

分 类 号: TP311
研究生学号: 2019532054

单位代码: 10183
密 级: 公 开



吉 林 大 学

硕士学位论文

(学术学位)

基于文本分析的开源 npm 包分类研究

Research on Open-Source npm Packages Classification

Based on Text Analysis

作者姓名: 王禹

专 业: 计算机软件与理论

研究方向: 数据驱动的智能化软件工程

指导教师: 刘华虢 副教授

培养单位: 计算机科学与技术学院

2022 年 5 月

基于文本分析的开源 npm 包分类研究

Research on Open-Source npm Packages Classification
Based on Text Analysis

作者姓名：王禹

专业名称：计算机软件与理论

指导教师：刘华琥 副教授

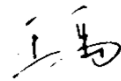
学位类别：工学硕士

答辩日期：2022 年 5 月 15 日

吉林大学硕士学位论文原创性声明

本人郑重声明：所呈交学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：



日期： 2022 年 5 月 23 日

关于学位论文使用授权的声明

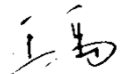
本人完全了解吉林大学有关保留、使用学位论文的规定，同意吉林大学保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权吉林大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。


（保密论文在解密后应遵守此规定）

论文级别： ☒ 硕士 ☐ 博士

学科专业： 计算机软件与理论

论文题目： 基于文本分析的开源 npm 包分类研究

作者签名： 

指导教师签名： 

2022 年 5 月 23 日

摘要

Npm (Node Package Manager)作为 JavaScript 语言的软件包管理器,管理着超过 250 万的开源第三方库——npm 包。由于缺少合适的分类方式,海量的软件资源面临着管理与检索的难题。在开发者分享和交流 npm 包的开源社区中,自定义的标签在描述包功能的同时也起到了一定程度的分类作用,然而当前的标签机制存在着内容混杂、同义词表述形式不同等问题,无法满足管理与检索的需求。不仅如此,npm 社区中超过四成的包缺少标签,且庞大的数量使得无法以人工方式实现标签补全。为此,本文围绕 npm 社区类别标签构建和自动化的 npm 包多标签分类方法这两个问题展开研究。

(1)提出了一种基于标签关联关系构建类别标签的方法,来为 npm 包面向功能构建分类类别。该方法首先利用关联关系挖掘算法为 npm 社区中的标签生成标签关联关系图。其次通过社区检测算法将标签基于关联关系聚类,形成多个表示独立功能的标签关联社区。最后,通过人工方式筛选、合并标签关联社区,并根据标签在社区中的影响力设计了类别标签识别机制。本文对 npm 社区中被依赖次数最多的 8000 个包应用此方法,得到了具有代表性的 35 个类别标签。

(2)提出了一种面向 Readme 文档的多标签文本分类方法,实现 npm 包的自动化分类。该方法首先根据自述文档 Readme 的内容结构,制定了针对于提取功能描述信息的内容切分方案。其次采用带有权重的关键词集去捕捉 Readme 文档中的分类信息与类别标签之间的语义关联,使得方法在分类准确性上优于传统的多标签文本分类方法。方法中先基于有监督的主题模型 L-LDA 训练时生成的主题词分布来构建类别标签关键词集。再利用词移距离算法计算待分类包的 Readme 文档与各类别标签关键词集的语义相似度。最后根据相似度的排序结果赋予待分类包类别标签。

经过实验验证,本文所提出的面向 Readme 文档的多标签文本分类方法可以有效地为 npm 包从功能上进行分类。与作为基线的多标签分类方法相比,本文方法在 Macro-F1, Hamming Loss 和 LRAP 三个多标签分类评价指标上均有大幅提升,验证了方法分类的准确性。同时,该方法对实际的无标签包分类中也有较好表现,从而验证了方法的有效性。此外,本文为开源 npm 包分类研究建立

了具有代表性的数据集。

关键词：

npm；开源软件分类；多标签分类；文本分析

Abstract

As the package manager for the JavaScript language, npm (Node Package Manager) manages more than 2.5 million open-source third-party libraries—npm packages. Because there are no suitable classification methods for npm packages, the massive software resources are faced with the difficult problem of management and retrieval. In the open-source community where developers share and communicate npm packages with each other, custom tags play a certain role in classification while describing the functions. However, the current tagging mechanism has some problems such as mixed content and different expressions of synonyms, so that it cannot meet the needs of management and retrieval. Not only that, but more than 40% of packages in the npm community lack tags, and the large number makes it impossible to manually complete tags. For the above reason, this paper focuses on the two problems of category tag construction in the npm community and automated npm packages multi-label classification methods.

(1) We propose a method for building category tags based on the associations between tags, which aims to build function-oriented taxonomy categories for npm packages. Firstly, the method uses the association rule mining algorithm to generate a tag association graph for tags in the npm community. Secondly, the tags are clustered based on the association relationship through the community detection algorithm, and several tag communities representing independent functions are formed. Finally, the tag community is filtered and merged manually, and we design a category tag recognition mechanism according to the influence of the tags in the community. This paper applies this method to the 8000 most depended upon packages in the npm community, and obtains 35 representative category tags.

(2) We propose a multi-label text classification method for Readme documents to automate the classification of npm packages. This method firstly formulates a content segmentation scheme for extracting functional description information according to the content structure of Readme file. Secondly, a weighted keyword set is used to capture the semantic association between the classification information and category tags in the

Readme document, which makes the method perform better than the traditional multi-label text classification method in classification accuracy. The method first constructs a keyword set for each category tag based on the topic-word distribution generated during the training of the supervised topic model L-LDA. Then, we use Word Mover's Distance algorithm to calculate the semantic similarity between the Readme document of the package to be classified and the keyword set of each category tag. Finally, the method assigns the category tag to the package according to the sorting result of similarity.

After experimental verification, the multi-label text classification method for Readme document proposed in this paper can effectively classify npm packages in terms of the functionality. Against the multi-label classification methods as the baseline, the method in this paper has greatly improved the three multi-label classification evaluation metrics of Macro-F1, Hamming Loss and LRAP, which confirms the classification accuracy of the method. At the same time, the method also has good performance in the actual untagged packages classification, which confirms the effectiveness of the method. Furthermore, this paper builds a representative dataset for open-source npm package classification research.

Keywords:

npm; open-source software classification; multi-label classification; text analysis

目 录

第 1 章 绪论.....	1
1.1 研究背景与意义	1
1.2 研究现状	2
1.2.1 npm 包相关研究	2
1.2.2 开源软件分类	3
1.3 本文主要工作	4
1.4 论文结构安排	5
第 2 章 相关理论技术	6
2.1 主题模型	6
2.1.1 主题模型 LDA	6
2.1.2 有监督主题模型 L-LDA	7
2.2 文本相似度	9
2.2.1 词嵌入	9
2.2.2 WMD 词移距离算法	10
2.3 多标签文本分类	11
第 3 章 类别标签构建方法	13
3.1 标签关联社区	13
3.1.1 标签关联关系挖掘	13
3.1.2 标签社区检测	15
3.2 类别标签识别机制	16
3.3 本章小结	18

第 4 章 面向 Readme 文档的多标签文本分类	19
4.1 预处理.....	19
4.1.1 Readme 内容切分	19
4.1.2 文本处理	22
4.2 基于关键词的多标签分类方法	23
4.2.1 类别标签关键词生成	24
4.2.2 Readme 权重词集	25
4.2.3 基于词移距离计算相似度	26
4.3 本章小结	28
第 5 章 实验及结果分析	29
5.1 实验数据及评价指标	29
5.2 参数设定（RQ1）	30
5.3 方法有效性评估（RQ2）	31
5.4 类别标签间差异（RQ3）	33
5.5 本章小结	35
第 6 章 总结与展望	36
6.1 工作总结	36
6.2 未来展望	37
参考文献.....	38
作者简介及在学期间取得的学术成果	41
致谢.....	42

