
[23] BORGES H, VALENTE M T. What's in a GitHub star? Understanding repository starring practices in a social coding platform[J]. Journal of Systems and Software, 2018, 146: 112-129.
[24] TRAAG V A, WALTMAN L, VAN ECK N J. From Louvain to Leiden: guaranteeing well-connected communities[J]. Scientific Reports, 2019, 9: 5233.
[25] FREEMAN L C. Centrality in social networks conceptual clarification[J]. Social Networks, 1978, 1(3): 215-239.
[26] ALY M. Survey on multiclass classification methods[EB/OL]. [2023-09-19]. https://www.semanticscholar.org/paper/Survey-on-Multiclass-Classification-Methods-Mehra-Gupta/f0fcf860031b356a3c68b330735634c00e5d7602.
[27] NAIK A, RANGWALA H. Large scale hierarchical classification: state of the art[M]. Berlin, Germany: Springer International Publishing, 2018.
[28] SILVA-PALACIOS D, FERRI C, RAMÍREZ-QUINTANA M J. Probabilistic class hierarchies for multiclass classification[J]. Journal of Computational Science, 2018, 26: 254-263.
[29] ROBILLARD M, WALKER R, ZIMMERMANN T. Recommendation systems for software engineering[J]. IEEE Software, 2010, 27(4): 80-86.

编辑 陆燕菲