基于大语言模型的开源许可证冲突检测与修复研究

何思洋 曾华正
https://github.com/huazZeng/license_llm_public

摘要

本研究旨在提出并实现一个开源许可证冲突检测修复系统，用于检测开源项目内的许可证冲突情况并且给出修复建议。基于大语言模型的强大理解能力，系统实现了对开源许可证内容的自动化识别，以便于后续检测冲突。我们使用 GPT-4 模型进行了一系列实验，并通过检索增强生成（RAG）缓解长上下文中模型检索能力下降的问题，通过 CoT、self-refine 等方法减少典型逻辑错误、提高识别精度。除此以外，本文构建了许可证相关的数据集，同时提出了较于其他许可证相关工具精度更高的项目扫描方法，并且在现实项目中进行了初步实验，证明了系统在开源许可证冲突检测和修复上的有效性。

关键词

开源软件生态 开源许可证 大语言模型

1. 引言

开源软件（OSS）[1]在现代软件开发和技术创新中扮演着至关重要的角色。通过共享源代码，开源软件推动了知识共享、技术进步和社区合作，同时为企业和个人用户提供了低成本的软件解决方案，减少软件开发和采购的支出。然而随着开源软件的广泛采用和扩散，许可证不兼容问题开始显现，且近年来逐渐引起更多人的重视。曾有对 1846 个 GitHub 项目的实证研究显示，72.91%的调查项目遭受许可证不兼容问题[2]。现实中的多个案例最终的判决结果显示，开源许可证是受法律保护的，对于开源软件用户来说，使用免费得到的开源代码，如果违反了原始许可证协议，会被法律判定为侵犯开源软件作者著作权。

近几年迅速发展的大语言模型经过大规模数据训练展现出强大的语义理解能力、互动性以及知识推理能力。考虑到许可证文本中包含大量自然语言表述，本课题利用 GPT-4 模型[3]在自然语言处理上的优势，基于 RAG[4, 5],CoT[6,7], self-refine [8] 技术实现对许可证条款实体的自动化识别和高精度理解，同时对各项目建立许可证依赖层次结果，从而更好地检测并修复许可证不兼容问题。本课题的研究成果可以有效减少开源项目中的许可证冲突现象，减少侵权的风险，促进开源软件的合法和合规使用，提高开源项目的可维护性和可扩展性。

2. 相关工作

2.1 许可证识别

为了便于对许可证进行分析，第一步是从文本或代码文件中自动识别许可证。Tuunanen 等人[9]开发了 ASLA，基于正则表达式来识别许可证。ASLA 表现出与两个开源许可证分析器（即 OSLC [10]和 FOSSology [11,12]）相竞争的性能。然而，他们没有报告那些未准备好正则表达式的未知许可证。为了解决这个问题，German 等人[13]设计了 Ninka，首先将许可证声明分解成句子，然后将句子标记与每个许可证已经手动识别的句子标记(即许可证规则)进行匹配以识别许可证。如果没有匹配到许可证，Ninka 将报告一个未知许可证。为了帮助生成 Ninka 报告为未知的许可证的许可证规则，Higashi 等人[14]利用分层聚类将未知许可证分组成具有单一许可证的文件群集。此外，Vendome 等人[15]提出了一种基于机器学习的方法来识别附加到许可证的异常。Kapitsaki 和 Paschalides[16]设计了 FOSS-LTE 来从许可证文本中识别许可证条款。ScanCode[17]是一个工具，通过扫描软件项目中的第三方包，帮助发现并标准化许可信息、依赖关系和来源等数据，简化许可证合规性管理。

由于我们的工具不仅需要能识别 SPDX 许可证，还需要能够提取相关的许可证文本，所以我们选取了我们的工作使用 Scancode 来识别每个文件中声明的许可证。

2.2 许可证条款实体理解

过去最先进的许可证自动化理解工具之一 FOSS-LTE[18]利用 LDA 无监督机器学习的方式，结合人工分类的训练集以及许可证语料库，建立了一个从许可证条款实体到标准许可证条款表述的映射。较前沿的许可证冲突检测工具 Lidetector[19]主要思路是将许可证分为句（term）后，进一步标记出句内有实义的词组实体（entity）：以人工标记作为训练集输入，利用机器学习训练出合适的概率模型，用于后续自动标记实体。再利用预训练的小型 NLP 模型 Probabilistic Context-Free Grammar (PCFG) model 进行语法以及情感分析。除此以外还可运用两种常见的自然语言处理技术。一是利用优化后的 word2vec 模型[20]计算语义相似度来建立映射；二是规则匹配，它通过预定义一组关键词的模式来实现实体提取[13]。