第 1 章 绪论

1.1 研究背景与意义

近年来,JavaScript 已经成为应用于 web 浏览器和服务器的最流行的编程语言之一^[1]。Stack Overflow 2021 年的开发者调查中显示,JavaScript 连续九年成为最常用的编程语言。Npm (Node Package Manager) 作为 JavaScript 的包管理器方便了开发者在软件开发中依赖和重用 JavaScript 代码,这些由 npm 管理的软件包被称为 npm 包。Npm 包的开源属性促进了开发者们的交流与共享,数量日渐庞大的 npm 包和积极贡献的开发者形成了一个成熟的开源社区生态,即 npm 社区。现如今,npm 社区已经有超过 250 万个开源软件(npm 包),并且还在持续增加。与此同时,越来越多的软件系统依赖于 npm 包所提供的多样且高效的功能。

Npm 包开源的特性,庞大的数量和丰富的功能使得 npm 社区蓬勃发展的同时,却也带来一些难题。一方面,组织和管理海量的软件本身就是比较具有挑战性的问题;另一方面,除关键词检索外,npm 社区并没有根据功能需求去定位和检索相关 npm 包的额外途径,而关键词检索的方式既无法控制输入词的准确性又无法保证检索的成功率。因此,如何有效地管理和定位 npm 包,进而提升检索效率成为了 npm 开源生态的难题。

分类作为组织和检索资源的一种有效方式^[2],成为了解决上述难题的必要手段。在为软件共享和复用而诞生的开源软件社区中,对软件的分类往往是以软件的功能作为分类类别的。这样可以对不同功能的软件进行区分管理,不仅加速了资源定位过程,而且缩小了查找范围,从而降低了检索时间。而在如今的互联网时代,尤其在 npm 这样的开源社区中,大多使用标签来实现分类所发挥的组织 and 检索资源的作用^[3]。标签作为 npm 社区中筛选信息的重要手段,以其简洁直观的描述方式对包的功能进行了概括。包与包之间因相同的标签被关联起来,也因不同的标签而被分隔开来。而在检索时,开发者不仅可以通过浏览标签快捷地获得包的描述信息,而且可以通过点击标签来得到与这个标签相关的所有的包,缩小了搜索的范围^[4]。不仅如此,对于包的创建者来说,为创建的包打上恰当的标签也提高了其被索引到的概率,可以被更多的开发者所使用。同时,多标签共存的特性得已让 npm 包展示所包含功能的多样性,便于开发者在检索时更全面

地了解包的功能。因此，npm 社区的标签机制在一定程度上解决了 npm 开源生态的难题。

尽管 npm 社区中早已存在标签机制，但是截至 2020 年，仍有超过 40% 的 npm 包没有标签。此外，由于没有明确的针对于标签的规范，很多标签被开发者主观且随意地创建，这使得标签种类繁多且没有一致性^[5]。这使得现有的标签无法达到类目划分的效果。不仅如此，从开发者检索的目标来看，软件的功能才是关注的对象，只有描述功能的标签才能达到帮助开发者分类检索的效果，然而很多标签与包的功能毫不相关。因此，npm 社区现有的标签机制是不完善的，无法充分地发挥其分类的作用，更不足以帮助开发者提升检索效率。

虽然通过人工手段可以弥补上述标签机制的不足，例如人工识别出可以作为功能类别的标签，并手动地将这些标签分配给无标签的包，完成多标签分类的过程。但是人工方式不仅耗时而且需要大量的人力资源，对于拥有海量软件包的 npm 社区来说是不现实的解决方案。在这种情况下，如何为 npm 社区打造一套描述功能的且具备一致性的标签系统，并对 npm 包自动化完成多标签分类是一个非常具有实际意义的问题，对软件工程领域具有重要的研究意义。同时，npm 社区是众多开源社区中最具代表性的社区之一，对于开源社区的研究也具有重要借鉴意义。

1.2 研究现状

1.2.1 npm 包相关研究

目前关于 npm 包的研究多集中于包的质量、依赖关系和风险性。Kyriakos C. 等人使用静态分析工具分析 npm 包，提取如可维护性和安全性等详细的质量属性，从而完成关于包质量的检测^[6]。Zimmermann 等人通过系统地分析包之间的依赖关系、负责这些包的维护者以及公开报告的安全问题来研究 npm 用户的安全风险^[7]。Magnus 等人提出一种可用于检测与事件处理相关的错误的程序表示——基于事件的调用图，用以检测 npm 包程序的编程错误^[8]。Riivo 等人对引入开源软件所造成的传递依赖风险进行了实证研究。结果表明，JavaScript 的传递依赖项的数量较去年增长了 60%，JavaScript 开发人员应该更仔细地检查软件中的依赖项有无风险^[9]。

拥有百万数量开源第三库的 npm 社区所蕴含的科研问题远不止上述研究

所涉及的方面。相较于更为流行的 GitHub 平台, npm 社区中的包背后的编程语言、应用方向和运行环境都是一致的,这使得软件间的对比、关联以及改进等关系则更易挖掘。因此,基于 npm 社区这些特性,将有更多有趣的,实用的科研问题等待挖掘。本文对 npm 包在功能上的分类问题进行了探索,提出了一种新颖的多标签分类的方法来帮助开发者识别 npm 包的功能类别。

1.2.2 开源软件分类

开源软件分类主要是利用软件仓库中的文本数据,如 Readme 文档、源代码,标签和主题等作为软件的特征,再应用传统的文本分类算法实现分类。如 Abhishek 等人提出了一种通过主题模型分析 Readme 中的描述功能性的文本,从而为开源社区 GitHub 中的项目构建类别系统的方法^[10]。Claudio 等人通过分析 GitHub 仓库的 Readme 和源码,研究了朴素贝叶斯网络在 GitHub 软件分类的应用^[11]。Zhang Yu 等人研究了 GitHub 仓库的层次分类问题,提出的方法利用异构信息网络捕捉软件仓库中的多元信息,再通过 CNN 神经网络对由关键字生成的伪文档分类^[12]。上述的研究都是传统的预先构建分类类别的方式,并且在类别上集中于编程语言,运行环境等软件外部特征,缺乏对软件内部功能上的分类研究。

除了传统的软件分类方式,还可以通过软件的标签来展示类别属性。Camilo 等人提出了 MUTAMA,一种将从软件字节码中提取的信息用于 Maven 库的多标签分类方法^[13]。Izadi 等人将 GitHub 中的 29k 个子主题映射到 228 个特色主题,通过训练 LR, DistilBERT-based 等多个多标签分类器实现项目在特色主题上的分类^[14]。Cai 等人以 StackOverFlow 上的领域知识为 GitHub 构建实体标签图(ETG),在 ETG 上使用迭代随机游走算法为软件分配合适的标签^[15]。上述研究都结合所研究的开源社区特点构建了一套与社区软件特征相符的标签或主题系统,再用各种不同的方法来实现多标签分类。此外,一些标签推荐的工作^{[16][17][18]}则是直接根据软件自身信息生成特定标签,这种方式可以看作拥有更细化类别的软件分类。标签推荐的方法虽然能更细致化地展示软件的特征,但是并不适用于以标签进行类别检索的需求。

