计算机科学
COMPUTER SCIENCE

一种基于集成学习的开源许可证检测与兼容性判断的方法

白江浩, 朴勇

引用本文

白江浩, 朴勇. 一种基于集成学习的开源许可证检测与兼容性判断的方法[J]. 计算机科学, 2024, 51(12): 79-86.

BAI Jianghao, PIAO Yong. Ensemble Learning Based Open Source License Detection and Compatibility Assessment [J]. Computer Science, 2024, 51(12): 79-86.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于大语言模型的电力知识库智能问答系统构建与评价

Construction and Evaluation of Intelligent Question Answering System for Electric Power Knowledge Base Based on Large Language Model

计算机科学, 2024, 51(12): 286-292. https://doi.org/10.11896/jsjkx.240300104

基于大语言模型的移动应用可访问性增强方法

Large Language Model-based Method for Mobile App Accessibility Enhancement

计算机科学, 2024, 51(12): 223-233. https://doi.org/10.11896/jsjkx.240400077

文本人格检测研究综述

Study on Text-based Personality Detection—A Review

计算机科学, 2024, 51(12): 209-222. https://doi.org/10.11896/jsjkx.240500071

汽车验证电控系统中的测试用例自动生成方法

Automatic Test Case Generation Method for Automotive Electronic Control System Verification

计算机科学, 2024, 51(12): 63-70. https://doi.org/10.11896/jsjkx.240900093

面向大语言模型的推荐系统综述

Survey of Recommender Systems for Large Language Models

计算机科学, 2024, 51(11A): 240800111-11. https://doi.org/10.11896/jsjkx.240800111

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231200100

一种基于集成学习的开源许可证检测与兼容性判断的方法

白江浩　朴　勇
大连理工大学软件学院　辽宁 大连 116620
(ssdutbjh@163.com)

摘　要　软件供应链的安全性和可靠性对软件质量和演化有重要影响,而软件组件的许可证分析正是软件供应链中不可或缺的一环。开源许可证约束着开源软件的使用条件,以保护知识产权并维持开源软件的长远发展。为了避免法律风险与财产损失,识别开源软件许可证并判断开源许可证之间的兼容性至关重要。文中提出了基于集成学习的开源许可证的检测方法与依据兼容性的许可证推荐方法。具体来讲,提出了以基于大语言模型的集成学习为主,以规则匹配为辅的方法来进行开源许可证检测,并依据需求与有向图算法来完成许可证的兼容性判断与推荐。实验表明,相比于传统方法,该方法在更少的维护成本与高扩展性的优势下具有更好的检测效果,也能够有效地检测出兼容性并推荐结果。

关键词:大语言模型;集成学习;开源许可证;句向量相似度;兼容性判断
中图分类号　TP391

Ensemble Learning Based Open Source License Detection and Compatibility Assessment

BAI Jianghao and PIAO Yong
School of Software Engineering,Dalian University of Technology,Dalian,Liaoning 116620,China

Abstract　The quality and evolution of software are profoundly influenced by the security and reliability of the software supply chain. An essential element of this chain is the analysis of licenses associated with different software components. Open source licenses play a vital role in defining conditions for using open source software,safeguarding intellectual property,and ensuring the sustained development of open source projects. To mitigate legal risks and protect against property losses,it is imperative to accurately identify open source software licenses and assess their compatibility. In this paper,we propose an innovative method for detecting open source licenses using ensemble learning,complemented by a recommendation system based on compatibility. Our main approach leverages ensemble learning techniques,particularly emphasizing the use of large language models. To bolster the accuracy of open source license detection,this methodology is augmented with rule matching. Subsequently,compatibility assessments and license recommendations are derived using directed graph algorithms. Experimental results validate the effectiveness of our method,showcasing not only reduced maintenance costs and heightened scalability but also superior detection performance in comparison to traditional methods. The proposed approach excels in identifying compatibility issues and provides dependable recommendations,thereby contributing to a more secure and reliable software supply chain.

Keywords　Large language model,Ensemble learning,Open source license,Sentence vector similarity,Compatibility assessment

1　引言

开源组件通常包含一种或多种类型的开源许可证,它阐述了基于该开源组件进行软件开发时需要遵循的条款和规定[1]。开源组件的许可证分析有助于开发团队了解使用的每个组件的许可证限制,从而避免潜在的法律问题,尊重知识产权,是软件供应链中不可或缺的一环[2]。

开源许可证授权用户在承认软件原作者的著作权的前提下,对软件源代码进行使用、修改、商业化等,以确保开源软件能够被软件开发者合法自由使用和共享。在使用开源软件的时候,为了避免引起法律纠纷,有几个方面需要考量[3-4],例如开源要求、修改声明、专利许可、许可证兼容性、商标使用、网络传播等。以不合规的方式使用开源软件,会违反开源许可证中的条款,带来法律问题[5]。如微软 Windows7 就因为其下载软件使用了"GPLv2"的 ImageMaster 工具代码,最终引发了开源法律问题。许可证问题非常普遍,Wolter 等[6]发现在 github 上很多开发者没有声明其源代码中包含的开源许可证。许可证问题产生的原因有很多,包括但不限于:开发人员对开源许可证关注不够,对其了解不足[7];开发人员可能只管理直接依赖,而缺乏对传递依赖的足够控制[8]。

许可证类型繁多且申明内容多变,OSI 组织认证的开源许可证数量已达上百种[9]。这使得人工检测方法既费时又费力,利用程序进行许可证检测已成为必然需求。German 等[10]构建了 4 种信息库用于许可证检测,包括关键字、等价

