计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(8):237−242 [doi: 10.15888/j.cnki.csa.008085]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于用户反馈的 API 推荐工具①

杨忻莹[1], 周　宇[1,2]

[1](南京航空航天大学 计算机科学与技术学院, 南京 211106)
[2](南京航空航天大学 高安全系统的软件开发与验证技术工信部重点实验室, 南京 211106)
通讯作者: 周　宇, E-mail: zhouyu@nuaa.edu.cn

摘　要: 在软件开发的过程中, 开发人员经常会检索合适的 API 来完成编程任务. 为了提高软件开发效率, 大量 API 推荐方法及工具应运而生. 然而, 这些方法大多数都没有考虑用户交互信息. 本文提出了一个基于客户端/服务器架构的 API 推荐工具, 将其以插件的形式集成到 VS Code IDE 中. 本工具使用现有的 API 推荐工具生成初始 API 推荐列表, 结合用户反馈信息, 利用排序学习和主动学习技术对 API 推荐列表进行重新排序, 实现了用户个性化推荐. 大量实验证明, 随着反馈数据量的增加, 本工具的性能稳步提升.

关键词: API 推荐; 用户反馈; 排序学习; 主动学习

引用格式: 杨忻莹,周宇.基于用户反馈的 API 推荐工具.计算机系统应用,2021,30(8):237–242. http://www.c-s-a.org.cn/1003-3254/8085.html

API Recommendation Tool Based on User Feedback

YANG Xin-Ying[1], ZHOU Yu[1,2]

[1](School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)
[2](Key Laboratory of the Ministry of Industry and Information Technology for Software Development and Verification Technology for High Security Systems, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract: Developers often search for appropriate APIs to complete programming tasks during software development. Many API recommendation approaches and tools have been proposed to improve the efficiency of software development, but most of these approaches do not consider user interaction information. In this study, we propose an API recommendation tool based on client/server architecture and integrate it into the VS Code IDE in the form of a plug-in. In our tool, initial API recommendation lists are generated by existing tools. Learning-to-rank and active learning techniques are used, combined with user feedback information, to re-rank the API recommendation lists, achieving personalized recommendation. Extensive experiments demonstrate that the performance of this tool has been steadily improved with the increase in feedback data.

Key words: API recommendation; user feedback; learning-to-rank; active learning

在软件开发过程中, 应用程序接口 (API) 发挥了重要作用[1], 开发人员可以使用 API 更有效地执行编程任务. 但是由于 API 的数量庞大, 开发者不可能熟悉所有的 API, 也不可能始终为特定的开发任务选择合适的 API. 当遇到不熟悉的编程任务时, 开发者往往会通过上网搜索相关功能的案例进行代码复用, 或者搜索相关的 API 文档, 学习相关 API 的使用方法. 能够针对程序员的需求进行合理的 API 接口推荐便成为了提高软件开发效率的重要途径之一. 为了解决这个问题, 国内外大量科研人员已在 API 接口推荐领域展开了一系

① 基金项目: 国家重点研发计划 (2018YFB1003902); 国家自然科学基金 (61972197)
Foundation item: National Key Research and Development Program of China (2018YFB1003902); National Natural Science Foundation of China (61972197)
收稿时间: 2020-11-24; 修改时间: 2020-12-22, 2021-01-08; 采用时间: 2021-01-26; csa 在线出版时间: 2021-07-31

列研究工作, 并实现了许多 API 推荐方法, 以减轻开发人员理解和搜索 API 的负担.

Thung 等通过将历史特征请求与 API 文档信息相结合的方式, 进行 API 方法推荐[2]. BIKER[3] 获取 Stack Overflow 问答网站上的问答信息和 JDK 的 API 文档信息[4], 根据帖子问题与用户查询的相似性推荐 API.

这些方法往往基于信息检索技术以及自然语言处理相关技术提取关键字, 来缩小目标 API 的搜索范围、加快推荐效率. 然而他们大多均未考虑用户交互信息 (例如用户从推荐列表中选择 API), 这些信息通常被认为能够大大提高 API 推荐性能.