结合软件多功能,多特征的特点,多标签分类是更符合软件分类的分类方式。虽然上述提到的研究的分类方法各不相同,但是本质上都是将开源软件的价值

的信息挖掘出来作为文本数据再进行下一步的分类工作。考虑到信息内容的多样性,如何更合理,更高效地利用这些文本数据是本文的研究重点,也是保证分类准确性的关键之处。同时,上述提到的研究中绝大多数都是关于 GitHub 社区的研究, GitHub 作为最成功和最流行的开源社区确实应该得到更多的关注。但是不同开源社区特点不同,如本文所研究的 npm 社区内的软件具有语言统一性 (JavaScript),而 GitHub 中的软件背后的语言则琳琅满目。因此,在社区中的软件特征相同点较多的情况下,更应该致力于软件最核心的功能性进行分类,才能更好地帮助开发者检索合适的软件。综上所述,本文针对于 npm 包的功能,合理地利用其文本信息,进行软件的多标签分类研究。

1.3 本文主要工作

本文通过分析 npm 社区中现有的数据,为 npm 社区构建了描述功能的类别标签,并利用文本分析技术,为没有类别标签的 npm 包进行多标签分类。主要研究内容如下:

(1) 为明确 npm 包的功能类别,发挥标签针对功能的分类作用,同时解决标签一致性的问题,本文对 npm 社区中已经存在的标签进行分析,为 npm 社区构建了 35 个类别标签。首先,对所有标签使用关联关系挖掘算法挖掘标签间的关联关系,构建标签关联关系图。其次,为归类标签,在标签关联关系图应用社区检测算法,基于标签间的关联关系将所有标签划分为若干社区。最后,基于标签关联社区,本文制定了一套类别标签识别机制,通过人工识别标签在社区中的影响力,确定了最终的类别标签。

(2) 为实现 npm 包的自动化多标签分类,本文提出了一种面向 Readme 文档的多标签文本分类方法。首先,利用有监督的主题模型 L-LDA 为每个类别标签生成关键词集。其次,将待分类的 npm 包的 Readme 文档转化为权重词集的形式。最后,利用词移距离算法计算权重词集与每个类别标签关键词的相似度,并对其进行排序,根据排序结果挑选出合适的标签,从而完成 npm 包的多标签分类。

(3) 为验证本文提出方法的有效性,本文设置了多个实验进行方法准确性和有效性的评估。具体包括:对方法内参数设定的实验以获得最佳参数,使方法准确率达到最佳;与四个多标签分类方法进行对比实验以评估方法的有效性;对

实际的没有标签的 npm 包使用本文方法以评估方法在真实情况下的有效性；对各类别标签的分类准确性进行分析。实验结果表明，本文方法可以有效地完成 npm 包的多标签分类工作。

1.4 论文结构安排

根据研究内容，本文组织如下：

第 1 章 绪论。介绍本文的研究背景与意义，相关研究工作，本文主要研究内容及文章结构安排。

第 2 章 相关理论与技术。介绍本文的相关理论与技术，包括主题模型，文本相似度和多标签文本分类三部分。

第 3 章 类别标签构建方法。阐述了构建表示 npm 包功能类别的类别标签的方法。首先介绍利用关联关系挖掘算法和社区检测算法分析 npm 社区中现有的标签，构建标签关联社区。再通过人工的类别标签识别机制识别类别标签。

第 4 章 面向 Readme 文档的多标签文本分类。首先介绍了针对于特殊文本 Readme 文档的预处理过程。其次阐述了基于关键词的多标签分类方法，主要包括类别标签关键词集的生成，Readme 权重词集以及基于词移距离算法计算相似度。

第 5 章 实验及结果分析。介绍了实验数据集的收集和评价指标，还包括三个实验，分别是：参数设定，方法有效性和各类别标签间差异性。实验结果验证了本文方法的有效性。

第 6 章 总结与展望。总结了全文的工作，并对未来工作加以展望。

本章的主要内容是介绍本文工作所涉及的相关理论技术，包括主题模型、文本相似度和多标签文本分类三个方面。

主题模型是一种常用的应用于软件工程领域的文本挖掘技术。主题模型可以挖掘出隐藏在文本中的语义结构，并把它们抽象化为主题。其中的主题被表示为单词分布，文档则是由混和的主题构成。基于主题模型提出的 Latent Dirichlet Allocation (LDA)^[19]和 Labeled LDA(L-LDA)^[20]被广泛应用于软件工程领域的任务，例如特征定位与提取。

2.1.1 主题模型 LDA

[illegible]

其中 Z 表示主题随机变量, M 、 N 和 K 分别表示文档数, 文档中的词个

数和主题数, W 表示文档中的词。随机变量 θ 为“文档-主题”分布, 随机变量 φ 为“主题-词”分布, 二者均服从狄利克雷分布。 α 、 β 分别是随机变量 θ 、 φ 的超参数。LDA 为生成概率模型, 即模型假定文档中的词都是通过先由概率选择主题, 再按照一定概率在对应主题下生成, 其具体过程如表 2.1 所示。

表 2.1 LDA 模型文档词语生成过程

LDA 模型语料生成过程描述
1. 对每一个主题 根据 $\varphi \sim \text{Diri}(\beta)$ 采样“主题-词”分布参数 φ
2. 对每一篇文档 根据 $\theta \sim \text{Diri}(\alpha)$ 采样“文档-主题”分布参数 θ
3. 对每个文档中的每个词语 根据“文档-主题”分布 $Z \sim \text{Multi}(\theta)$ 为该词采样潜在主题变量 Z 由主题变量 Z , 根据“主题-词”分布 $W \sim \text{Multi}(\varphi)$ 采样生成词 W

模型参数 θ 和 φ 常用 EM 算法^[21]或吉布斯采样^[22]来计算得到。

2.1.2 有监督主题模型 L-LDA

与 LDA 模型一样, Labeled LDA(L-LDA) 也是挖掘文档主题的统计模型。与 LDA 不同的是, L-LDA 为有监督主题模型, 即模型所针对的文档都限制于固定的主题集中, 主题则以标签的形式对文档进行标记, 每一个标签与一个主题相对应。作为有监督的生成模型, L-LDA 在主题与标签建立映射关系的基础上, 与 LDA 模型类似的过程进行文档语料的生成。图 2.2 为 L-LDA 模型示意图, 如图所示, 与 LDA 模型不同之处在于采样“文档-主题”分布 θ 时, 将主题限制在了文档标签 Λ 上, 将文档原有的主题信息嵌入到了模型中。在 LDA 模型的基础上加入了文档标签 Λ 和其先验参数 η 。L-LDA 模型文档词语生成具体过程如表 2.2 所示。

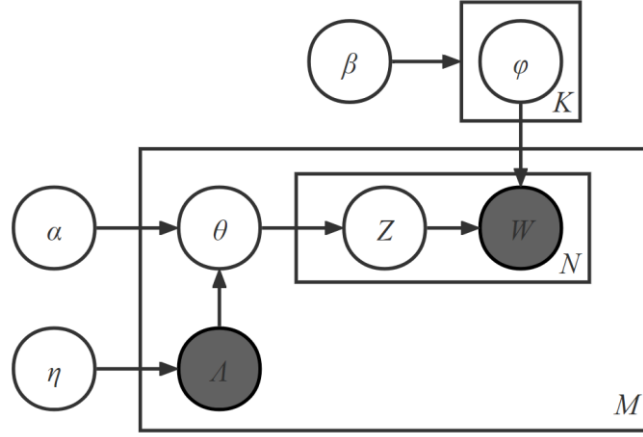
图 2.2 Labeled-LDA 模型^[20]