到稿日期:2023-12-15　返修日期:2024-05-06
通信作者:朴勇(piaoy@dlut.edu.cn)

短语集合、句子标记表达式和许可证规则，通过将待分析数据替换为标准短语集的形式进行正则匹配来检测许可证的类型，从而创建了 Ninka 工具。惠普公司的 Fossology[11] 工具是通过模式匹配程序创建的，其使用一种 bSAM 符号对齐矩阵对每个文件进行分析。Pombredanne 等通过创建大量的规则文件与许可证文件的匹配结果给出许可证类型，实现了检测许可证工具 Scancode-Toolkit。一些开发人员在此基础上实现了基于需求的许可证检测与推荐功能[12-14]。

上述的许可证检测工具都需要构建复杂的匹配信息库，并使用传统匹配方式来检测许可证。这种方式在需要更新许可证信息或添加新许可证时，需要大量人力参与匹配库的维护[15]。而许可证申明具有规范性强、重复度高等特点，采用相似度检测不仅能节省构建和维护匹配规则的时间成本，还能在检测结果中体现出相似度。近年来，大语言模型的迅速发展，也使得通过对比可以表征语义的特征向量的相似度来进行许可证识别具有可行性。现有的关于兼容性以及冲突分析方法主要停留在人工分析的层面，需要有相关专业知识的人员对项目中全部的许可证类型进行分析。在此背景下，自动化推荐许可证需求应运而生。

综上所述，为了避免大量匹配所带来的高成本，本文主要使用了相似度比较作为检测许可证的方法，通过机器学习的方式来检测，而不用消耗大量人力去构造完整的匹配规则。

本文的主要贡献为：1）提出了一种基于集成学习的开源许可证的检测方法；2）归纳制作了开源许可证数据集来训练多个大语言模型，比较了其对于许可证文本的表征能力，提高了许可证的检测效率；3）基于许可证的兼容性关系，实现了依据需求的许可证推荐系统。

2 相关工作

本章主要介绍开源许可证、BERT 模型生成文本向量，以及其生成向量的各向异性问题与解决策略。

2.1 开源许可证

开源许可证是对代码的一种许可证授权，其以法律的形式对受版权法保护的开源软件的使用、复制、修改和分发等行为进行规范[16]。许可证识别和兼容性判断问题主要有 3 类方向。1）寻找许可证申明：许可证的申明包括项目许可证与源代码许可证。项目许可证的申明存在于项目根目录的“LICENSE”文件或“README”文件中。源代码许可证申明常在其头部的注释中。2）识别许可证类型：项目许可证声明文件通常是许可证文本的拷贝，可以使用 Jaccard 方法直接比较文本相似度，时间复杂度低，效果好。对于源代码头部注释的许可证申明，通过观察可以将其分为两类：按照许可证文本指示编写申明，以及简要地编写申明许可证。规范的许可证一般都带有许可证申明编写的范例，如图 1 所示。简要的许可证申明通常只会说明许可证类型，或是额外提供许可证的 URL 链接或项目中许可证的地址，例如“This program is under xxx License, see〈URL〉.”，从而根据申明的语义相似度来判断类型。3）许可证兼容性分析：对每个许可证的版权、专利许可、商标许可、分发限制条件、免责声明与责任限制等多个行为的声明进行兼容性分析，通常需要法律领域人士对许可证条款进行详细的分析解读。同时，许可证数量庞大且许多许可证的自定义与编写不规范，使得兼容性分析更加困难。本文参考 Kapitsaki[17] 以及 Wang 等[18] 所提出的兼容性模型，利用有向图算法实现了许可证兼容性的检测。

```
Copyright(C) 〈year〉〈name of author〉
…it under the terms of the 〈License type〉…
This program …〈License Rule〉…
…see 〈contact information(url/mail)〉 …
```

图 1 许可证申明范例

Fig. 1 Example of license declaration

2.2 BERT 模型与各向异性

BERT 模型在微调后具有强大的语义理解能力。BERT 模型的训练层由 12 层 TransformerEncoder 层拼接而成，每一层的输出为下一层的输入。每一层的输入和输出都是将数据处理成一个首部标识位加上数据中的分词再加上分隔位的标准形式，然后编码成固定长度 768 维的向量，并进行多次特征提取。最后输出的向量的相似度一定程度上代表了原句的语义相似度。

BERT 等语言模型在多数 NLP 任务中能取得优异的表现，但如果直接取 BERT 输出的句向量作表征，得到的效果甚至还不如 Glove 词向量。产生该现象的原因是 BERT 模型产生的句向量是各项异性的[19]，又称为表达退化问题，表现状态就是向量会呈现不均匀分布，低频词离原点远，分布稀疏，高频词向量离原点近，分布紧密，如图 2 所示。

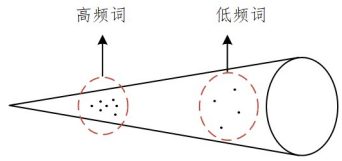

图 2 BERT 句向量各向异性

Fig. 2 Anisotropy in BERT sentence vectors

利用本文所提出的基于集成学习的方式来对许可证进行检测的核心在于文本语义的相似度是否与模型所生成的句向量相似度相匹配。但由于各向异性问题，BERT 模型训练得到的向量都挤在一起，彼此之间的余弦相似度都很高，不能反映许可证的相似性，因此要针对各向异性问题进行改进。BERTflow[19] 的思想就是将原来向量空间的分布校准为高斯分布，并将其映射为各向同性的；Bert-whitening[20] 是对句向量进行白化操作，即将 SVD 分解的结果旋转缩放后得到一个标准正态分布。还有模型通过对比学习方法，在保证向量空间高均匀性的情况下，也提高其一致性，例如 ConSERT[21]，SimCSE[22] 和 ESimCSE[23] 等。这些模型是集成学习方法的一部分。