针对这一不足之处, 有少数工作利用了用户反馈信息. 例如, NLP2API[5] 利用来自 Stack Overflow 网站的问答对来模拟伪用户交互, 从而重新制定查询语句, 实际上并没有进行真正的用户交互. Wang 等[6] 将反馈纳入到代码搜索过程中, 提出了一种主动代码搜索方法, 尽管他们的工作利用了反馈信息, 但需要用户明确的对推荐结果中的每一条信息进行相关性评分, 开销太大. 这部分工作虽然将反馈信息从传统的推荐系统领域引入到代码推荐领域, 但仍有改进的空间.

因此, 在现有的 API 推荐工作上, 本文提出一个新颖的 API 推荐工具.

首先, 记录用户使用本工具时产生的真实交互信息 (即用户输入的查询语句和用户从推荐列表中选择的 API), 将其作为用户反馈数据, 即反馈信息, 并构建反馈库, 将反馈数据以<用户查询, API>对的形式存入其中. 故用户在使用时不需要主动提供额外信息 (如评分), 可减轻用户负担.

其次, 引入排序学习和主动学习技术以提升 API 推荐性能. 通过将 API 视为文档, 本工作将排序学习技术应用至 API 推荐领域中, 利用用户反馈等数据构建特征向量, 训练排序学习模型, 将用户反馈有效的融入到 API 推荐中, 以提高 API 推荐结果的准确性; 为了缓解“冷启动”问题、加速反馈数据的学习过程, 本工作采用了主动学习, 通过收集 Stack Overflow 的问答信息, 构造问答数据对, 用于辅助训练主动学习模型, 从而保证即使在反馈数据十分稀缺时, 仍能实现良好的推荐效果.

最后, 为了实现更好的扩展性, 可利用第三方 API 推荐方法作为组件, 根据用户需求, 灵活地嵌套到其他的 API 推荐方法之上.

本工具以插件的形式集成到了 VS Code IDE 中, 向用户提供 API 推荐, 这将会对 API 推荐系统产生积极影响. 值得强调的是, 尽管本工作目前专注于 Java, 但是我们希望可以将其应用于其他编程语言的 API. 本工作不仅提出了一种推荐方法, 而且提高了 API 推荐性能. 本文着重介绍了本工作的工具实现, 读者可参考原始论文[7] 以获得更多详细信息.

为了评估所提出方法的有效性, 本文实验选择了最先进的 API 推荐方法 BIKER[3] 作为对比, 利用 Hit@k/Top-k, MAP, MRR 作为评估指标. 实验结果表明, 对比 BIKER, 本工具 API 推荐的 Hit@1 准确率相对提升了 18.2%, 高达 51.8%.

1 方法实现

工作流程如下, 首先从 JDK 8 官方文档中提取用于描述 API 类和方法的总结句, 作为 API 文档, 再利用现有 API 推荐方法, 对用户输入的自然语言查询语句进行推荐, 得到初始 API 推荐列表. 通过语义相似度计算模型 (1.1 节) 进行相似度计算, 对 API 推荐列表中的 API 构建特征向量 (1.2 节), 利用训练好的排序学习和主动学习模型 (1.3 节), 对初始 API 推荐列表进行重新排序, 并向用户呈现重排序的 API 推荐列表 (1.4 节). 此外, 可将用户从列表中选择的最合适的 API, 与本条查询语句一起形成反馈数据, 存入反馈库中 (1.5 节), 再根据反馈数据对推荐结果进行优先级调整, 使得用户选择的推荐项位于推荐结果列表中更加靠前的位置, 从而提高 API 推荐准确率. 本工作的具体实现细节可以在文献 [7] 中找到, 在此不做赘述. 本工具的整体架构如图 1 所示.

1.1 语义相似度计算模型构建

本模块构建一个语义相似度计算模型, 为特征提取模块提供相似度计算方法. 首先从 Stack Overflow[8] 问答网站上下载历史问答信息[9], 问答信息以.xml 格式文件保存. 由于本工具是针对 Java 相关的 API 推荐, 因此只提取带有 Java 标记的帖子信息. 随后利用 NLTK[10] 对帖子信息进行传统的文本预处理操作, 包括标记和词干分析. 接着使用 Word2Vec[11] 对词嵌入模型进行训练, 并计算预处理后语料库中每个单词的 IDF (逆文档频率), 从而构建一个 IDF 逆文档频率表作为词嵌入模型的权重项. 利用训练得到 Word2Vec 模型和 IDF 逆文档频率表文档, 构建语义相似度计算模型[12].

