# A P I H A RV E ST：从各种在线来源收集 API 信息

Ferdian Thung新加坡管理大学新加坡
ferdianthung@smu.edu.sg

Kisub Kim新加坡管理大学
新加坡
kisubkim@smu.edu.sg

Ting Zhang新加坡管理大学新加坡
tingzhang.2019@phdcs.smu.edu.sg

Ivana Clairine Irsan新加坡管理大学新加坡
ivanairsan@smu.edu.sg

Ratnadira Widyasari
新加坡管理大学
新加坡
ratnadiraw.2020@phdcs.smu.edu.sg

周洋
新加坡管理大学
新加坡
zyang@smu.edu.sg

David Lo新加坡管理大学
新加坡 davidlo@smu.edu.sg

arXiv:2304.02514v1 [cs.SE] 5 Apr 2023

摘要——**使用 API 开发软件应用已成为常态。API 能帮助开发者更快地构建应用，因为他们无需重复造轮子。因此，开发者了解计划使用的 API 至关重要。开发者还应及时了解与 API 相关的信息更新。为此，开发者需要查找并跟踪所关注 API 的相关信息。然而，API 信息分散在各种在线来源中，难以手动跟踪。此外，识别与某个 API 相关的内容也并非易事。受这些挑战的启发，我们在本研究中推出了一款名为 APIHARVEST 的工具，旨在简化从各种在线来源查找 API 信息的过程。APIHARVEST 基于将 API 或库与各种在线来源关联起来的研究成果构建。它支持查找 GitHub 仓库、Stack Overflow 帖子、推文、YouTube 视频以及通用漏洞和暴露（CVE）条目中的 API 信息，并且可以扩展以支持其他来源。**

索引词—**API 挖掘、API 信息、多源**

## I. 引言

API 是现代软件开发中不可或缺的组成部分 [1]。API 使开发者能够专注于开发软件的主要功能，因为他们可以依赖 API 实现常见功能，从而缩短软件开发时间并提高开发者的生产力。鉴于 API 的重要性，开发者应充分了解计划使用的 API。他们可以从在线来源阅读 API 相关信息，例如 GitHub 上的相关代码、Stack Overflow 上的相关帖子，或 Twitter 上的相关推文。

虽然开发者可以自行查找 API 信息，但手动查找可能既繁琐又耗时。首先，API 信息分散在各种在线来源中。因此，开发者需要打开许多不同的网站，并逐一搜索。其次，网站中的搜索功能通常并非专门针对 API 搜索而设计。因此，开发者可能需要浏览大量搜索结果，才能找到与所搜索 API 相关的内容。每条内容可能包含代码或文本，其中提及了所搜索的 API。这些提及可能含义模糊，尤其是当 API 名称较为通用时（例如 read），它可能指不同库中的 API，也可能只是一个英文单词 读取。

现有研究致力于解决将与 API 或库相关的内容关联起来的挑战，同时处理 API [2]–[5] 的可能提及中存在的歧义。Luong 等人 [2] 提出了一种利用语义和语法分析来判断 Stack Overflow 讨论串是否与某个 API 相关的方法。Asyrofi 等人 [3] 开发了一种方法，通过对检索到的代码进行类型解析，实现更准确的 GitHub 代码搜索，并仅返回与特定 API 相关的代码。Zhang 等人 [4]研究了若干预训练模型在判断推文是否与某个库相关这一任务中的效果。他们发现 RoBERTa [6] 的表现最佳。Haryono 等人 [5] 研究了多种极端多标签学习（XML）技术，以从 CVE 条目中识别库，其中 LightXML [7] 取得了最佳表现。

尽管上述工作可用于识别 API 内容，但它们只能单独使用。因此，整体体验与同时打开多个网站并无太大区别。此外，由于某些内容的性质（例如推文），直接提及 API 的情况极为罕见，甚至完全不存在。对于这类内容，可能只有库链接方法可用。在这种情况下，使用现有的链接