3 开源许可证检测与兼容性分析方法

本节主要介绍如何完成开源许可证的识别，以及基于许可证兼容性的分析与推荐。所提系统的基本架构如图 3 所示。

系统主要分为文件输入层、文件扫描层、许可证类型检测层与许可证分析层 4 个模块，其中许可证类型检测层是核心。

项目许可证(PL)的声明是规范的“.license”文件，这是许可证的官方文本的拷贝。由于是对规范的文本文件进行相似度对比，因此该部分使用效率较高的 Jaccard 相似度方法进行度量检测。对代码许可证(CL)的检测就复杂得多，因为其许可证申明一般位于源代码头部的注释中，该部分并不是规范的许可证文本，存在简易声明、自定义声明等，甚至有些使用 URL 链接外部声明。本文的重点便是对代码文件的许可证进行检测。

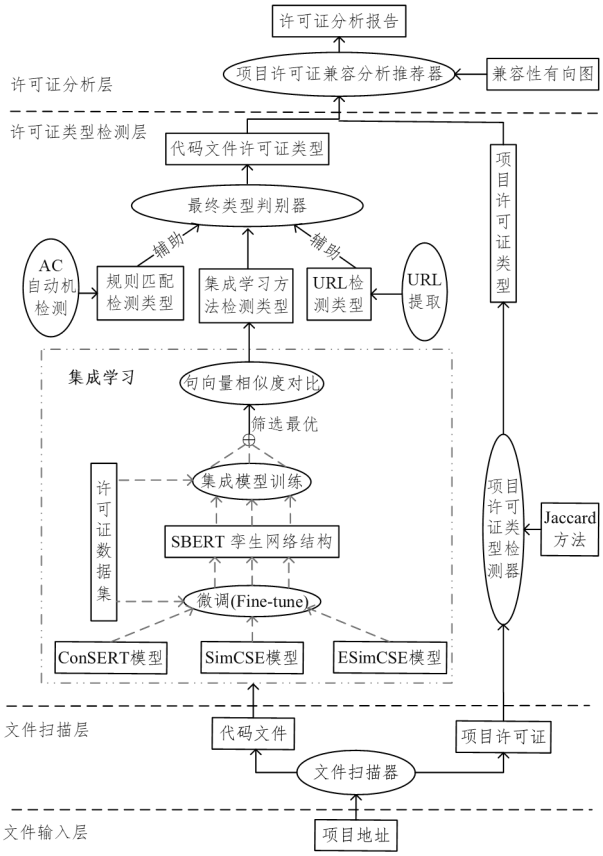

图 3 系统架构

Fig. 3 System architecture

我们在许可证识别中主要提出了基于大语言模型的集成学习方式来进行许可证检测。同时，为了提高许可证识别的准确性，本文还实现了基于关键词与 URL 的多模式匹配方法来辅助进行许可证的检测。在基于许可证兼容性的分析推荐中，利用有向图的广度优先遍历算法来实现。

3.1 基于集成学习的许可证的识别

为了能高效地进行许可证的检测识别，我们详细比较了各个大语言模型对许可证文本语义的表征能力，并观察了数据集的训练对模型的表征能力的提升效果。使用自制的许可证数据集对原始 BERT，SBERT[24]，ConSERT，SimCSE 以及 ESimCSE 模型进行训练与验证，提高了模型关于许可证文本语义的表征能力。随后，创新性地提出了一种基于大语言模型的集成学习训练方法，进一步提升了模型表征能力，并使用该方法进行基于相似度对比的许可证识别。本部分将主要介绍以基于大语言模型的集成学习方式为主，以基于许可证关键词的多模式匹配为辅的方式进行许可证检测。

集成学习方法的核心思想是，通过将多个基础模型的预测结果进行组合，产生一个强大的集成模型，其能够在各种情况下表现得更好。利用这种思路，将上述的大语言模型按特定方式与结构集成在一起，通过自制的许可证相关数据集进行集成学习，从而大幅提高模型对许可证的语义表征能力，并据此来完成许可证的识别。下面将详细介绍集成学习的训练方法。

3.1.1 集成学习方法

预训练语言模型在进行文本相似度任务时使用的传统策略是进行二分类任务，常见的是在模型的顶层添加一个 sigmoid 激活函数输出概率。结合常见预训练模型与 SBERT 的双塔孪生网络结构的思想，提出了一种集成学习方法来完成文本相似度与许可证检测任务。

首先，实现上述模型的模型结构，在预训练好的模型参数上使用许可证相关数据集进行模型微调，使得模型在特定的许可证数据中的表征能力大幅提高。随后，利用双塔孪生网络结构，针对不同的预训练模型构建新的模型结构。这个模型是孪生结构，每个孪生网络均去除了原模型的分类层之后的结构，权重共享，即每个孪生网络的输出为长度相同的表征向量。接着，根据数据集的类型，使用不同的损失函数来进行训练。具体的训练过程如图 4 所示。