表 2.2 L-LDA 模型文档词语生成过程

L-DA 模型预料生成过程描述

1. 对每一个主题:
根据 $\varphi \sim \text{Diri}(\beta)$ 采样“主题-词”分布参数 φ
2. 对每一篇文档 $d \in \{1, \dots, D\}$:
对每一个主题 $k \in \{1, \dots, K\}$
生成 $\Lambda_k^{(d)} \in \{0,1\} \sim \text{Bernouli}(\cdot | \eta)$
根据 $\Lambda_k^{(d)}$ 生成特定于文档的标签投影矩阵 $L^{(d)}$, 保存文档标签信息
生成 $\alpha^{(d)} = L^{(d)} * \alpha$
生成 $\theta^d = (\theta_1^d, \theta_2^d, \dots, \theta_K^d) \sim \text{Dir}(\cdot | \alpha^{(d)})$
3. 对每个文档中的每个词语 $i \in \{1, \dots, N_d\}$
根据“文档-主题”分布 $z_i \sim \text{Mult}(\cdot | \theta^d)$ 为该词采样潜在主题变量 z_i
由主题变量 z_i , 根据“主题-词”分布 $w_i \sim \text{Mult}(\cdot | \beta_{z_i})$ 采样生成词 w_i

本文在第四章中使用 L-LDA 模型得到关于类别标签的“主题-词”分布, 并经过进一步的筛选处理得到类别标签的关键词集, 上述过程均在 L-LDA 模型的训练阶段完成。由于训练好的 L-LDA 模型可以对没有标签的文档进行标签的预测, 因此 L-LDA 模型本身就是一个经典的多标签分类方法, 本文在第五章实验部分将本文方法与 L-LDA 模型在多个多标签分类指标上进行对比。实验结果显

示，本文所提出的方法优于 L-LDA 模型。

2.2 文本相似度

文本相似度是自然语言处理领域最基础的技术之一，也是软件工程领域最常用的技术手段之一。文本相似度模型主要通过测量两个文本块在表层词汇相似度和内部语义相似度从而得到文本相似的程度。表层词汇相似度主要是基于词语匹配，主要方法有 TF-IDF^[23]、BM25^[24]、simhash^[25]等。此类方法的核心思想均为逐字比较，不会考虑单词背后的实际含义，也不会考虑上下文语义的关联。语义相似度方法则完美地补足了基于词语匹配方法的缺点，即通过捕捉单词在上下文中的语义来进行语义相似度的计算。此类方法一般是 LSA 类模型^[26]，首先通过 LSA 得到文本主题矩阵，然后利用类似余弦相似度的计算方法对文本主题矩阵进行计算。

由于基于语义相似度的方法在实际效果上优于基于词语匹配的方法，因此本文选用词移距离这一基于语义相似度算法来实现文本相似度的计算。在本文第四章第二小节中，完成类别标签关键词生成和 Readme 权重词集两部分后，需要计算关键词集和 Readme 权重词集间的相似度来最终确定预测的类别标签结果。在此过程中，本文应用词移距离算法实现该步骤。由于词移距离算法是建立在词嵌入的基础上，因此在本小节中介绍了词嵌入向量相关理论和词移距离算法的具体过程。

2.2.1 词嵌入

词嵌入是单词的一种向量表示方法，它将语料库词汇表中的每个单词映射到预定义的 n 维空间中的一组实值向量。词嵌入是根据语料库词汇中每个词在句子中的使用情况，来捕捉这些词在语料库词汇中的语义、语境和句法意义。词汇表中的每个单词都有一组独特的向量表示，同时具有相似语义和上下文意义的单词的向量表示是相似的。

目前比较经典的词嵌入模型方法有：Word2Vec 和 GloVe。

Word2Vec 来源于 Mikolov 等^[27]在 2013 年发表的论文，该算法建立在分配假设的思想。分布假说认为，出现在相似语境中的词也具有相似的语义。Word2Vec 使用这个概念将语义相似的单词在 n 维向量空间中几何接近地映射到一起。Word2Vec 通过训练一组浅层的 2 层神经网络的方法来构建单词的上下文

环境，并以一个大型的文本语料库作为输入，产生一个维度为数百的向量空间。语料库词汇表中每个唯一的单词在空间中被分配一个唯一的对应向量。Word2Vec 模型可以通过 CBOW 和 Skip Gram 两种模型实现。

GloVe 是 2014 年由斯坦福大学的 Pennington 等人开发的无监督学习算法^[28]。Word2Vec 技术依赖局部的上下文特征来生成语义向量，而 GloVe 技术则更进一步，将局部上下文特征与 LSA (Latent Semantic Analysis) 等全局特征的矩阵分解方法相结合，以捕获单词的全局语义关系。

本文在使用 WMD 词移距离算法测量文本相似度中，先得到了语料库中所有单词的词嵌入向量，训练使用的模型为 Word2Vec 模型。同时，本文所应用的 Word2Vec 模型是由 Skip Gram 模型实现的。

2.2.2 WMD 词移距离算法

2015 年，Kusner 等提出了一种基于词移动距离 (Word Mover's Distance, WMD) 的文本相似度的计算方法^[29]。该算法将文本文档视为单词的加权点云，通过计算由一个文档的单词加权点云移动到另一个文档加权点云的最小累计距离得到文档间的相似度。加权点云中单词的移动距离是利用单词的词嵌入向量计算两词之间的欧式距离。由于算法并不限制某个词所移动到的目标词，这就导致一个词就有多种的移动方案，但是需要保证该词最终选择移动的目标使得两者的距离是所有方案中距离最小的。两文档的相似度则是这些词最小移动距离的累加值。

算法具体思想如下：设文档的向量化表示为 d ，其词典大小为 n ，词 i 在文档文本中出现的次数为 c_i 。首先定义词 i 的归一化词频如公式 2.1 所示：

$$d_i = \frac{c_i}{\sum_{j=1}^n c_j} \dots\dots\dots (2.1)$$

两文档中词的移动距离为欧式距离，词 i 与词 j 的移动距离定义如公式 2.2 所示：

$$C(i, j) = \|x_i - x_j\|_2 \dots\dots\dots (2.2)$$

假设原文档为 d ，目标文档为 d' ， d 中的每个词 i 都可以全部或者部分转移到 d' 的每个词。定义一个稀疏的转移矩阵 $T \in R^n \times n$ ， T_{ij} 表示 d 中的

词 i 到 d' 中的词 j 的移动距离, 且 $T_{ij} \geq 0$ 。那么, 从 d 到 d' 的全局转移代价累加和表示为: $\sum_{ij} T_{ij} C(i, j)$ 。

这里要求出最小的全局转移代价累加和, 如公式 2.3、2.4 和 2.5 所示:

$$\min_{T \geq 0} \sum_{i,j=1}^n T_{ij} C(i, j) \dots\dots\dots (2.3)$$

$$\sum_{j=1}^n T_{ij} = d_i, \forall i \in \{1, \dots, n\} \dots\dots\dots (2.4)$$

$$\sum_{i=1}^n T_{ij} = d'_j, \forall j \in \{1, \dots, n\} \dots\dots\dots (2.5)$$

在计算过程中, 词移距离算法将文档视为单词加权点云, 而单词的加权点云实质上就是带有权重的词集, 因此十分符合本文在第四章计算类别标签关键词和 Readme 权重词集相似度的需求。上述提到的两个特殊的词集均为带有权重的词集形式, 因此词移距离算法完美地适配于两词集计算相似度的问题。

2.3 多标签文本分类

多标签文本分类是文本分类中的经典问题, 也是自然语言处理领域重要的研究问题。传统的分类问题中, 无论是二元分类, 还是多类别分类都限制了样本只能有一个类别, 且往往分类粒度比较粗。随着文本信息的日益丰富, 一篇文档的内容可能会涉及多个方面, 在分类中归属于多个类别, 这就有了多标签文本分类的概念。在多标签文本分类中, 分类粒度较细, 一篇文档可以同时拥有多个分类标签, 分类难度也会增大。本文利用 npm 包的 Readme 文档完成包的多标签分类, 实质上属于多标签文本分类问题。因此在验证方法有效性时, 评估指标选择了标准的多标签文本分类评价指标, 并与两个传统的多标签文本分类器进行了对比。