可见过去大多数相关工作利用机器学习，结合 word2vec 等预定义的单词嵌入模型或其他小型 NLP 模型来理解许可证条款实体并映射到标准许可证条款表述。

2.3 许可证冲突检测

Kapitsaki 等在许可证冲突检测工具构造 SPDX-VT[1]的过程中，人工建立了包含 19 个 SPDX 标准许可证的兼容性有向图，运用 BFS 算法检测项目中各许可

证是否兼容。 German 等人提出的 Ninka[13]利用包括“等价短语替换-句子过滤-token 匹配”在内的句子匹配法识别许可证，结合 tldrlegal[21]上记录的各 SPDX 许可证所规定的权利与义务，便可完成许可证的理解，从而进行冲突检测。Librariesio[22]也聚焦于检测一组 SPDX 许可证之间的兼容性。

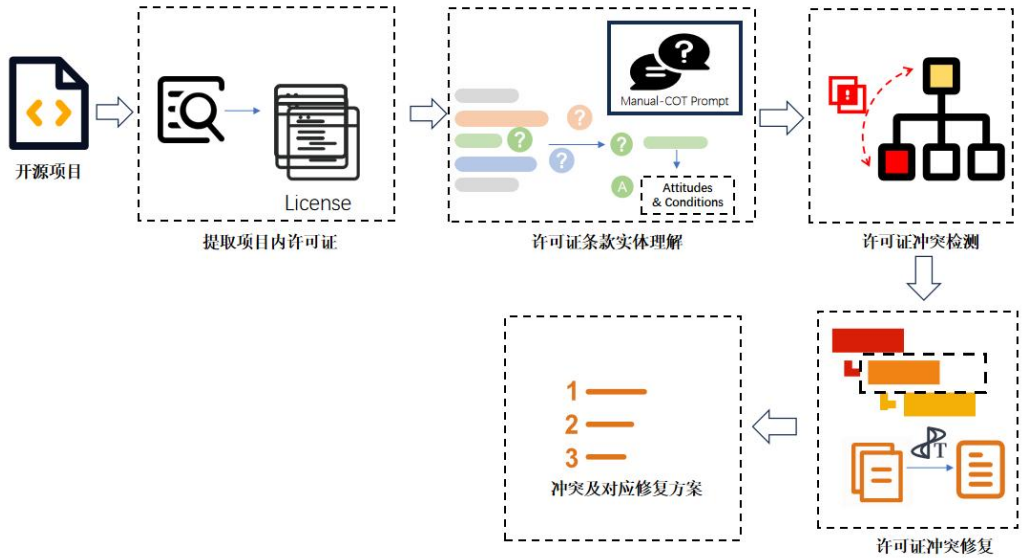
可见过去部分相关工作着眼于检测人工标记后的 SPDX 标准许可证之间的不兼容问题，一定程度上忽视了自定义许可证的存在。

除此以外，目前的许可证冲突检测工具在检测许可证不兼容问题时倾向于将所有许可证置于平等位置，比如 SPDX-VT[23]罗列所有涉及的许可证，而后判断他们在兼容有向图上是否弱连通；或是像 Lidectector[19]一样将所有第三方组件许可证置于同等位置，只将项目许可证置于它们上层，这在一定程度上忽视了项目中的层次架构。

2.4 许可证冲突修复

Kapitsaki 等人提出了 FindOSSLicense[24]来推荐开源软件许可证。它通过一系列提问了解用户需求，以及类似用户或类似项目使用的许可证，同时考虑有向兼容图中刻画的兼容关系提出了一种混合方法。然而，许可证图仅涵盖了少量许可证，未在许可证图中涵盖的许可证仅可被标记为“警告”，即候选许可证较少。Liu 等人[25]观察到软件更改可能导致许可证更新，提出在软件更改存在的情况下预测源代码文件级别的候选许可证。然而，这一许可证预测工具仅适用于软件更改，其应用范围受到限制。Liresolver[26]在修复许可证冲突方面，优先考虑改用 SPDX 官方许可证，若都不能兼容地嵌入原项目则考虑自定义许可证，同时最大程度地使得对原许可证的修改降到最低。这一解决方案相对以往相关工作有了一定的改进。