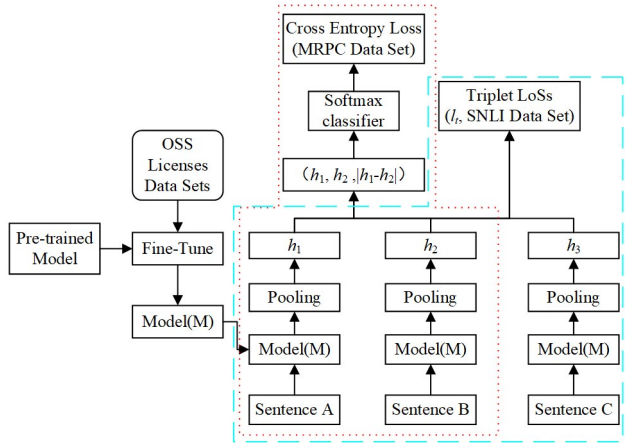

图 4 集成学习的训练过程

Fig. 4 Training process of ensemble learning

为了能显著提高语言模型对许可证文本语义的表征能力，设计制作了两种数据集来进行模型训练与预测，结果也表明这种方式起到很大效果。制作的数据集有两种格式，分别为 MRPC 数据集格式与 SNLI 数据集格式。

1)第一类数据集格式为〈sentence1，sentence2，label〉，每个样本由一对句子和一个二元标签组成。标签指示两句话是否是语义等价的复述，即若一个样本两句话分别表示的许可证条款语义相同，则标签为 1，反之标签为 0。在使用这类数据集进行训练时，模型经过编码、分类层后，输出‘语义相似’这一结果的概率。损失函数为二分类交叉熵损失，如式(1)所示：

$$L(label,p)=-(label * \log(p)+(1-label) * \log(1-p)) \tag{1}$$

2)第二类数据集格式为＜anchor，positive，negative＞，每个样本由 3 个句子组成。其中 positive 是 anchor 的合理推论或蕴含关系，negative 与 anchor 是矛盾关系，即 positive 句子

代表的许可证条款可以被 anchor 所推论出或者语义相似，而 negative 句子代表的许可证条款与 anchor 句子代表的条款相矛盾。具体地，假设 h_i 表示 anchor，h_i+ 和 h_i- 分别是推论样本和对立样本，则训练损失函数为：

$$\ell_i=-\log\frac{e^{sim(h_i,h_i^+)/\tau}}{\sum_{j=1}^{N}(e^{sim(h_i,h_j^+)/\tau}+e^{sim(h_i,h_j^-)/\tau})} \tag{2}$$

使用集成学习方法进行训练并将训练后的模型运用于项目中的流程如算法 1 所示。

算法 1 集成学习算法

输入：预训练模型(pretrain_model)，扫描到的许可证文本(Data)，MRPC 格式许可证训练集(license_dataset_MRPC)，SNLI 格式许可证训练集(license_dataset_SNLI)

输出：许可证文本经过集成学习方法训练后的模型所编译的句向量(h)

Step1：对预训练模型进行微调

1. Fine_Tune(pretrain_model, license_dataset) / * 使用许可证数据集对预训练模型进行微调
2. M=model.get_parameters() / * 记录微调后的模型参数为 M

Step2：使用孪生结构对模型进行训练

3. trained_model=load_parameters(M) / * 加载微调后的模型参数
4. for sample in license_dataset_MRPC:
5. sentence_A, sentence_B=sample["Sentence A""Sentence B"] / * 提取样本
6. $\mathbf{h}_1,\mathbf{h}_2$ = Encode(trained_model, sentence_A, sentence_B, pooling) / * 编码
7. h_diff=$|\mathbf{h}_1-\mathbf{h}_2|$ / * 取两个句向量的绝对值
8. concatenated_vector=Concatenate($\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}$_diff) / * 将这 3 个向量在−1 维度进行拼接
9. probabilities = Softmax(concatenated_vector) / * 将向量进行归一化处理
10. loss=BinaryCrossEntropy(probabilities, sample["label"]) / * 优化目标为二分类交叉熵函数
11. UpdateParameters(trained_model, loss) / * 训练模型，更新参数
12. end for
13. for sample in license_dataset_SNLI:
14. sentence_A, sentence_B, sentence_B=sample["Sentence A", "Sentence B", "Sentence C"]
15. h_1,h_2,h_3 = Encode(trained_model, sentence_A, sentence_B, sentence_C, pooling) / * 编码
16. loss=func(h_1,h_2,h_3) / * 优化目标为三元组损失函数
17. UpdateParameters(trained_model, loss) / * 训练模型，更新参数
18. end for
19. EL=trained_model.get_parameters() / * 记录通过集成学习方法训练后的模型参数为 EL

在进行集成学习训练时，首先使用许可证数据集对预训练模型进行微调，记录模型微调后的参数。随后将数据集放入孪生结构模型中进行训练，若使用的是 MRPC 数据集格式的许可证数据，则 Sentence A 与 Sentence B 表示一个样本中的两个句子，经过已经微调过的模型与池化层后得到句向量 $\boldsymbol{h}_1$ 与 $\boldsymbol{h}_2$，$|\boldsymbol{h}_1-\boldsymbol{h}_2|$ 表示两个向量的绝对值。根据 SBERT 模型提供的思路，将这 3 个向量在−1 维度进行拼接，再使用 Softmax 函数归一化为概率分布，使用二分类交叉熵函数(见式(1))进行优化训练。若使用的是 SNLI 数据集格式的，则 Sentence A，Sentence B 与 Sentence C 表示一个样本中的 3 个句子，且句子 B 与 A 是蕴含关系或语义相近，句子 C 与 A 是矛盾关系。将数据输入已微调过的模型进行编码并进行池化得到句向量 $\boldsymbol{h}_1$，$\boldsymbol{h}_2$ 与 $\boldsymbol{h}_3$，随后使用上述有监督学习中介绍的式(2)来进行优化训练。通过实验发现，训练过程中池化层使用平均池化效果较好。