方法查找 API 信息之前，还需要额外识别 API 所属的唯一库名称。

在本研究中，我们构建了一款名为 APIHARVEST 的工具，可在一个界面中整合现有的 API/库链接方法。该工具旨在简化从多种不同内容中查找 API 的流程。对于仅提供库链接方法的内容，APIHARVEST 可以自动将 API 映射到其对应的库，以利用该库链接方法。对于没有可用链接方法的内容，它也支持通用文本搜索。凭借这些功能，APIHARVEST 可以扩展以支持各种类型的内容。

本文其余部分的结构如下：第 II 节讨论预备知识，其中包括我们构建工具所依据的一些 API/库链接方法。第 III 节介绍所提出工具的详细信息。第 IV 节描述 APIHARVEST 的评估。最后，第 V 节和第 VI 节分别介绍相关工作和结论。

## II. 预备知识

我们介绍 APIHARVEST 所采用的 API/库链接方法。这些方法可用于判断某项内容是否与 API 或库相关。

### *A. ARSeek*

ARSeek [2] 是一种通过分析帖子中的自然语言段落和代码片段来确定 Stack Overflow 上的帖子是否在讨论特定 API 的方法。它有两个组件：DATYS+和 API 相关性分类器。第一个组件 DATYS+ 接收潜在帖子（即包含与给定 API 方法简单名称匹配的单词的帖子）以及 API 候选者（即与给定 API 方法具有相同简单名称的 API 方法）。然后，它计算给定 API 方法在该线程中被引用的置信度分数。第二个组件 API 相关性分类器将 Stack Overflow 线程中的自然语言段落和代码片段以及给定 API 方法的注释和实现代码转换为 API 相关性嵌入。然后使用这些嵌入来确定该线程讨论给定 API 方法的可能性。随后从这些嵌入中计算出一个分数。如果分数高于某个阈值，则认为该线程与给定的 API 方法相关。

### *B. AUSearch*

Asyrofi 等人 [3] 创建了 AUSearch，这是一种利用类型系统和 API 方法签名（由类名、简单名称和参数组成）来提高 GitHub 代码搜索精度的方法。[1] 该方法解决了 GitHub 代码示例中 API 调用含义不明确的问题，而这正是 GitHub 代码搜索的一项局限。通过实现类型解析，AUSearch 排除了不相关的代码示例，并显著提高了 GitHub 代码搜索查找包含该 API 的代码时的准确性。