目前多标签文本分类的学习算法根据解决策略角度可以分为两大类, 分别为问题转换方法和算法适应方法^[30]。问题转换方法将多标签分类任务转换成通过其他成熟的算法解决问题的方案。BR(Binary Relevance) 算法^[31]就是典型的问题转换方法, 它不考虑标签之间的相关性从而把每一个标签当成是单标签, 并为每个标签建立一个单独的二分类器, 将多标签分类问题拆解为二分类问题。分类器链算法 (Classifier Chains, CC)^[32]在 BR 算法的基础上进行改进, 引入链式结构

解决了 BR 算法忽视了标签相关性的问题。本文选择了这两个传统的机器学习算法作为基准方法，在第五章实验部分将他们与本文方法进行了比较，实验结果显示本文方法在准确率上大幅优于这两个方法。

不同于问题转换方法，算法自适应方法直接应对多标签分类任务，采用合适的算法来解决。代表性的方法包括通过改进支持向量机 SVM 从而处理多标签问题的 Rank-SVM 算法^[33]和多标签版本的 KNN 算法^[34]。

此外，本文选用了三个多标签文本分类方法的评价指标来验证方法有效性：*Macro-F1*、*Hamming Loss* 和 *Label Ranking Average Precision (LRAP)*。*Macro-F1* 定义如公式 2.6 所示，其中 N 为标签总数， i 为标签索引。*Macro-F1* 定义为各标签 $F1$ 的平均值，在第五章实验部分在本文工作情境下做更详细的解释。

$$Macro - F1 = \frac{1}{N} \sum_{i=0}^N F1_i \dots\dots\dots (2.6)$$

Hamming Loss ^[35]是一种损失函数，表示错误标签占总标签个数的百分比，最优值为 0。其定义如公式 2.7 所示，其中 L 为标签集，包含标签数量为 $|L|$ ； D 为数据集， Y_i 为目标， Z_i 为预测结果， Δ 表示两个集合的对称差，对应布尔逻辑中的异或操作。

$$Hamming Loss = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|} \dots\dots\dots (2.7)$$

LRAP ^[36]评估分类结果中标签的排名情况，即正确标签占靠前排名标签中的百分比。*LRAP* 的取值范围在 0 到 1 之间，值越高代表分类结果越好。给定真实标签的二分类矩阵 $y \in \{0,1\}^{n_{samples} \times n_{labels}}$ 和每个标签的相关分数值 \hat{f} ，其定义如公式 2.8 所示。

$$LRAP(y, \hat{f}) = \frac{1}{n_{samples}} * \sum_{i=0}^{n_{samples}-1} \frac{1}{\|y_i\|_0} \sum_{j: y_{ij}=1} \frac{|L_{ij}|}{rank_{ij}} \dots\dots\dots (2.8)$$

其中 L_{ij} 和 $rank_{ij}$ 的定义如公式 2.9 和 2.10 所示。

$$L_{ij} = \{k: y_{ij} = 1, \widehat{f}_{ik} \geq \widehat{f}_{ij}\} \dots\dots\dots (2.9)$$

$$rank_{ij} = |\{k: \widehat{f}_{ik} \geq \widehat{f}_{ij}\}| \dots\dots\dots (2.10)$$

第3章 类别标签构建方法

标签以简洁明了的单词或词组描述 npm 包的功能和特征，方便开发者去找到满足自身需求的包。然而由于标签内容是包的创建者主观赋予，这导致很多任意创建的标签对于描述包的功能是无用的。同时，描述同一功能的标签可能会有多种表述形式，使得包的归类出现困难。因此，为了高效地展示 npm 包的核心功能，同时达成标签一致性以满足分类需求，本文提出类别标签的概念并以此为 npm 包进行多标签分类。

类别标签指能够展示 npm 包的核心功能同时起到功能类别划分作用的标签。本文首先通过挖掘 npm 社区中已经存在的标签的关联关系，构建标签关联关系图。其次，基于标签关联关系图，应用社区检测算法将标签聚类为标签关联社区。最后，使用类别标签识别机制来识别能够代表社区的类别标签。

3.1 标签关联社区

Npm 包的创建者在为自己的包打标签时，通常会用多个标签的组合来描述包的功能与特征，使得在描述上达到更详尽的效果。标签之间的组合构成了标签间的相关性，他们之间或是功能递进的关系，或是同一功能的不同特征的关系，又或者是一个包同时拥有的两个独立功能等多种不同关系。基于标签间的关联关系，标签被组织成关联网络，描述同一功能的标签在关联网络中聚集，形成标签社区。标签社区可看作同一类标签的集合，这使得不同社区分别代表着独立的功能，成为构建类别标签的重要条件。本小节首先利用关联关系挖掘算法对 npm 社区中已经存在的标签进行分析，捕捉标签间的关联关系，构建标签关联关系图。其次，为将描述相同功能的标签进行归类，对关联关系图使用社区检测算法，使得标签汇聚成若干社区，且每一个社区代表一个独立的功能。

3.1.1 标签关联关系挖掘

Npm 社区中已经存在的标签包含了开发人员对于 npm 包功能的认知，是了解 npm 包功能类别的极为重要的资源，因此本小节所有工作都是建立在这些标签之上的。其中，标签的数量以及标签间的组合同时影响了标签之间的关联关系，为准确构建标签关联关系图以及支持后续的识别类别标签的工作，本文做了如下定义：

定义 1 标签句 *TagSentence*: 对于存在标签的 npm 包 p , 所属于 p 的所有经过词干提取和词形还原的标签 $\{t_1, t_2, \dots, t_n\}$ 构成的标签集。

针对标签内容的词干提取和词形还原过程保证了标签的一致性, 避免了创作者个人习惯的影响, 优化了后续关联关系挖掘的计算。与此同时, 为排除一些独创的, 难以被大众理解的标签的干扰, 针对每一个标签统计其数量与 *TagSentence* 总量的比例, 并进行排名, 筛选掉后 10% 的标签。

定义 2 标签关联关系 $t_i \Rightarrow t_j$: 对于两个标签 t_i, t_j , 若存在由关联关系挖掘算法^[37]得到的 $Lift(t_i \Rightarrow t_j)$ 大于阈值 *threshold*, 则认定两标签构成关联关系。公式 3.1、3.2 和 3.3 为详细定义。

$$support(t_i) = \frac{\#tagSent \text{ containing } t_i}{\#tagSent} \dots\dots\dots (3.1)$$

$$confidence(t_i \Rightarrow t_j) = \frac{\#tagSent \text{ containing } (t_i \text{ and } t_j)}{\#tagSent \text{ containing } t_i} \dots\dots\dots (3.2)$$

$$Lift(t_i \Rightarrow t_j) = \frac{confidence(t_i \Rightarrow t_j)}{support(t_j)} \dots\dots\dots (3.3)$$