通过这种集成学习的训练方法，模型对许可证条款语义的表征能力有了大幅提升。之后的实验也表明这种方法可以高效地对文本相似度进行对比，从而完成许可证的识别。

使用集成学习方法训练好模型后，将从待测项目中扫描到的代码许可证申明输入到模型，而后进行编码，生成对应句向量，最后对句向量进行相似度比较，从而确定出待测的许可证类型，完成许可证识别任务。

对于同一类型的不同许可证，例如 BSD-2-Clause 和 BSD-3-Clause 申明只差一句话，其他部分完全相同，这导致相似度判定无法很好地区分两者。针对这种情况，将包含多版本的许可证类型合并为一个类型集，相似度检测结果归属这个集合而不是某一具体许可证。例如，常见的 GPL，BSD 和 MIT 等都被作为集合处理。作为补充，我们实现了 AC 自动机关键字提取技术来对这种情况进行辅助类型判断。

3.1.2 AC 自动机关键字提取

在扫描项目中的许可证申明时，用模式集合 $P\{p_1,p_2,\cdots,p_m\}$ 对一段文本运用 AC 自动机进行匹配检测。假设文本长度为 n，可以在 $O(n)$ 时间复杂度内找到文本中的所有匹配模式。

对于模式匹配 $P\{$'*he*','*his*','*her*','*she*'$\}$，可以创建树形结构，如图 5 所示。树中虚线表示下一次匹配中如果没有子节点能匹配上时，当前节点应该移动的位置。带有下划线的节点存储当前成功匹配的模式信息。假如待匹配文本为'sher'，在匹配到'she'后，当前节点的位置在右下角的'e'节点，而'r'并没有出现在当前节点的子节点中，因此按虚线移动当前节点到左边的'e'节点，左边的'e'节点存储了'he'的模式信息，所以'he'也匹配上了，同时当前节点的子节点有'r'节点并移动，'her'也匹配文本。仅遍历了一次待匹配文本，就能将所出现的模式全部匹配上。本文将节点中的字符替换为字符串，创建了类似于{"Lesser General Public License v2.1","Affero General Public License version 1.0","Apache License Version 2.0",…}的关键字匹配模式，通过匹配申明中的关键字，可以进一步确定申明的类型。

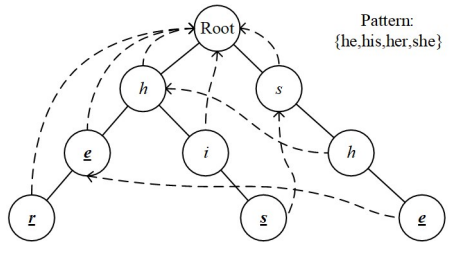

图 5 AC 自动机样例

Fig. 5 Example of AC automaton

3.2 基于兼容性的分析与推荐

本文利用兼容性先验知识构建许可证兼容性有向图来对项目中许可证的兼容性进行分析，并据此进行推荐。Kapitsaki 等分析常用的许可证类型之间的兼容性，并设计了较为全面的许可证兼容性有向图；根据先验知识重新构建了兼容图，并进行了部分优化。图 6 为改进的部分许可证兼容性有向图。

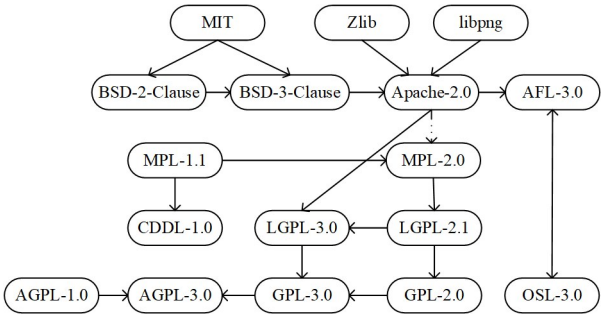

图 6 许可证兼容有向图

Fig. 6 Compatibility directed graph for common licenses

在有向图中，箭头指向表示许可证单向兼容，如“AFL-3.0”到“OSL-3.0”有单向箭头，其表示若项目中出现了这两种许可证，可以按照箭头指向的“OSL-3.0”许可证进行分发；虚线则表示两端许可证的兼容性不能继续向下传递，即“Apache-2.0”与“MPL-2.0”兼容，但与“LGPL-2.1”不兼容。最终可推荐的许可证不必限制于原有许可证类型中，但需要寻找到最优推荐。例如项目中所含许可证有[“Apache-2.0”，“MPL-1.1”]，经过推荐算法得出项目最终可以按“MPL-2.0”进行分发。算法 2 描述了分析项目许可证组兼容性的过程，利用 res_list(比较集合)存储已访问过的许可证的推荐结果，利用 lic_temp(中间集合)存储单个许可证与已访问过的许可证的推荐结果。

算法 2 许可证组兼容性分析算法

输入：license_list{L_1，L_2，…，L_n}/*项目许可证类型集合*/

输出：res_list{L_1^r，L_2^r，…，L_n^r}/*许可证集合推荐许可证，若不兼容则为空*/

```
Begin/*对类型集合进行预处理，初始化中间集合*/
1. for lic in license_list:
2.    if len(res_list)==0:
3.       res_list.append(lic)
4.    else:
5.       for lic_res in res_list:
6.          if lic_res==lic:
7.             lic_temp.append(lic)
8.             continue
9.          else:
10. lic_temp=list_add(lic_temp,compatibility(lic,lic_res))
11.         if len(lic_temp)==0:
12.            return None
13.         else:
14.            res_list=lic_temp.copy()
15.            lic_temp.clear()
End/*最终的比较集合则为推荐结果*/
```