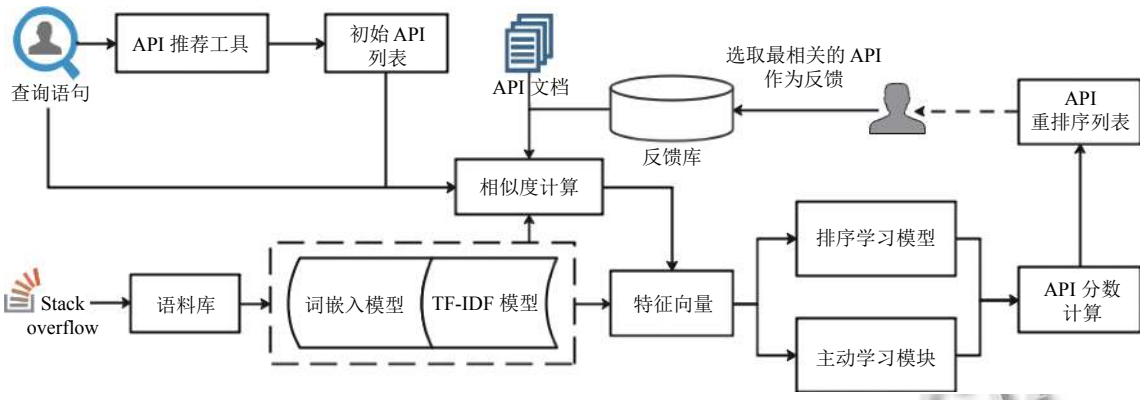

图 1 方法整体架构

1.2 特征提取

特征提取模块主要任务是根据构建的语义相似度计算模型, 进行相似度计算, 并构建特征向量, 构造完成的特征向量可作为训练模型或预测模型的输入数据.

在接收到查询语句、初始 API 推荐列表之后, 结合反馈库数据, 为初始 API 推荐列表中每个 API 提取一个特征向量, 特征向量包括两个部分, 相关信息特征和反馈特征. 相关信息特征可从 API 文档中获取, 由 API 路径功能和 API 描述功能组成, 分别表示推荐的 API 与用户查询和相关文档描述的相关性; 反馈特征可从反馈库中提取, 代表与反馈库中 API 的相关性.

1.3 排序学习和主动学习模型训练

本模块用于训练排序学习和主动学习模型, 为 API 重排序模块提供模型支撑. 通过 1.2 节中特征提取过程, 对反馈库中的每条反馈数据提取特征向量, 以构建训练集.

通过将特征向量输入到排序学习模型, 最终得到训练良好的排序学习模型.

训练主动学习模型之前, 先利用来自 Stack Overflow 的问答信息构造<查询, API>问答数据对, 其中查询与问题相对应, API 与接受的答案中的 API 相对应. 这些问答数据对可对主动学习模型选取的样本提供标注, 将其放入训练集中再次训练模型. 通过迭代此过程, 可以获得训练好的主动学习模型.

1.4 API 重排序推荐

本模块的目的是根据训练好的模型, 得到用户输入查询对应的重排序 API 推荐列表, 供用户选择. 对于用户输入查询语句, 首先通过 1.2 节得到其对应的 API 特征向量, 将其分别输入到训练好的排序学习和主动学习模块中, 得到各个模块的预测值, 计算出 API 推荐列表中每个 API 的分数. 列表中的 API 排名位置根据分数降序排列, 得到重排序 API 推荐列表, 供用户进行选择, 使得用户选择的 API 在列表中位于更加靠前的位置. API 综合得分的计算公式如下:

$$AllScore_Q(i)=\frac{Score_Q(i)-Score_{\min}}{Score_{\max}-Score_{\min}}+\mu ALrelev_Q(i) \quad (1)$$

其中, $Score_Q(i)$表示第i个 API 在初始 API 列表中的排序学习预测值, $ALrelev_Q(i)$表示列表中第i个 API 的排序学习预测值. $Score_{\max}$和$Score_{\min}$分别为排序分数的最大值和最小值, 用于进行归一化处理. 相关性得分的权重是一个动态值, 它依赖于第i个 API 在列表中所处的位置, 即pos_i.

1.5 用户反馈

用户反馈模块负责更新反馈库. 用户从重排序 API 推荐列表中, 根据自身需求, 选择 API, 将查询语句和用户所选 API 作为反馈数据, 存入反馈库, 为后续推荐提供反馈信息.