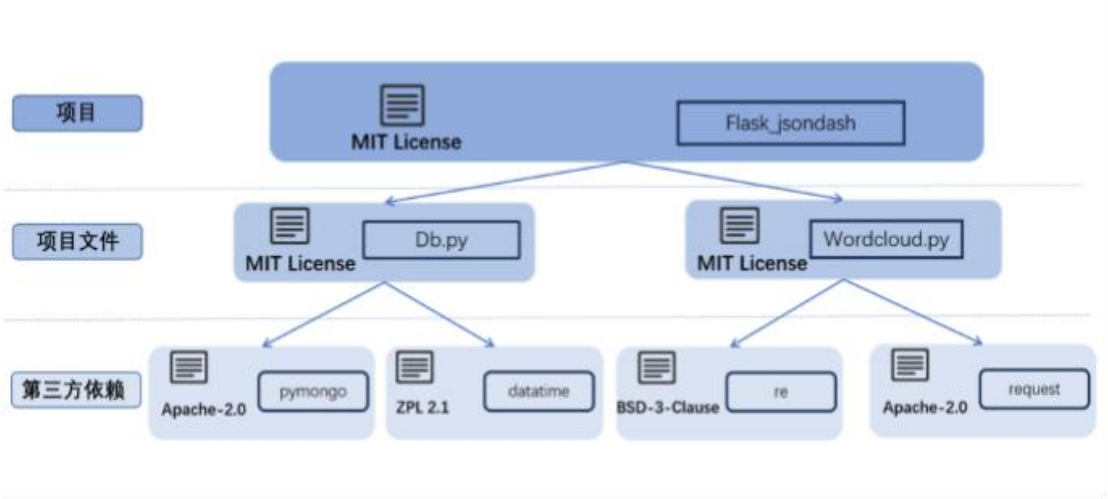
3. 方法设计



本工具主要包括四个核心模块：项目内许可证提取、许可证条款实体理解、许可证冲突检测以及许可证冲突修复。最终的输出结果包括项目中的许可证冲突情况及相应的修复方案。

3.1 项目内许可证提取

本工具首先对项目内的许可证进行提取。项目中的许可证主要分为两类：文件内部的许可证文本或引用，以及第三方组件的许可证。开源项目通常包含多个文件许可证和第三方组件许可证，这些许可证并非处于同一层级，而是呈现出层次化的父子关系，如图所示。



许可证冲突常发生在父节点许可证与子节点许可证之间。为精准识别不兼容问题，提取项目的许可证层次结构至关重要。我们基于软件包的抽象语法树 (AST) 进行入口层次分析，提取软件包的文件层次结构及其依赖关系，为后续分析奠定基础。

本工具结合 Scancode[17] 和动态扫描方法进行许可证提取。Scancode 可从源代码文件中识别许可证文本，支持 SPDX 标准，具备高准确性、丰富的匹配规则和完善的生态系统，相较于 Ninka[13] 等工具表现更优。通过大量开源项目测试，Scancode 在复杂的代码库中保持了稳定的性能和较低的误报率。

此外，通过动态扫描提取第三方组件的许可证信息。我们首先配置项目运行环境以确保项目可执行，然后使用包管理工具获取第三方组件及其许可证信息，确保提取的许可证与实际使用版本一致，即使第三方组件在后续版本中更换了许可证，扫描结果也不受影响。开发者仅需在项目环境中执行本工具，即可高效、准确地获取所有第三方组件的许可证信息。