其中, $\#tagSent$ 表示 *tagSent* 的数量。 $support(t_i)$ 计算标签 t_i 出现在所有标签句中的次数, $confidence(t_i \Rightarrow t_j)$ 计算拥有这两个标签的标签句占只含有标签 t_i 的标签句的比例。 $Lift(t_i \Rightarrow t_j)$ 将 *support* 值与 *confidence* 值结合起来, 用以表示标签 t_i 与 t_j 相关程度, *Lift* 值越大, 两标签间的相关性越强。值得一提的是, 由于标签间的相关性代表了多种标签关系, 如之前提到的同一个包有两个描述不同功能的标签, 那么这两个标签也存在相关性。然而, 这种相关程度是远小于两个描述同一功能标签之间的相关程度。此外, 本文的研究目标是找出类别标签并将他们赋予给缺失类别标签的包以完成分类, 在挖掘标签关联关系时, 只有描述同一功能的标签所形成的关联才有助于将同一类别的标签汇聚到一起。因此, 设置阈值 *threshold* 以保留具有强相关性的标签关系, 作为关联关系。

定义 3 标签关联关系图 $G < V, E >$: 对于标签关联关系 $t_i \Rightarrow t_j$, 标签 t_i, t_j 作为图 $G < V, E >$ 中的两个以无向线段连接的两个节点。所有存在标签关联

关系的标签构成节点集 V ，它们的关联关系构成边集 E 。

标签关联关系图 $G < V, E >$ 中，两标签节点所构成的边的长度反比于他们的 *Lift* 值，即两标签关联关系程度越强，边长越短，关联程度越弱，边长越长。此外，为方便观察，将每个标签节点标注上标签名称，且节点大小以及对应标签尺寸正比于节点的度。即若一个标签与其他标签的关联数量较多，那么其对应节点在图中显示较大，标签尺寸也较大。图 3.1 为标签关联关系图示例。

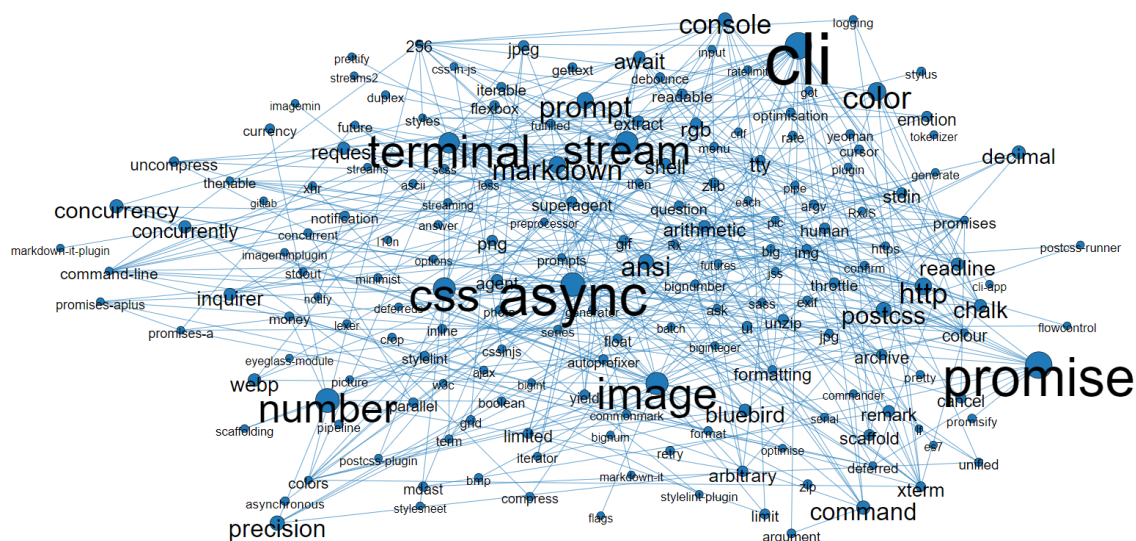


图 3.1 标签关联关系图示例

3.1.2 标签社区检测

标签关联关系图展示了标签间关联的紧密程度，并且经过观察发现关联紧密的标签间呈现出表示相同功能的趋势，而距离较远的标签描述的功能类别差异较大。然而，复杂的网络让人难以从整体角度捕捉标签间的功能类别关系，尤其是相同功能标签间的归纳和不同功能标签间的区分，因此本小节利用社区检测算法^[38]对标签关联关系图中的标签进行处理，使标签基于关联关系进行聚类，达到更好的观察效果。

定义 4 标签关联社区 *Tag Community*: 标签关联关系图经由社区检测算法所划分出的社区。为更好地观察, 对各个社区用以不同颜色进行区分, 示例如图 3.2 所示。

标签关联社区内各标签间连接紧密,关联程度高,绝大多数是描述相同功能

的标签。不同标签关联社区间的标签连接稀疏，关联程度弱，所描述的功能也相差较远。例如，在图 3.2 中，最下方的淡粉色标签关联社区中都是与图片相关功能的标签。而在上方与它相邻的两个社区，一个是与前端 CSS 相关功能的社区，另一个是与命令行相关功能的社区。通过观察聚类成社区的标签，研究者能够辨别该社区所描述的功能，这样一来，标签关联社区直观地展示了 npm 包的各个功能类别。

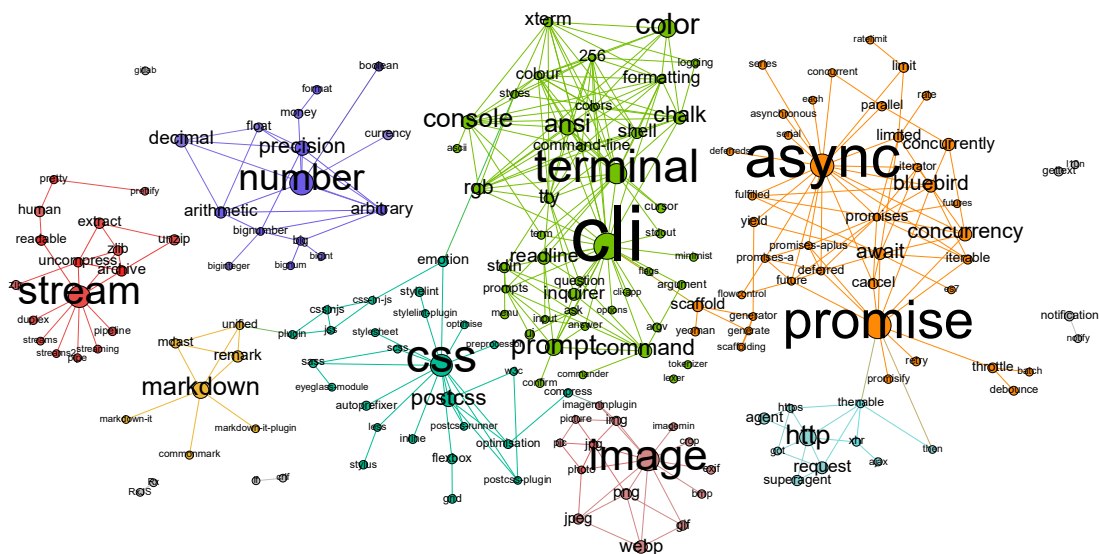


图 3.2 标签关联社区示例

本文选用 npm 社区中被依赖次数最多的 8000 个包作为样本来构建标签关联关系图，并从中划分出 169 个标签关联社区。为确保社区的划分质量，在划分社区的同时采用模块度来衡量社区质量。对于无向无权重的网络，模块度的取值范围是 $[-0.5, 1]$ ，模块度越大，社区划分质量越高。通常认为模块度在 0.3 到 0.7 就会出现明显的社区结构^[39]。本文所形成的社区模块度为 0.916，已经达到了非常好的社区结构。

3.2 类别标签识别机制

通过标签关联社区可以识别出与社区内标签相关的特征。如此一来，一个描述功能的标签关联社区可以表示一个独立的功能类别。然而，由标签关联关系图得到的所有标签关联社区并不能完全对应本文所研究的 npm 包功能的类别。这是因为 npm 包功能的类别是描述 npm 包的核心功能，具有高度概括的性质且相互独立。而生成的标签关联社区中，有些并不是关于功能的，例如与开源代码