[1]https://github.com/search

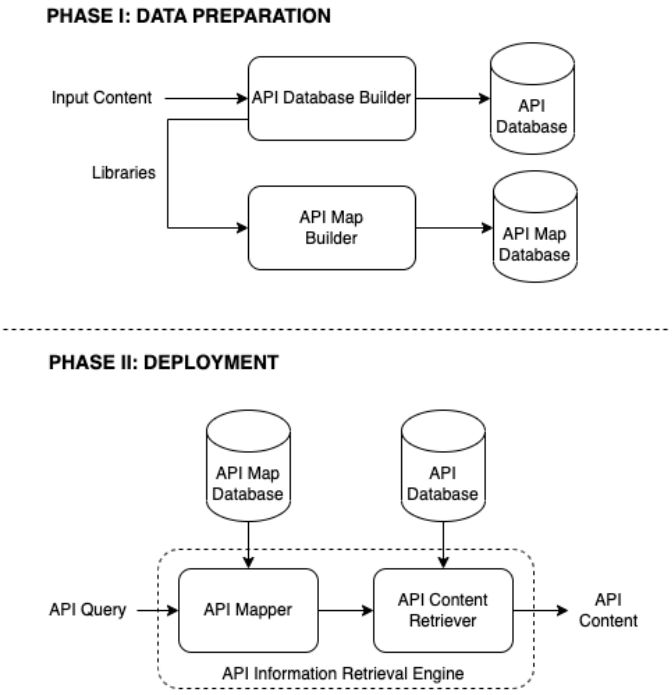



图 1. APIHARVEST 的架构

### *C.* 用于库相关推文的预训练模型

Zhang 等人 [4] 探索了多种预训练模型，以识别推文是否与某个库相关。他们测试了通用预训练模型（即 BERT [8]，RoBERTa [6],和 XLNet [9]）以及领域专用预训练模型（即 BERTweet [10] 和 BERTOverflow [11]）。他们针对库相关推文分类任务对这些模型进行了微调。他们发现，RoBERTa [6]，（一种经过稳健优化的 BERT） [8], 表现最佳，并超越了现有最先进方法。

### *D.* 用于识别 *CVE* 中库的 *XML* 技术

Haryono 等人 [5] 探索了多种极端多标签学习（XML）技术，用于识别 CVE 条目中提及的库。他们评估了五种传统模型：FastXML [12], DiSMEC [13], Parabel [14], Bonsai [15], 和 ExtremeText [16]。他们还评估了两种基于深度学习的模型：XML-CNN [17] 和 LightXML [7]。这些模型在一个 CVE 数据集上进行训练，该数据集的标签是所有 CVE 条目中提及的库。结果表明，Bonsai 和 LightXML 的效果最佳，其中 LightXML 表现尤为突出。

## III. 工具架构

### *A.* 概述

图 1 展示了 APIHARVEST 的整体架构。APIHARVEST 由三个主要组件构成：(1) *API* 数据库构建器、(2) *API* 映射构建器和 (3) *API* 信-息检索引擎。API 数据库构建器和 API 映射构建器属于数据准备 阶段，而

API 信息检索引擎属于部署阶段。

在数据准备阶段，APIHARVEST 分别使用*API* 数据库 和*API* 映射数据库 存储来自输入内容和库的 API 知识。API 数据库存储来自各种来源且已链接到 API 的内容，而 API 映射数据库存储从 API 到其库的映射。*API* 数据库构建器 使用现有的 API/库链接方法（如果有）为输入内容添加标签。只有当现有的 API/库链接方法将输入内容标记为相关内容时，它才会将输入内容存入 API 数据库。*API* 映射构建器 仅在存在库链接方法、可用于从输入内容中提取信息时，才添加从 API 到库的映射。在部署阶段，*API* 信息检索引擎 接收 API 查询，并利用 API 数据库和 API 映射数据库返回相关的 API 内容。我们将在以下小节中进一步详细介绍 APIHARVEST 的各个组件。

### *B. API* 数据库构建器

给定一段输入内容，API 数据库构建器首先识别其内容来源。如果存在支持该内容来源的 API 链接方法，API 数据库构建器就会对输入内容运行相应的 API 链接方法。如果输入内容被标记为与一个或多个 API 相关，就会连同指向所有相关 API 的链接一起存入 API 数据库。同样，如果只有支持该内容来源的库链接方法，API 数据库构建器就会运行相应的库链接方法，并将输入内容连同指向所有相关库的链接一起存入 API 数据库。另一方面，如果没有支持该内容来源的 API/库链接方法，输入内容就会在不带任何链接的情况下存入数据库。在上述所有情况下，API 数据库构建器都会运行一个脚本，将链接方法的输出格式转换为数据库格式。

### *C. API* 映射构建器

如果输入内容来自仅支持库链接方法的来源，API 数据库构建器会将库链接方法生成的库发送给 API 映射构建器。API 映射构建器会扫描所有这些库，并从每个库中提取所有公共 API（例如，使用 javap[2] 等工具）。然后，它会将库名称和公共 API 的名称存储在 API 映射数据库中。