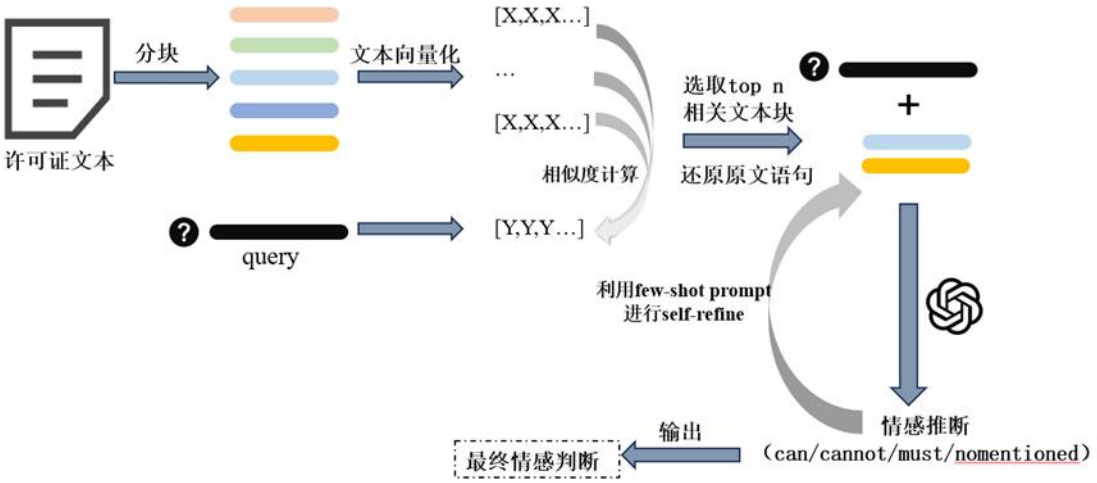
3.2 许可证条款实体理解

该部分实验的基本思路是将许可证长文本去除冗余语句之后进行基于滑动窗口的切割处理，把较大的文本块切割成小块，然后利用较小的模型将这些文本

块转化为向量表示。在测试该许可证对于某一 query 的情感判断结果时，首先将 query 中的条款行为基于同一嵌入模型转化为向量，并与已转化的文本块进行余弦相似度匹配，从中选出最相关的，即获得相似度 top n 的文本块（n 与原文本长度相关）。再将文本块还原为原文完整语句并去重[4,5]。

接着，将这些相关的文本与原始查询一起输入大模型，以生成初步的情感判断。为进一步提高准确性，实验还引入了自我优化机制。具体来说，基于第一次模型的输出，结合少量的示例（few-shot prompt[6,7]），再一次将输入提供给大模型，让其通过自我修正[8]来纠正初步判断中的错误，尤其是一些典型逻辑问题，例如过度推理、误解转折式逻辑等等。

最终，经过自我优化后的大模型会输出最终的情感判断结果。该方法的核心意义在于，仅将与 query 强相关的文本输入到大模型中，而不是将整个许可证长文本都输入。这种方式有效避免了在处理长文本时可能出现的检索能力下降的问题，同时提高了计算效率和情感判断的准确性。因此，通过这种方法，能够更精确地从海量许可证文本中提取出有价值的信息，为法律条款的理解与应用提供了强有力的技术支持。



3.3 许可证冲突解决与修复

在提取了许可证层次结构树及每个许可证对 23 类条款行为的态度和附加条件后，若树中每个节点的许可证均比其子节点更具限制性，则不存在许可证冲突。因此，我们找出所有比父节点更严格或不兼容的许可证，确定冲突点。

由于第三方组件许可证不可更改，我们采用自底向上的方式，将当前节点的许可证限制传递给父节点。如果父节点满足此限制，则继续向上传递；若不满足，则进行修复并继续传递修复后的许可证限制，从而消除所有冲突。若产生第三方组件之间的许可证冲突，则无法修复，需要更换第三方组件。

为修复冲突，我们预先构建了 SPDX 标准许可证数据库。根据父节点和子节点在 23 类条款行为上的态度，定义了“许可证态度向量可行域”。若有多个 SPDX 许可证符合该可行域，则通过相似度分析选择与原许可证最相似的候选许可证；若无符合的 SPDX 许可证，则建议用户修改原许可证条款并提供修改建议。

4. 实验设置与结果

4.1 许可证分布

为了分析文件内许可证在开源项目中的分布情况，我们在 63 个开源项目中进行了扫描并提取了相关许可证信息, 共提取到 189 个许可证。经分析发现, 69.8% 的项目内许可证数量为 1-2 个，因此许可证冲突更多发生在第三方组件许可证与文件内许可证之间。此外，我们对提取到的 9190 条许可证相关文本进行了分析，发现了一些被误识别为许可证的文本。为此，我们为工具添加了过滤机制，从而避免了过度扫描。

4.2 许可证提取

相较于其它类似工具 LiResolver、LiDetector，在文件许可证提取上，我们采用的 Scancode 工具相较于 ninka 具有更高的准确率，在由 96 个 SPDX 许可证构成的数据集内，Scancode 达到了的 90.4%的准确率，而 ninka 只有 22.9%的准确率；Scancode 的大部分错误是由许可证版本变动引起的，而我们的调查发现，大部分许可证版本变动对条款的影响很有限，这体现了 Scancode 的强大能力。

同时，在第三方组件许可证提取上，我们使用的动态扫描方法相较于他们使用的静态扫描方法具有着更高的准确度，在 7 个现实大型项目的测试中，我们发现静态扫描方法的误报率甚至达到 60.5%，主要是因为静态扫描无法辨别自建组件和第三方组件；而动态扫描方法在配置好项目运行环境后，准确率为 100%，并且能够获取最精确的许可证，不受版本变动影响。