其中，compatibility 函数为两个许可证求推荐许可证，使用有向图的广度优先遍历算法实现了该函数的功能，所推荐的许可证为同一深度的许可证，即宽松程度相当的许可证，从而保证寻找到的许可证是最优推荐。在有向图中虚线表示不可扩展兼容性，我们使用虚拟复制节点方法来进行实现，复制节点只存储虚线关系，原节点存储实线关系。上述方法能准确地检测出许可证兼容关系，并给出兼容的推荐许可证。

系统中每个许可证信息并没有关联性，它们是一个个不相关的独立个体，因此对其进行标注和兼容性判断至关重要。在实际的许可证检测与兼容性分析过程中，分别对代码文件与项目许可证声明文本进行扫描，并依据集成算法来识别许可证，得到一个项目许可证 PL，以及若干个代码许可证 $CLs=\{CL_1,CL_2,\cdots,CL_n\}$。随后，利用算法 2 对所有的 CL 进行兼容性分析后，得到一个兼容于 CLs 的开源许可证 CL_r。最后对 CL_r 和 PL 进行兼容性分析。若 PL 兼容于 CL_r，则该系统许可证无兼容性隐患；若上述兼容性分析过程中存在冲突，则记录冲突，反馈结果。

4 实验

本章展示了本文所提出的基于集成学习的开源许可证检测方法的评估结果，证明了该方法能够有效地检测出项目中包含的开源软件许可证，同时也展示了通过有向图推荐算法来判定许可证兼容性的有效性。

4.2 数据来源

本文的一部分数据集采用的是 Scancode 中的部分许可证模式规则总结、源代码头部注释文本等，另一部分是实验组将常见许可证根据其条款划分并进行细粒度数据建模得到的结构化数据。通过使用自然语言处理技术对上述数据进行处理，得到许可证数据集。

4.3 实验结果

首先研究了 BERT 模型产生句向量的各向异性问题，将带有“LGPL-2.1”的句子与带有其他常见开源许可证名称的句子分别输入 BERT 模型并比较其输出的句向量之间的相似度，通过它们之间的相似度来反映 BERT 模型对于许可证文本的语义表征效果。实验结果如表 1 所列。

表 1 BERT 模型句向量具有各向异性

Table 1 BERT model sentence vectors are anisotropic

License Embedding	Similarity	License Embedding	Similarity
GPL-2.0	0.9871	GPL-3.0	0.9535
AGPL-1.0	0.9693	CDDL-1.0	0.9497
libpng	0.9616	BSD-2-Clause	0.9446
AGPL-3.0	0.9578	MPL-1.1	0.9373
Zlib	0.9561	MIT	0.9353
LGPL-2.1	0.9554	Apache-2.0	0.9203
BSD-3-Clause	0.9544	OSL-3.0	0.9165
LGPL-3.0	0.9543	AFL-3.0	0.9106

从表 1 中可以看到，经过 BERT 模型编码后，“LGPL-2.1”的句向量与其他常见开源许可证的句向量相似度都在 90% 以上，大多数相似度超过了 95%。因此，不能直接将其用于相似度对比的任务中。

4.2.1 消融实验

为了验证集成学习训练方法的有效性，我们进行了消融实验。首先，从 Huggingface 上下载了多个预训练模型参数，并在不进行微调的情况下直接使用许可证文本数据集的测试集进行测试，以判定未微调的模型对许可证的语义表征能力。从表 2 可以看出，BERT 模型的准确率远低于其他预训练模型。

表 2 微调前预训练模型的准确度

Table 2 Accuracy of pre-trained models before fine-tuning

Pre-trained Model	*Accuracy*
bert-base-uncased	0.275
sup-simcse-bert-base-uncased	0.522
unsup-simcse-bert-base-uncased	0.558
esimcse-bert-base-uncased	0.561

为了证明制作的数据集对于提高模型语义表征能力的有效性以及本文所提出的集成学习方法的有效性，我们使用开源许可证数据集对不同的模型结构进行训练与测试。训练过程中，为了满足消融实验条件并突出不同模型结构对结果的影响，将使用相同的预训练模型参数来进行实验。本轮实验所使用的是从 huggingface 中下载的官方提供的“bert-base-uncased”模型参数。在测试过程中我们所使用的是“sentence-transformer”包中的 BinaryClassificationEvaluator 类，该类的主要功能是通过比较句子嵌入的相似度来进行二分类任务的评估。该类提供了多种评估指标来量化模型表现，包括准确度、F1 分数、精确度、召回率以及平均精确度。消融实验结果如表 3 所列，其中 MRPC 与 SNLI 分别表示使用 MRPC 与 SNLI 数据集格式的开源许可证数据集进行的训练。

表 3 消融实验结果

Table 3 Results of ablation study

(%)

	Accuracy	*F1*	*Precision*	*Recall*	*Average Precision*
BERT	69.75	51.53	39.12	75.81	48.66
Unsup_SimCSE	75.08	61.01	48.62	81.86	65.84
Unsup_ConSERT	76.49	62.96	62.67	63.26	68.98
Unsup_ESimCSE	76.98	62.25	51.52	78.6	69.58
Sup_ConSERT (MRPC)	83.48	75.81	76.15	83.29	80.11
Sup_ConSERT (SNLI)	85.39	79.53	77.21	85.57	84.79
Sup_SimCSE (MRPC)	83.77	78.52	75.09	82.14	80.92
Sup_SimCSE(SNLI)	85.85	80.74	76.27	85.35	85.03
Ensemble Learning (MRPC)	91.07	86.90	85.91	87.83	91.69
Ensemble Learning (SNLI)	93.10	90.39	85.19	93.28	94.38
Ensemble Learning	96.71	95.22	93.31	97.21	99.33