### *D. API* 信息检索引擎

给定一个 API 查询，API 信息检索引擎首先将查询传递给 *API* 映射器。*API* 映射器会查询 API 映射数据库，以确定查询中 API 所属的库名称。随后，它将 API 名称和库名称都传递给 *API* 内容检索器。API 内容检索器会遍历 API 数据库中记录的所有可用 API 信息来源（例如 Stack Overflow、GitHub 等）。对于每个来源，它会使用 API 名称或库名称查询 API 数据库

[2]https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javap.html

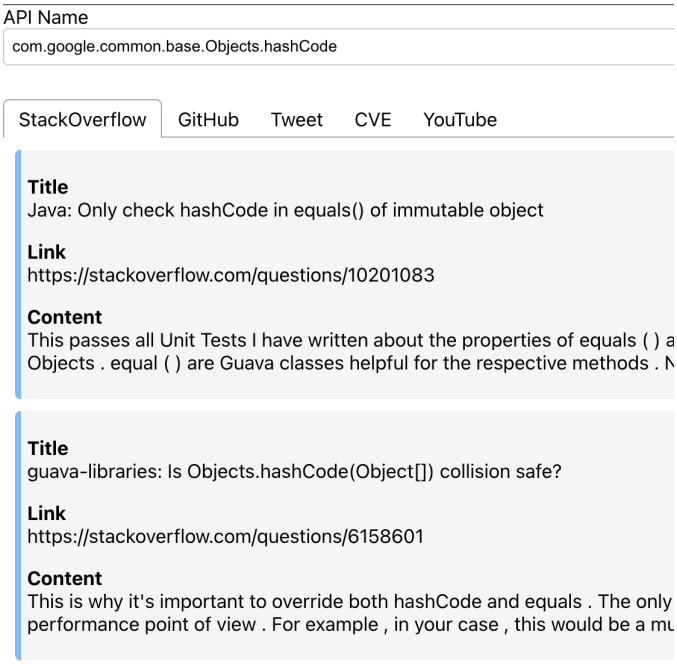



图2. APIHARVEST 的用户界面

（具体取决于该来源采用的链接方法）。如果采用 API 链接方法，则使用 API 名称查询数据库。同样，如果采用库链接方法，则使用库名称进行查询。另一方面，如果两种链接方法都未采用，API 内容检索器会利用经典的信息检索策略来检索相关内容。它会计算并对该来源中的所有内容与 API 名称之间的 BM25 分数进行排序 [18] 。API 内容检索器从每个来源获取相关内容后，会汇总所有内容并将其作为 API 内容返回。

图2展示了搜索与 API 方法 com.google.common.base.Object.hashCode() 相关内容时的用户界面。用户可以在搜索栏中输入 API 名称。用户停止输入后，API 名称会被发送到 API 信息检索引擎，由其检索相关内容并显示给用户。来自不同来源的内容会以选项卡形式分类：StackOverflow、GitHub、推文、CVE 和 YouTube。每个选项卡都包含来自相应来源的相关 API 内容（如有）。该图展示了 StackOverflow 中与给定 API 名称相关内容的示例。

## IV. 评估

### *A. API* 信息来源

为进行评估和构建原型，我们将以下信息来源整合到 APIHARVEST 中：Stack Overflow 帖子、GitHub 代码片段、推文、CVE 条目和 YouTube 视频。我们运行 ARSeek [2]和 AUSearch [3]，分别将 API 与 Stack Overflow 帖子和 GitHub 代码片段关联起来。

同样，我们运行 Zhang 等人提出的方法 [4]和 Haryono 等人提出的方法 [5]，分别将软件库与推文和 CVE 条目关联起来。我们使用他们研究中的数据集，并将测试集作为输入内容。我们还使用了他们在复现包中提供的实现。对于每种来源，我们都编写了一个脚本，将关联内容中的数据格式转换为数据库格式。请注意，如果要扩展 APIHARVEST 以支持新的 API/库链接方法，我们也需要编写这样的脚本。