4.3 许可证理解

本实验设置了 23 类条款行为：参考以往相关工作 Liresolver[19], Lidetector[26], 将许可证中主要涉及的条款行为分为 23 类。其中前 11 类涉及用户权利，例如“adding other licenses with the software”；后 12 类涉及用户义务或限制，例如“giving explicit credit or acknowledgement to the author with the software”。
涉及四种情感态度，分别为：“can”（表示允许用户进行该操作），“cannot”（禁止用户进行该操作），“must”（规定用户必须执行该操作），以及“nomentioned”（未提及相关内容）

为了便于后续不兼容性的判断，在此对“nomentioned”进行语境下的同义转化：23 类条款行为中前 11 类为权利，若对这 11 类条款行为之一的态度为“nomentioned”，即未提及该行为，可以理解为未赋予该权利，故前 11 类中“nomentioned”可转化为“cannot”；后 12 类为义务，若对这 10 类条款行为之一的态度为“nomentioned”，即未提及该行为，可以理解为未规定该义务，故后 12 类中“nomentioned”可转化为“can”。

许可证长文本切割阶段，经试验，滑动窗口大小取近似 200 字符，窗口之间有至多 50 字符重叠，窗口首末保留完整单词。对于文本块的嵌入模型，基于命中率对比 OpenAI Embedding、BAAI/bge-large-en、sentence-t5-large 三种常见的嵌入模型，最终选择 sentence-transformers/sentence-t5-large 作为本次任务的文本嵌入模型。选择与 query 相似度最高的 top n 文本块时，n 的值依据原文本长度动态选取：

$$n = \begin{cases} 5, & \text{if } 0.05 \times \text{文本总块数} < 5 \\ \text{Int}(0.05 \times \text{文本总块数}), & \text{if } 5 \leq 0.05 \times \text{文本总块数} \leq 15 \\ 15, & \text{if } 0.05 \times \text{文本总块数} > 15 \end{cases}$$

本次实验使用 gpt-4[3]作为主要模型，self-refine[8]阶段针对 4 个初次情感判断错误率较高的 query，将各自情感推断的典型逻辑错误案例整合为 few-shot prompt，与第一次返回的 answer 一起再次输入大模型，从而获得模型自我反思及纠正之后的答案，提升情感推断准确率。

该步骤评估时所用的许可证文本数据集均源自 FOSSA 开源许可证管理平台。对 query4、query18、query19、query22 分别进行 self-refine，各许可证对它们 self-refine 前后平均准确率对比如下：

	Query 4	**Query 18**	**Query 19**	**Query 22**
Before self-refine	38.23%	52.94%	70.59%	41.18%
After self-refine	55.88%	88.23%	94.12%	73.5%

可见 self-refine 模块可以有效地促使模型反思自己的逻辑错误并提升情感推断准确率。

在 40 个许可证文本上进行测试，最终在 23 个 query 上的平均情感推断准确率为 87.82%。

4.4 许可证冲突检测与修复

我们在 35 个 Pypi 第三方组件项目中进行了工具的测试，检测到 12 个项目存在许可证不兼容的情况，其中共有 40 个冲突节点。工具为这 40 个冲突节点给出了修复方案，结果验证，我们的修复方案可以修复上述所有冲突。

5. 总结与展望

本课题提出了一种基于大语言模型的许可证冲突检测修复工具。工具通过大语言模型实现了对许可证文本的准确理解，同时使用更加精确的许可证提取方法，显著提高了开源项目许可证冲突情况检测与修复的效率。实验结果表明，工具在许可证提取，许可证理解方面的准确度都超过了其它同类工具，展现了工具的高准确度。

未来的工作将集中在拓展工具在更复杂环境下的使用，包括探索更复杂的许可证分布情况，丰富许可证条款数量，添加对许可证特殊文件的额外处理等，以提高工具在复杂环境下的有效性。

参考文献

[1] Opensource. 2021. What is open source? Retrieved 27th Sep 2021 from https://opensource.com/resources/what-open-source

[2] A. Hemel, K. T. Kalleberg, R. Vermaas, and E. Dolstra. 2011. Finding software license violations through binary code clone detection. In Proceedings of the 8th Working Conference on Mining Software Repositories.

[3] OpenAI. "GPT-4: A Report on the Capabilities and Improvements of the Latest Model." OpenAI, 2023, www.openai.com/research.