首先将表 3 中的准确度与表 2 中的准确度作对比，各模型的准确度较微调之前的预训练模型的准确度有了极大幅度的提高，证明了我们制作的数据集对各个模型语义表征能力的提高是有效的。从不同模型的结果来看，我们提出的集成学习的训练方法获得了最好的成绩，表明该方法最能够反映出开源许可证条款之间的相似度与其输出句向量之间的相似度的关联性，证明了我们所提出的集成学习方法对于识别许可证任务的有效性。

结合表 2 的数据，为了获得最优异的语义表征模型，使用表现较好的预训练模型参数来进行集成学习训练，最终的准确率超过了 95%。因此，将使用该方法训练的模型所输出的句向量来完成许可证的检测识别任务。

在消融实验中，我们发现训练集文本中，语义混乱的 URL 链接对模型的训练结果具有较大影响。表 4 对通过数据预处理消除 URL 等一些无关语义的字符前后的实验结果进行了对比。

表 4 URL 链接等无关字符对训练的影响

Table 4 Impact of URL links on training

Category	*ACC*	*AUC*
Common data sets	0.8982	0.8985
Remove URL-linked data sets	0.9656	0.9643

4.2.2 许可证识别任务与兼容性测试结果

下面是对项目本身的效果进行测试。从 Debian5.0.2 中随机挑选了 250 个应用程序，并从每个应用程序中随机抽取一个代码文件，以构建一个数据集。将自行开发的程序(MySystem)的检测结果与手工标注结果进行了对比验证，将对比结果分为 3 类：C 类表示许可证匹配且类型版本一致，或都未检测出许可证；I 类指系统识别至少有一个不正确或与手工标注不一致；U 类表示该文件的手工识别结果为有许可证，系统识别时能够识别到许可证但无法检测出类型。表 5 与图 7 展示了各项工具的实验结果。

表 5 各项工具的分类结果

Table 5 Classification Results of each tool

	C	I	U
Ninka	200	7	43
Fossology	137	112	1
ohcount	83	167	0
OSLC	57	193	0
MySystem	232	18	0

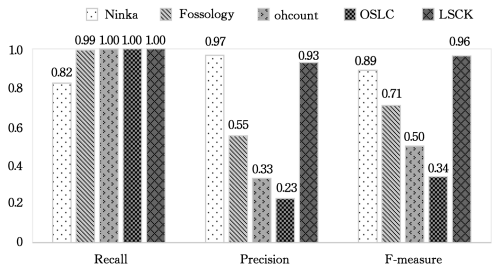

图 7 各类工具实验结果指标对比

Fig. 7 Results of various tool experiments

Ohcount，OSLC 以及我们的系统都无法识别未知许可证，因此召回率达到了 100%。在准确度上，我们的系统略低于 Ninka 但远高于其他软件，这可能是由于我们的系统受限于许可证库，尚未扩展到某些小众许可证类别，在这些许可证识别上与成熟的 Ninka 工具略有差距。我们的系统在 F-measure 指标上领先于其他的系统，这得益于我们的系统中基于集成学习的相似度检测和关键字检测的多层检测机制。同时，我们的系统在数据扩展和维护方面也更加便捷，这意味着系统有巨大的提升空间。综合实验结果，在目前主流的

研究工具中我们的系统优势巨大。

在许可证推荐模块中，我们在实际项目中检测到了多种许可证的组合使用形式，系统推荐的许可证结果与手工分析结果一致。同时，在对工具进行测试的过程中，我们选取了一些开源项目利用工具进行分析。表 6 与表 7 分别为工具分析了在 GitHub 上的 Scancode 与 Jieba 两个开源项目中一些文件的分析结果。

表 6 Scancode 开源项目实际分析

Table 6 Analysis results of Scancode

Relative paths	Tool analysis results
.\cli.py	MIT
.\interrupt.py	Apache-2.0
.\lookfile.py	Apache-2.0
.\outdated.py	BSD-3-Clause
.\pool.py	MIT
.\LICENSE	Apache-2.0

表 7 Jieba 开源项目实际分析

Table 7 Analysis results results of Jieba

Relative paths	Tool analysis results	Manual analysis results
.\creator.py	Apache-2.0	Apache-2.0
.\nets.py	Apache-2.0	Apache-2.0
.\predict.py	Apache-2.0	Apache-2.0
.\small.py	Apache-2.0	Apache-2.0
.\utils.py	Apache-2.0	Apache-2.0
.\LICENSE	MIT	MIT

表 6 展示了对一个项目系统中许可证的标注结果。该项目的文本许可证有“MIT”“BSD-3-Clause”以及“Apache-2.0”，兼容于所有文本许可证的是“Apache-2.0”，同时项目许可证也为“Apache-2.0”。因此，通过对项目中文本许可证与项目许可证的识别标注以及兼容性分析，得出该项目中的许可证是兼容的。

从表 7 中可以看出，我们的系统对检测与手动标注的结果是一致的。需要注意的是，项目中的某些开源组件使用的是“Apache-2.0”许可证，但该项目却按照“MIT”许可证进行发布。我们的系统推荐的项目许可证为“Apache-2.0”。这是因为“MIT”许可证比“Apache-2.0”许可证更加宽松，也就是说当项目中同时有“MIT”与“Apache-2.0”许可证的开源组件时，组合后形成的项目按照“Apache-2.0”进行发布才是合规的。项目所使用的发布许可证必须是项目中所有开源组件携带许可证所向下兼容的。在我们进行工具测试的过程中发现了很多这类许可证不兼容的问题，可见开源许可证兼容性所带来的法律隐患是巨大的，我们也已通知部分项目作者关注相关问题。