平台 Github 相关的标签会形成社区；有些是功能相关的，例如与数据库功能相关的两个细化功能社区 SQL 和 Database driver。这表明研究方法得到的标签关联社区不是完美的，需要进一步的进行筛选和合并，才能与 npm 包的功能类别对应起来。

基于标签关联社区，本文采用人工筛选合并的方法，得到最终能够代表功能类别的若干社区。本文利用卡片分类法完成标签关联社区的筛选和合并过程：挑选具备编程经验的研究生 5-7 名作为参与者。将每个标签关联社区进行编号并看作一张卡片，由参与者自行对标签关联社区进行归类。首先做二分类的封闭式卡片分类，即将与描述功能类别无关的社区归为一类，相关的社区归为一类，对应于筛选社区的步骤。其次是对描述功能类别相关的社区进行开放式卡片分类，即以独立的功能为评判标准，若多个社区描述的功能相关或相似则将他们归为一类，否则自成一类，对应于合并社区的步骤。整理分析所有参与者的分类情况，采用卡片-分组矩阵归类法，将社区编号作为纵轴，横轴为各参与者所有的归类情况，然后把参与者的选择以百分比的形式标示出来。最终，综合选择数值最大的归类情况作为标签关联社区筛选合并的结果。在上一节所构建的 169 个标签关联社区中，经过筛选合并后，最终得到了 35 个代表功能类别的社区。

得到经过筛选合并后的类别关联社区后，从中挑选最具代表性的标签作为类别标签。类别标签指能够展示 npm 包的核心功能同时起到功能类别划分作用的标签。根据类别标签的特性，本文从两个方面来衡量标签的“代表性”：1.与社区内其他成员标签的关联数量。标签关联社区本质上是聚类簇，一个标签与社区内其他成员标签的关联数量越多（表现为节点的度），它成为聚类中心的可能性越大。作为聚类中心的标签在社区中无疑是具有代表性的。2.标签在包描述语句中出现的频次。Npm 包除了标签能够展示包的特性，描述语句（Description）同样也是重要的文本信息。包的描述语句通常为一句话，概括性地介绍包的相关功能。经过研究发现，描述核心功能的标签大概率会被使用在描述语句中，即出现内容重叠。这种现象本质上是对相同内容的强调，着重描述内容的重要性。因此，出现重叠现象的标签同样体现了其代表性。综合以上两点，设定参数 *Influence*，用以衡量标签在社区中的影响力，挑选能够代表社区的标签，定义如公式 3.4 所示：

$$Influence(t_i) = degree(t_i) * \frac{\#description\ overlapping\ t_i}{\#tagSent\ containing\ t_i} \dots\dots\dots (3.4)$$

其中， $degree(t_i)$ 为标签关联社区中标签 t_i 的度； $\#description\ overlapping\ t_i$ 与标签 t_i 重叠于所在包描述语句的数量； $\#tagSent\ containing\ t_i$ 为包含标签 t_i 的标签句数量。对于每一个标签关联社区，选择社区中 $Influence$ 值最大的标签作为该社区的类别标签。

从前文得到的 35 个类别标签关联社区挑选出的类别标签如表 3.1 所示。

表 3.1 类别标签

类别标签
number, string, time, collection, text, promise, stream, security, debug, error, fs, http, url, util, cli, documentation, image, loader, log, validation, test, database, framework, authentication, email, parser, typescript, markdown, event, network, polyfill, template, ast, css, cache

3.3 本章小结

本章通过分析 npm 社区中已经存在的标签，构建描述功能类别的类别标签。首先利用关联关系挖掘算法构建标签关联关系图，其次使用社区检测算法将标签关联关系图中的标签聚类成若干社区，最终通过人工筛选合并的类别标签识别机制确定最终的类别标签。同时，选用 npm 社区中被依赖次数最多的 8000 个包作为样本来应用上述方法，得到了 35 个类别标签。

第4章 面向 Readme 文档的多标签文本分类

自述文档 Readme 从 npm 包的功能、特征以及使用方法等方面做了全面的描述。阅读 Readme 文档是深入了解 npm 包的最重要渠道。开发者在搜寻到感兴趣的 npm 包时，往往都会去详细阅读 Readme 文档来进一步了解包的功能，以确保符合自身的需求。而对于没有标签的，或是缺失描述功能的标签的 npm 包，Readme 文档是他们介绍自身功能的唯一途径。因此，本文通过挖掘 Readme 文档中关于功能描述的关键信息，来为 npm 包打上类别标签，以实现针对于功能分类的目标。

本章介绍了利用 Readme 文档实现 npm 包多标签分类的具体步骤，分为两个方面：1. Readme 文档中信息混杂，为提高方法准确率，本文首先对 Readme 文档进行预处理，包含内容切分和文本预处理两部分。2. 受人们在快速阅读文章时通过关键词区分文章类别这一过程的启发，本文提出了基于关键词的多标签分类方法。首先利用有监督的主题模型 L-LDA 获得关于类别标签的关键词集。其次对给定的待分类的 npm 包，转换其 Readme 为权重词集。最后应用词移距离算法计算权重词集与各类别标签关键词集的相似度，并将结果进行排序，从中选取标签完成分类。

4.1 预处理

Readme 文档从各个角度向读者介绍了 npm 包的详细信息，其中既包括最为关键的功能简介和使用方法，也有相对次要的参考引用、贡献人员以及协议等。多样的内容使开发者们可以全方位地掌握关于 npm 包的信息，然而对于只想了解功能的开发者们来说内容的多样同时也是一种负担。与此同时，由于本文研究目标在于 npm 包的功能分类，作为分类方法输入文本的 Readme 文档，其中与描述功能无关的部分是无意义的，冗杂的文本甚至会降低算法的精度。因此，本小节首先对 Readme 文档在内容上进行切分，保留关于功能描述的信息，并为了后续工作的使用，对文本进行预处理。

4.1.1 Readme 内容切分

Christoph 等人^[40]的研究将 Readme 文档内容分类成了 8 个方面，分别是‘What’（项目的简介），‘Why’（项目的优势），‘How’（项目的使用方法），

‘When’（项目的版本），‘Who’（项目的创始团队），‘References’（项目的参考引用），‘Contribution’（项目的贡献指南）以及 ‘Other’（其他）。8 个方面中，‘What’ 和 ‘How’ 分别介绍了软件整体上的功能和使用方法，是所有内容类目中包含功能描述最多的两个部分，也是占 Readme 全文比重最大的两个部分。此外，‘Why’ 类目中在阐述项目优势的同时也会提及项目的功能，而其余类目则很少包含描述功能的信息。因此，为提升方法准确率，降低计算成本，本文对 Readme 文档进行内容切分，只保留与功能描述相关的文本内容作为后续方法的输入文本。

Readme 文档通常以结构字段进行章节组织，并以非结构化文本作为章节内容。作为章节标题的结构字段在起着组织作用的同时，也概括了内容文本的主题。举例来说，标题为 ‘Feature’ 的章节，其文本内容为描述软件的特征和优势，对应于前述分类内容的 ‘Why’；标题为 ‘Contribute’ 的段落描述的是如何规范化地贡献代码到项目中，对应于 ‘Contribution’ 类目。如此一来，某些特定的分类内容与章节形成了映射关系。此外，由于构成标题的结构字段为简洁的单词或短语，这导致标题一般具有通用性，如刚提到的 ‘Contribution’ 类目所对应的章节标题普遍为 ‘Contribute’，‘Contributor’ 等同词根单词。因此，本小节利用章节标题与内容类目的映射关系对 Readme 文档进行内容切分处理，从而进行特定部分的提取。