2 工具实现

本工具以插件的形式集成到 VS Code IDE 中, 采用客户端/服务器架构. 服务器端对 VS Code 获取的用户输入数据进行处理, 返回数据处理结果, 并通过客户端将结果展示给用户.

为方便用户在编程过程中快速查询所遇到的问题, 本工具直接内嵌在 VS Code IDE 中, 不涉及另外的系统界面. 用户可以在源码编程面板任意位置点击右键, 或者使用快捷键唤醒.

根据用户交互信息, 本工具为用户提供了个性化的 API 推荐列表, 实现了 API 个性化推荐功能. 系统架构主要分为两个模块: 用户输入查询模块和 API 推荐

反馈模块.

2.1 用户输入查询模块

用户输入查询模块的整体流程如图 2 所示, 该模块的目的是提供一个接口, 以获取用户输入的自然语言查询语句, 传递给后台服务器进行 API 推荐的后续处理工作.

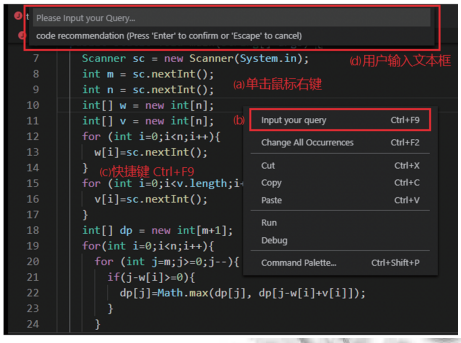

图 2 用户输入查询流程

具体来说, 当用户在编程环境中, 想要获取 API 推荐, 可以通过图 2 中的操作 (a)、(b) 或 (c) 打开文本输入框 (d) 以进行自然语言查询输入. 客户端从文本输入框中获得查询语句, 传给服务器端进行处理, 再将得到的推荐结果返回给用户, 即 API 推荐列表页面 (图 3).

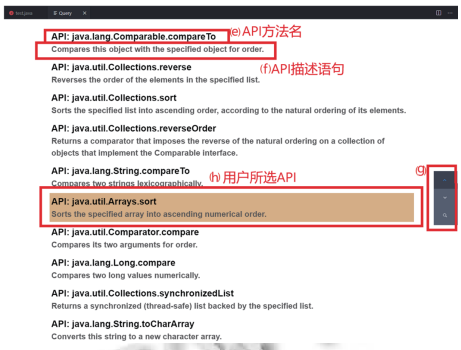

图 3 API 推荐列表页面

2.2 API 推荐反馈模块

服务器获取自然语言查询语句后, 会返回相关的 API, 客户端为用户展示如图 3 所示的 API 推荐列表页面. 用户可在该页面上进行 API 选择, 客户端将查询语句连同用户选择的 API 视为反馈数据传递给服务器. 服务器将其放入反馈库, 用以优化模型.

2.2.1 API 推荐

服务器将与查询语句最相关的前 30 个 API 返回给客户端, 客户端以每页 10 个 API 的方式分页展示, 用户可以通过图 3 中操作 (g) 切换页面, 以浏览更多的 API. 本工具不仅为列表中每个 API 提供 API 方法名 (e), 还提供对应的描述语句 (f), 为用户选择反馈提供参考依据.

2.2.2 用户反馈

用户可以选中一个 API(h), 此时会弹出一个对话框, 向用户确认是否将该 API 作为反馈信息存入反馈库中.

若用户选择“是”, 则该 API 与本次输入的自然语言查询语句一起构成反馈数据, 存入反馈库, 为下一次用户查询提供更加个性化的推荐; 若用户选择“否”, 则会返回至 API 推荐列表界面.

3 应用场景

本章通过示例来说明本工具如何提升 API 推荐性能. 在通常情况下, API 方法名和文档描述信息有助于用户了解 API 的用途. 对于诸如: How to convert int to string 的查询语句, 用户在浏览 API 方法名时可以快速找到正确的答案: java.lang.Integer.toString. 但是, 在许多情况下, 仅通过 API 方法名来判断该 API 是否有用是很困难的. 例如, 给定查询: Make a negative number positive, 其对应的答案: java.lang.Math.abs 在语义上与查询语句没有重叠, 但借助文档描述信息 (即: 返回参数的绝对值), 可以很容易判断该 API 是正确答案.