结束语 本文的主要贡献是提出了一种基于集成学习的开源许可证的检测方法，该方法的主要思想是通过向量相似性对比来确定许可证。设计的基于大语言模型的集成学习的训练方式有效地缓解了 BERT 模型句向量各向异性的问题，大幅提高了模型对许可证文本的语义表征能力和许可证识别的准确率。通过对大语言模型详细的研究与训练，我们的基于集成学习的检测方法在具体项目实验中的效果超过了绝大多数传统的基于规则匹配的检测方式，也克服了其需要耗费大量成本编写匹配规则的缺点，更加便于维护与更新。

目前各个开源社区都发展迅猛，但开源项目中许可证的不兼容情况却非常普遍，法律隐患也十分巨大。开源许可证正是为了保护自身的合法权益而设立的，是互联网开源思想盛行的基础保障。因此，建议大家在使用开源代码或是创造开源项目时认真研读许可证条款。只有更多的人关注开源许可证，并让这一部分的知识得到更广泛的普及，才能让开源发展更加规范。

下一步在许可证兼容性分析阶段，我们将尝试使用知识图谱的补全来在先验的兼容性关系中寻找潜在的许可证兼容性关系，从而实现自动化的许可证兼容性判断与推荐系统。

参 考 文 献

[1] XU S，GAO Y，FAN L，et al. Lidetector：License incompatibility detection for open source software[J]. ACM Transactions on Software Engineering and Methodology，2023，32(1)：1-28.

[2] ZHAO L. An Analysis of Open Source Components in Mixed Source Software Projects [J]. Computer Science，2020，47(S2)：541-543，583.

[3] TU L Y. An Analysis of Legal Attribute of Open Source Software License [J]. Legal System and Society，2021(17)：189-190.

[4] LIU B B. Research on the Legal Issues of Open Source License Agreement [D]. Lanzhou：Lanzhou University，2020：1-45.

[5] KAPITSAKI G M，CHARALAMBOUS G. Find your Open Source License Now! [C]// Asia-Pacific Software Engineering Conference(APSEC). IEEE，2016：1-8.

[6] WOLTER T，BARCOMB A，RIEHLE D，et al. Open source license inconsistencies on github[J]. ACM Transactions on Software Engineering and Methodology，2023，32(5)：1-23.

[7] ALMEIDA D A，MURPHY G C，WILSON G，et al. Investigating whether and how software developers understand open source software licensing[J]. Empirical Software Engineering，2019，24：211-239.

[8] PASHCHENKO I，VU D L，MASSACCI F. A qualitative study of dependency management and its security implications[C]// Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020：1513-1531.

[9] OSI approved licenses [OL]. https://opensource.org/licenses/

[10] GERMAN D M，MANABE Y，INOUE K. A sentence-matching method for automatic license identification of source code files [C]// Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering. 2010：437-446.

[11] JAEGER M C，FENDT O，GOBEILLE R，et al. The FOSSology Project：10 Years Of License Scanning [J]. International Free and Open Source Software Law Review，2018，9(1)：9-18.

[12] KAPITSAKI G M，TSELIKAS N D，FOUKARAKIS I E. An insight into license tools for open source software systems [M]. Elsevier Science Inc.，2015：72-87.

[13] HARUTYUNYAN N，BAUER A，RIEHLE D. Industry requirements for FLOSS governance tools to facilitate the use of open source software in commercial products [J]. The Journal of

Systems and Software, 2019, 158(Dec.): 110390. 1-110390. 1-12.
[14] KAPITSAKI G M, CHARALAMBOUS G. Modeling and recommending open source licenses with findOSSLicense[J]. IEEE Transactions on Software Engineering, 2019, 47(5): 919-935.
[15] LIU X, HUANG L G, GE J, et al. Predicting licenses for changed source code[C]// 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2019: 686-697.
[16] BALLHAUSEN M. Free and open source software licenses explained [J]. Computer, 2019, 52(6): 82-86.
[17] KAPITSAKI G, KRAMER F, TSELIKAS N D. Automating the license compatibility process in open source software with SPDX [J]. Journal of Systems & Software, 2016, 131(Sep.): 386-401.
[18] WANG Z Q, WU S, XIAO G Q, et al. How to Properly Choose Open Source Software Licenses for Open Source Software [J]. Journal of Software, 2021, 32(5): 1227-1229.
[19] LI B, ZHOU H, HE J, et al. On the sentence embeddings from pre-trained language models[J]. arXiv: 2011. 05864, 2020.
[20] SU J, CAO J, LIU W, et al. Whitening sentence representations for better semantics and faster retrieval[J]. arXiv: 2103. 15316, 2021.
[21] YAN Y, LI R, WANG S, et al. Consert: A contrastive framework for self-supervised sentence representation transfer[J]. arXiv: 2105. 11741, 2021.
[22] GAO T, YAO X, CHEN D. Simcse: Simple contrastive learning of sentence embeddings[J]. arXiv: 2104. 08821, 2021.
[23] WU X, GAO C, ZANG L, et al. Esimcse: Enhanced sample building method for contrastive learning of unsupervised sentence embedding[J]. arXiv: 2109. 04380, 2021.
[24] REIMERS N, GUREVYCH I. Sentence-bert: Sentence embeddings using siamese bert-networks [J]. arXiv: 1908. 10084, 2019.

BAI Jianghao, born in 1999, postgraduate. His main research interests include natural language processing and so on.

PIAO Yong, born in 1975, Ph. D, associate professor, is a member of CCF (No. E3677M). His main research interests include data mining and intelligent computing.

(责任编辑:柯颖)