对于 YouTube 视频，我们从 Java 标准库中随机选取了 100 个类。[3] 随后，我们查询 YouTube Search API，并获取该 API 返回的前 100 条视频元数据。接着，我们编写了一个脚本，将这些元数据导入 API 数据库。由于这里没有可用的 API/库链接方法，APIHARVEST 会根据 BM25 分数执行信息检索。

### *B.* 易用性

我们从易用性角度评估 APIHARVEST。我们不评估链接效果，因为该效果本质上取决于相应的 API/库链接方法。我们邀请了 3 名计算机科学博士生和 2 名计算机科学研究工程师评估 APIHARVEST 的易用性。他们的编程经验均超过 5 年。我们让他们使用 APIHARVEST，并请他们对以下三个方面进行评分：APIHARVEST 是否易于使用、是否实用，以及他们是否愿意在未来使用 APIHARVEST。评分范围为 1 至 5，分别表示强烈不同意、不同意、中立、同意和强烈同意。评估者认为 APIHARVEST 易于使用（5 分中得 4.8 分）、实用（5 分中得 4.6 分），并且愿意在未来使用 APIHARVEST（5 分中得 4 分）。

## V. 相关工作

### *A. API* 链接方法

API 链接方法旨在将 API 与相关内容关联起来。一类方法旨在将 API 与代码片段 [3], [19]–[21] 关联起来。为识别代码片段中的 API 提及，Baker [19] 通过迭代执行演绎式链接分析。COSTER [20] 和 STATTYPE [21] 都通过捕获并学习代码元素周围的词元，并将这些词元与此前遇到的相似词元相关联来实现。另一方面，AUSearch [3] 利用类型解析来确定代码中调用的 API。

另一类方法旨在将 API 与 Stack Overflow 帖子等软件工程文本关联起来 [2], [22]–[25]。在这一研究方向中，若干研究 [22]–[25] 采用了经典的信息检索技术和／或一些启发式方法。Miler 由 Bacchelli 等人开发，[24],利用字符串匹配和信息检索技术，将电子邮件与软件系统中的源代码实体关联起来，例如面向对象系统中的类以及过程式语言中的函数系统。Dagenais 和 Robillard [25] 采用过滤启发式方法，将软件支持渠道（如邮件列表和论坛）中提及的 API 关联起来。Luong 等人开发的 ARSeek [2],会对 Stack Overflow 帖子进行语义和语法分析，以确定该帖子是否与某个 API 相关。

### *B.* 库链接方法

库链接方法旨在将库与相关内容关联起来。一类方法可用于将库与推文等微博内容关联起来 [4], [26], [27]。Prasetyo 等人 [26]提出了一种方法，用于识别微博是否与软件相关。他们从推文的文本内容和 URL 中提取特征，随后使用支持向量机（SVM）训练分类器，以判断推文是否与软件相关。Sulistya 等人 [27] 提出了一种方法，利用 Stack Exchange 和 Stack Overflow 等内容丰富的平台中的知识，并将其应用于 Twitter 等其他平台。他们从 Stack Exchange 和 Stack Overflow 中提取词嵌入，并将这些嵌入作为分类器的特征，以识别与软件相关的推文。这两种方法都可通过筛选包含库名称的软件相关推文，来识别与库相关的推文。最近，Zhang 等人 [4]探讨了预训练模型识别库相关推文的能力，并发现这些模型比基线方法更有效。

另一类方法旨在将库与 CVE 条目关联起来 [5], [28]。Chen 等人 [28] 首次提出，自动将库与 CVE 条目中提及的漏洞相关联。他们将该问题建模为极端多标签学习（XML），其中库作为标签，并使用 FastXML [12] 识别 CVE 条目中使用的库。Haryono 等人 [5] 沿着这一方向继续研究，评估了多种 XML 算法，并发现 LightXML [7] 是完成此任务的最佳 XML 算法。