当用户了解 API 的用法之后, 可以选择他认为合适的答案, 放入反馈库中保存, 这些反馈信息对于后续的推荐有显著帮助. 例如, 对于查询语句: killing a running thread in java, 在反馈库中没有相关查询语句的情况下, 根据 API 方法名和描述信息, 用户能很快确定排名第 7 的 API: java.lang.Thread.interrupt 是正确答案, 并对其进行选择, 放入反馈库中存储. 当用户之后再次查询类似的语句, 例如: how to stop a thread, 通过检索反馈库数据, 本工具能将原本排在第 8 位的: java.lang.Thread.interrupt 提升到第 1 位. 使得用户浏览 API 推荐结果时, 能迅速找到正确答案, 极大提升了 API 推荐效率.

4 实验评估

为了验证方法的有效性, 本文使用目前最先进的 API 推荐方法之一的 BIKER[3] 作为对比实验. BIKER 和现有的大多数 API 推荐方法一样, 仅利用了语义相

似度技术, 根据文本相似度分数向用户推荐 API. 而本文提出了一种基于用户反馈的 API 推荐工具, 将用户反馈融入到 API 推荐过程中, 并且采用了排序学习和主动学习技术, 通过语义相似度构建特征向量, 训练模型进行预测, 根据模型预测综合得分, 为用户提供个性化的 API 推荐.

为了确保与对比实验进行公平的比较, 本文实验的数据集复用 BIKER 发布的数据集, 即来自 Stack Overflow 上与 Java API 相关的 413 个问答对. 为了评估实验性能, 实验设置遵循标准的 10 折交叉验证, 即将数据集随机分割为 9:1, 每次使用 1 折作为测试数据, 其余 9 折用于模型训练. 重复实验 5 次, 将结果记录下来, 并计算平均值作为最终结果. 为了避免偏差, 反馈库与测试集中不存在相同的查询语句. 读者可以查看之前的工作[7], 以获取有关实验结果的更多详细信息.

4.1 增加反馈数据量

通过从训练集中随机选择问答对, 来构建反馈库. 反馈库的大小从训练集的 10% 到 100% 不等, 且增幅为 10%. 实验结果如表 1 所示, 结果表明, 在可接受的开销条件下, 与 BIKER 相比, 随着反馈数据的增加, 本工具的推荐性能稳步提升, Hit@1 准确率最高提升了 18.2%, MAP 和 MRR 提升了 12% 左右, Hit@5 突破了 80%. 实验结果验证了反馈信息有助于提升 API 推荐性能.

表 1 增加反馈数据量对工具的影响

反馈库	Hit@1	Hit@3	Hit@5	MAP	MRR
BIKER[3]	0.423	0.661	0.775	0.553	0.569
10%	0.442	0.682	0.783	0.569	0.582
20%	0.470	0.702	0.795	0.592	0.604
30%	0.493	0.714	0.802	0.607	0.619
40%	0.499	0.718	0.806	0.611	0.623
50%	0.502	0.718	0.807	0.613	0.625
60%	0.507	0.719	0.807	0.616	0.628
70%	0.511	0.720	0.808	0.618	0.631
80%	0.515	0.720	0.809	0.621	0.633
90%	0.517	0.721	0.810	0.621	0.635
100%	0.518	0.721	0.811	0.622	0.636

4.2 有效性分析

为了评估实验结果的意义, 本节对得到的结果进行了统计分析. 根据 Mann-Whitney U 检验[13], 可从统计意义上判定实验是否有显著的改善. 此外, 通过 Vargha 和 Delaneys 的$\hat{A}_{12}$测量 (一种标准化的非参数效应量测量) 来评估分析效应量的改进程度. 一般来说, 对于 A 和 B 两种算法, 如果$\hat{A}_{12}$为 0.5, 则认为这两种算法是等价的. 如果$\hat{A}_{12}$大于 0.5, 则算法 A 优于算法 B. $\hat{A}_{12}$通过以下统计量计算:

$$\hat{A}_{12}=\frac{\frac{R_1}{m}-\frac{m+1}{2}}{n} \tag{2}$$

其中, R_1是第一个数据组的秩和, m 和 n 分别是第一个和第二个数据样本中的观察次数. 在本文实验中, 运行了相同次数的两种算法, 即 m 和 n 的值都设置为 5.