[4] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv:2005.11401 [cs.CL]. https://arxiv.org/abs/2005.11401

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, Haofen Wang.2023. Retrieval-Augmented Generation for Large Language Models: A Survey.arXiv:2312.10997. https://doi.org/10.48550/arXiv.2312.10997

[6] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv:2201.11903 [cs.CL]. https://arxiv.org/abs/2201.11903

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, Denny Zhou.2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903v6.https://arxiv.org/abs/2201.11903

[8] Madaan, Aman, et al. "Self-Refine: Iterative Refinement with Self-Feedback." arXiv, 2023, https://doi.org/10.48550/arXiv.2303.17651

[9] T. Tuunanen, J. Koskinen, and T. Kärkkäinen. 2009. Automated software license analysis. Automated Software Engineering 16, 3 (2009), 455 – 490.

[10] daxlec, devil_moon, lrontyne, sjkaaria, and villoks. 2007. OSLC. Retrieved March 21, 2023 from https://sourceforge.net/projects/oslc/

[11] Linux Foundation. 2017. Fossology. Retrieved March 21, 2023 from https://www.fossology.org/

[12] Robert Gobeille. 2008. The fossology project. In Proceedings of the 2008 international working conference on Mining software repositories. 47 – 50.

[13] Daniel M. German, Yuki Manabe, and Katsuro Inoue. 2010. A sentence-matching method for automatic license identification of source code files. In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE '10). Association for Computing Machinery, New York, NY, USA, 437 – 446. https://doi.org/10.1145/1858996.1859088
[14] Y. Higashi, Y. Manabe, and M. Ohira. 2016. Clustering OSS license statements toward automatic generation of license rules. In Proceedings of the 7th International Workshop on Empirical Software Engineering in Practice. 30 – 35.
[15] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, and D. Poshyvanyk. 2017. Machine learning-based detection of open source license exceptions. In Proceedings of the IEEE/ACM 39th International Conference on Software Engineering. 118 – 129.
[16] G. M Kapitsaki and D. Paschalides. 2017. Identifying terms in open source software license texts. In Proceedings of the 24th Asia-Pacific Software Engineering Conference. 540 – 545.
[17] Scancode-toolkit , https://github.com/aboutcode-org/scancode-toolkit.
[18] Georgia M. Kapitsaki and Demetris Paschalides. 2017. Identifying terms in open source software license texts. In Proceedigns of the 24th Asia-Pacific Software Engineering Conference. 540 – 545. https://doi.org/10.1109/APSEC.2017.62
[19] Sihan Xu, Ya Gao, Lingling Fan, Zheli Liu, Yang Liu, and Hua Ji. 2023. LiDetector: License Incompatibility Detection for Open Source Software. ACM Trans. Softw. Eng. Methodol. 32, 1, Article 22 (January 2023), 28 pages. https://doi.org/10.1145/3518994
[20] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.’ s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014). https://doi.org/10.48550/arXiv.1402.3722
[21] kevin. 2012. Software Licenses in Plain English. https://tldrlegal.com/.
[22] librariesio. 2015. Check compatibility between different SPDX licenses for check-ing dependency license
[23] Georgia M. Kapitsaki, Frederik Kramer, and Nikolaos D. Tselikas. 2017. Automat-ing the license compatibility process in open source software with SPDX. Journal of Systems and Software (2017), 386 – 401. https://doi.org/10.1016/j.jss.2016.06.064
[24] Georgia M Kapitsaki and Georgia Charalambous. 2021. Modeling and recom-Mending open source licenses with findOSSLicense. IEEE Transactions on SoftwareEngineering 47, 5 (2021), 919 – 935. https://doi.org/10.1109/tse.2019.2909021
[25] Xiaoyu Liu, LiGuo Huang, Jidong Ge, and Vincent Ng. 2019. Predicting licenses for changed source code. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 686 – 697. https://doi.org/10.1109/ASE.2019.00070
[26] Sihan Xu, Ya Gao, Lingling Fan, Linyu Li, Xiangrui Cai, and Zheli Liu. 2023. LiResolver: License Incompatibility Resolution for Open Source Software. In Proceedings of the 32nd ACM SIGSOFT International Symposium on Software

Testing and Analysis (ISSTA 2023). Association for Computing Machinery, New York, NY, USA, 652 – 663. https://doi.org/10.1145/3597926.3598085

致谢

感谢陈碧欢老师、吴苏晟学长和黄凯峰学长的指导，让我们对科研有了第一次清晰的认识和完整的参与。希望未来能在开源软件生态领域继续努力，做出积极贡献!