## VI. 结论与未来工作

API 信息散见于各种在线来源。将这些信息与 API 关联起来并非易事，因为有些 API 可能存在歧义，例如不同 API 可能具有相同的方法名称。尽管已有一些方法用于执行关联，但它们通常只针对特定来源设计。对于某些来源，也没有可用的 API 链接方法。针对这些问题，我们推出 APIHARVEST，这是一款用于从各种在线来源查找 API 信息并将其汇集到单一界面中的工具。APIHARVEST 同时采用 API 链接方法和库链接方法来查找 API 信息。它还支持从各种类型的来源查找 API 信息，无论是否有可用的 API/库链接方法。我们的初步评估表明，APIHARVEST 易于使用。未来，我们计划通过采用更多 API/库链接方法来增加支持的在线来源。我们已在 https://github.com/soarsmu/APIHarvest 发布 APIHARVEST 的源代码。

[3]https://docs.oracle.com/javase/7/docs/api/overview-summary.html

## 参考文献


[1] C. R. De Souza、D. Redmiles、L.-T. Cheng、D. Millen 和 J. Patterson，“良好的软件实践如何阻碍协作：API 在软件开发中的多重角色”，发表于第 *12* 届 *ACM SIGSOFT* 软件工程基础国际研讨会论文集，2004 年，第 221–230 页。[2] K. Luong、M. Hadi、F. Thung、F. Fard 和 D. Lo，“Arseek：利用代码和 Stack Overflow 讨论识别 API 资源”，发表于 第 *30* 届 *IEEE/ACM* 国际程序理解会议论文集，2022 年，第 331–342 页。[3] M. H. Asyrofi、F. Thung、D. Lo 和 L. Jiang，“Ausearch：通过类型解析在 GitHub 仓库中准确搜索 API 用法”，发表于*2020* 年 *IEEE* 第 *27* 届软件分析、演化与再工程国际会议（*SANER*）。IEEE，2020 年，第 637–641 页。[4] T. Zhang 、 D. P. Chandrasekaran 、 F. Thung 和 D. Lo，“ 推文中的库识别基准测试”，发表于*2022* 年 *IEEE/ACM* 第 *30* 届国际程序理解会议（*ICPC*），2022 年，第 343–353 页。[5] S. A. Haryono、H. J. Kang、A. Sharma、A. Sharma、A. Santosa、A. M. Yi 和 D. Lo，“从漏洞数据中自动识别库：我们能做得更好吗？”，2022 年。[6] Y. Liu、M. Ott、N. Goyal、J. Du、M. Joshi、D. Chen、O. Levy、M. Lewis、L. Zettlemoyer 和 V. Stoyanov，“RoBERTa：一种稳健优化的 BERT 预训练方法”，*arXiv* 预印本 *arXiv:1907.11692*，2019 年。[7] T. Jiang、D. Wang、L. Sun、H. Yang、Z. Zhao 和 F. Zhuang，“LightXML：采用动态负采样的 Transformer，用于高性能极端多标签文本分类”，发表于*AAAI* 人工智能会议论文集，第 35 卷，第 9 期，2021 年，第 7987–7994 页。[8] J. Devlin、M.-W. Chang、K. Lee 和 K. Toutanova，“BERT：用于语言理解的深度双向 Transformer 预训练”，*arXiv* 预印本 *arXiv:1810.04805*，2018 年。[9] Z. Yang、Z. Dai、Y. Yang、J. Carbonell、R. R. Salakhutdinov 和 Q. V. Le，“XLNet：用于语言理解的广义自回归预训练”，神经信息处理系统进展，第 32 卷，2019 年。[10] D. Q. Nguyen、T. Vu 和 A. T. Nguyen，“BERTweet：一种用于英语推文的预训练语言模型”，*arXiv* 预印本 *arXiv:2005 10200*，2020。。