结果表明, 大多数 p 值在 0.003 到 0.005 范围内, 效应量为 1, 表明在 96% 的置信度上, 该提升具有统计学意义. 在 50 个案例中有 2 个案例 (反馈数据量为 10% 时的 Hit@3 和 Hit@5), 其中 p 值大于 0.005(即, 应否定原假设).

4.3 模拟用户反馈场景

本节设计了一个伪用户来模拟真实场景下的用户推荐及反馈过程. 实验随机选择 50 个查询, 将其作为伪用户在编程环境中输入的查询语句, 依次进行自然语言查询. 在每次查询期间, 本工具都会根据反馈库信息推荐 API, 而伪用户则会从推荐结果中选择 API. 查询语句和选择的 API 用于扩展反馈库. 实验结果如表 2 所示, 通过记录每次用户查询的评价指标, 可计算出评价指标的平均值 (表 2 Avg. Tool), 通过计算本文方法与对比方法的绝对提升值和相对提升值 (表 2 Abs. Imp.和 Rel. Imp.), 可以更清晰的看出本工作相对于对比实验的优越性.

表 2 模拟用户反馈场景

指标	Hit@1	Hit@3	Hit@5	MAP	MRR
BIKER[3]	0.423	0.661	0.775	0.553	0.569
Avg. Tool	0.480	0.700	0.800	0.592	0.598
Abs. Imp. (%)	5.7	3.9	2.5	3.9	2.8
Rel. Imp. (%)	13.4	5.9	3.3	7.0	5.0

5 结论与展望

本文提出了一种新的工具来提升 API 推荐方法的性能, 并将其以插件的形式集成到 VS Code IDE 中. 在用户编程环境中, 通过获取用户输入的自然语言查询语句, 推荐相关 API. 受传统推荐系统中的用户反馈信息的启发, 本工作通过不断收集用户对于 API 推荐列表的反馈, 提高推荐准确率. 实验表明, 本工具的推荐性能优于对比实验, 随着反馈信息数量的增加, API 推

荐的有效性也显著提升. 未来的工作计划是扩展工具以支持更多的编程语言. 此外, 本文中提出的方法实际上具有更广泛的适用性, 因此计划将其扩展到软件工程中的其他推荐场景.

参考文献

1 Brandt J, Guo PJ, Lewenstein J, *et al*. Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. Proceedings of the 27th International Conference on Human Factors in Computing Systems. Boston, MA, USA. 2009. 1589–1598. [doi: 10.1145/1518701.1518944]

2 Thung F, Wang SW, Lo D, *et al*. Automatic recommendation of API methods from feature requests. Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering. Silicon Valley, CA, USA. 2013. 290–300. [doi: 10.1109/ASE.2013.6693088]

3 Huang Q, Xia X, Xing ZC, *et al*. API method recommendation without worrying about the task-API knowledge gap. Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. Montpellier, France. 2018. 293–304.

4 Java documentation. http://docs.oracle.com/java-se/8/docs/api/. (2019-05-06).

5 Rahman MM, Roy CK. Effective reformulation of query for code search using crowdsourced knowledge and extra-large data analytics. Proceedings of 2018 IEEE International Conference on Software Maintenance and Evolution. Madrid, Spain. 2018. 473–484.

6 Wang SW, Lo D, Jiang LX. Active code search: Incorporating user feedback to improve code search relevance. Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. Vasteras, Sweden. 2014. 677–682.

7 Zhou Y, Yang XY, Chen TL, *et al*. Boosting API Recommendation with Implicit Feedback. IEEE Transactions on Software Engineering, 2021. [doi: 10.1109/TSE.2021.3053111]

8 Stack overflow. https://stackoverflow.com/. (2019-04-13).

9 Jentsch M. Stack overflow data dump. http://www.dieletztedomain.de/stack-overflow-data-dump/. (2014-01-27).

10 NLTK. http://www.nltk.org/. (2019-04-28).

11 Word2Vec. https://radimrehurek.com/gensim/models/word2vec.html.(2019-04-28).

12 Mihalcea R, Corley CD, Strapparava C. Corpus-based and knowledge-based measures of text semantic similarity. Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference. Boston, MA, USA. 2006. 775–780.

13 Arcuri A, Briand L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. Proceedings of the 33rd International Conference on Software Engineering. Honolulu, HI, USA. 2011. 1–10.