[11] J. Tabassum、M. Maddela、W. Xu 和 A. Ritter，“Stack Overflow 中的代码与命名实体识别”，*arXiv* 预印本 *arXiv:2005.01634*，2020 年。[12] Y. Prabhu 和 M. Varma，“FastXML：一种快速、准确且稳定的极端多标签学习树分类器”，发表于第 *20* 届 *ACM SIGKDD* 知识发现与数据挖掘国际会议论文集，2014 年，第 263–272 页。[13] R. Babbar 和 B. Sch¨olkopf，“Dismec：用于极端多标签分类的分布式稀疏机器”，发表于第十届 *ACM* 论文集国际网页搜索与数据挖掘会议，2017，第 721–729 页。[14] Y. Prabhu、A. Kag、S. Harsola、R. Agrawal 和 M. Varma，《Parabel：面向极端分类的分区标签树及其在动态搜索广告中的应用》，载于*2018* 年万维网大会论文集，2018，第 993–1002 页。[15] S. Khandagale、H. Xiao 和 R. Babbar，《Bonsai：用于极端多标签分类的多样化浅层树》，机器学习，第 109 卷，第 11 期，第 2099–2119 页，2020。[16] M. Wydmuch、K. Jasinska、M. Kuznetsov、R. Busa-Fekete 和 K. Dembczynski，《层次 softmax 在极端多标签分类中的一种无遗憾推广》，神经信息处理系统进展，第 31 卷，2018。[17] J. Liu、W.-C. Chang、Y. Wu 和 Y. Yang，《用于极端多标签文本分类的深度学习》，载于第 *40* 届 *ACM SIGIR* 信息检索研究与开发国际会议论文集，2017，第 115–124 页。[18] S. Robertson、H. Zaragoza等，《概率相关性框架：BM25 及其他方法》，信息检索基础与趋势®，第 3 卷，第 4 期，第 333–389 页，2009。[19] S. Subramanian、L. Inozemtseva 和 R. Holmes，《实时 API 文档》，载于*ICSE*，2014。[20] C. K. Saifullah、M. Asaduzzaman 和 C. K. Roy，《从示例中学习，在代码片段中查找 API 元素的完全限定名称》，载于*ASE*。IEEE，2019。[21] H. Phan、H. A. Nguyen、N. M. Tran、L. H. Truong、A. T. Nguyen 和 T. N. Nguyen，《在线论坛代码片段中 API 完全限定名称的统计学习》，载于*ICSE*。IEEE，2018。[22] G. Antoniol、G. Canfora、G. Casazza、A. De Lucia 和 E. Merlo，《恢复代码与文档之间的可追溯性链接》，*TSE*，第 28 卷，第 10 期，2002。[23] A. Marcus 和 J. I. Maletic，《使用潜在语义索引恢复文档与源代码之间的可追溯性链接》，载于*ICSE*。IEEE，2003。[24] A. Bacchelli、M. Lanza 和 R. Robbes，《关联电子邮件与源代码工件》，载于*ICSE*，2010。[25] B. Dagenais 和 M. P. Robillard，《恢复 API 与其学习资源之间的可追溯性链接》，载于*ICSE*。IEEE，2012。[26] P. K. Prasetyo、D. Lo、P. Achananuparp、Y. Tian 和 E.-P. Lim，《软件相关微博的自动分类》，载于*2012* 年第 *28* 届 *IEEE* 软件维护国际会议（*ICSM*）。IEEE，2012，第 596–599 页。[27] A. Sulistya、G. A. A. Prana、A. Sharma、D. Lo 和 C. Treude，《Sieve：通过跨平台分析帮助开发者从糟粕中筛选精华》，实证软件工程，第 25 卷，第 1 期，第 996–1030 页，2020。[28] Y. Chen、A. E. Santosa、A. Sharma 和 D. Lo，《基于漏洞数据自动识别库》，载于*ACM/IEEE* 第 *42* 届软件工程国际会议论文集：软件工程实践，2020，第 90–99 页。