由于 Readme 文档中章节多，信息冗杂，不容易直接把本文关注的 ‘What’，‘How’ 和 ‘Why’ 对应部分抽取出来。而与描述功能无关的部分比重小，内容针对性强，标题特征明显，因此剔除与描述功能无关的部分从而留下关注的信息文本是更为有效的途径。首先针对需要剔除的部分进行实验调查，以确定这些部分的标题是否有规律字段与其形成映射关系，具体步骤如下：选取 3 名吉林大学软件工程梯队硕士研究生，并在 npm 社区中随机抽取 1000 个 npm 包的 Readme 文档作为样本。实验要求 3 名实验人员详细阅读样本，人工判断哪些是与功能描述无关的内容，记录该内容所在章节的标题。将 3 项个人调查结果进行汇总，取交集作为最终的实验结果。其次，对每个由实验得到的标题进行分词，词形还原以及词干提取，得到关于标题的 26 个关键词词干，如表 4.1 所示。基于这些标题的关键词词干，对给定 Readme 文档的章节标题进行遍历，剔除掉包含关键词

词干的章节，留下的即为与功能描述相关的文本。

表 4.1 不相关标题关键词词干

不相关标题关键词词干
<p>contribut, copyright, author, contact, maintain, provid, backer, tank, note, licens, develop, communiti, contributor, author, set, test, start, peopl, document, changelog, credit, option, tabl, pattern, sponsor</p>

与此同时，为了后续的研究，本文对经过筛选得到的与功能描述相关的文本进行了更细致地划分，分割为简介部分和用法部分。图 4.1 展示以 npm 包 ‘nock’ 的部分 Readme 文档为例展示了内容切分结果。



图 4.1 Readme 内容切分示例

简介部分为纯文本，概括介绍 npm 包的功能。用法部分文本与代码相结合，详细介绍功能用法和优势特点。具体划分规则如下：

- 简介部分：内容为包在功能上的定义，对应于 ‘What’ 类目。此部分分割原则为：Readme 起始段以及标题中包含 ‘Introduction’，‘Description’ 以及 ‘Overview’ 的章节。

- 用法部分：Npm 包的用法说明，内容为包的使用指导和特点，对应于‘How’和‘Why’类目。此部分分割原则为：在功能描述相关的文本中除简介部分的所有内容。

4.1.2 文本处理

获得 Readme 文档中的描述信息之后，本文将对这些信息文本进行预处理以支持后续工作。Readme 文档作为指导性说明文档，并没有制定标准的书写规范，这导致其中混杂着多种形式的信息，例如：链接、表格和图片等。除纯文本外的不同形式的信息对本文的方法是无效的，甚至可能会干扰方法的准确率，因此需要剔除这部分内容。此外，经过上述处理过后的 Readme 文档依然为文本格式，为满足机器学习方法的输入，本文应用标准的文本清理流程，包括分词处理，词形还原。与此同时，即便是纯文本中也包含了许多噪声词，例如：停用词、非英文词汇等，也需要去除。上述对 Readme 文档的预处理过程详细步骤如下：

步骤 1：移除无用部分

无用部分包括链接、图片以及 html 格式的文本。由于 Readme 文档格式为 markdown，所有格式均可以转化为特定的字符串格式，如图片是以字符串‘’开头，后面再加上图片链接。这样一来，只需将无用部分的字符串格式进行整理，使用正则表达式，将无用的部分移除。制定的正则表达式如表 4.2 所示

表 4.2 需移除内容的正则表达式

内容	正则表达式
链接	((http ftp https):/)(([a-zA-Z0-9\._-]+\.[a-zA-Z]{2,6}) ([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))(:[0-9]{1,4})*/([a-zA-Z0-9&%\._/~-]*)?
图片	\[(?!\[(*?)\])([*?])\])([*?])\ 和 \(* - +)\ \[([*?])\]([*?])\)
html	</?\w+[^>]*>
非英文字符	[^A-Za-z \n[(*!@;'\?)]*
指令	\\$.*
标点	[^\w\s]

步骤 2: 分词和词形还原

移除无用部分后的 Readme 文档为纯文本格式, 首先对其进行分词处理。其次, 对每个词进行词性标注, 词形还原。此步骤所用的词典来自经典自然语言处理词库 WordNet。

步骤 3: 移除噪声词

噪声词包括停用词、非英文单词以及一些频繁出现的公有词汇。停用词通过词典进行识别移除, 非英文单词通过正则表达式移除。而对于频繁出现的公有词汇, 例如 npm, JavaScript 等, 通过手动建立词典的方式进行移除。

此外, Readme 文档中的代码虽然也是文本, 但是与传统文本不同的是, 代码中所包含的有价值的信息比重小, 以函数名、类名, 变量等特殊标识符和注释的形式存在。因此, 对于 Readme 文档中的代码, 仅保留上述提到的有价值的部分, 并对其使用预处理步骤。

4.2 基于关键词的多标签分类方法

当面对文档分类任务时, 人们可以根据某个类别的几个相关关键词, 快速区分文档是否属于该类别。这是因为人们可以通过类别的关键词与文档内容建立相关性。举例来说, 在人们大量阅读篮球杂志之后, 就会意识到涉及到“三分线”关键词的文章都是在讨论篮球的, 即使他们并不清楚“三分线”是篮球场上的哪一条线。这其中根本的原因在于人们在文章中总是能见到“三分线”与“上篮”, “投篮”等词汇共同出现, 脑海中已经把“三分线”归为与篮球相关的词语, 当新的文章中再次出现该词时, 很快就能与篮球构建联系, 最终成功分类。同理, 当开发者阅读 Readme 文档时, 根据自身对于不同功能类别的理解, 捕捉其中的关键词, 与类别建立联系, 自动完成分类。

受上述过程的启发, 本文提出了一种基于关键词的多标签分类方法, 来模拟这一过程。首先, 为获得关于功能类别也就是类别标签的全面理解, 使用 L-LDA 主题模型^[20]为每一个类别标签生成其关键词集。其次, 对于待分类的 npm 包, 根据每个词在 Readme 文档中不同部分出现的位置和频次, 将 Readme 文档转化为带有权重的词集。最终, 为构建两个词集之间的联系, 通过计算词移距离的方式得到 readme 权重词集与每一个类别标签关键词集的相似度。在对相似度进行排序后, 选出合适的类别标签, 完成多标签分类。

4.2.1 类别标签关键词生成

如前文提到的，人们不断扩充对某一领域或类别的认知关键词集，是通过阅读大量的相关文献并根据词的出现频率和共现关系学习到的。这一相关性学习过程类似于有监督的主题模型 L-LDA 的训练过程。L-LDA(Labeled LDA) 在经典主题模型 LDA 的基础上进行改进，训练样本改为有标记的文本数据并把标签与主题一一对应，使得模型中的两个重要参数：文档-主题分布和主题-词分布中的“主题”皆为已知标签。L-LDA 模型的训练过程为：

L-LDA 训练过程

1. 将语料库中的每一种标签对应一个主题，构建主题集合；
2. 对语料库中的每篇文档中的每个词 w ，随机的赋予一个该篇文档对应的主题 t ；
3. 重新扫描语料库，对每个词 w ，使用吉布斯采样 (Gibbs Sampling) 对其采样，求出它的 topic，在语料中更新；
4. 重复步骤 3，直到 Gibbs Sampling 收敛；
5. 统计语料库的 topic-word 共现频率矩阵。

在训练好的 L-LDA 模型中，根据 topic-word 共现频率矩阵，计算得到每一个主题的主题-词分布 (topic-word distribution)，如图 4.2 所示。

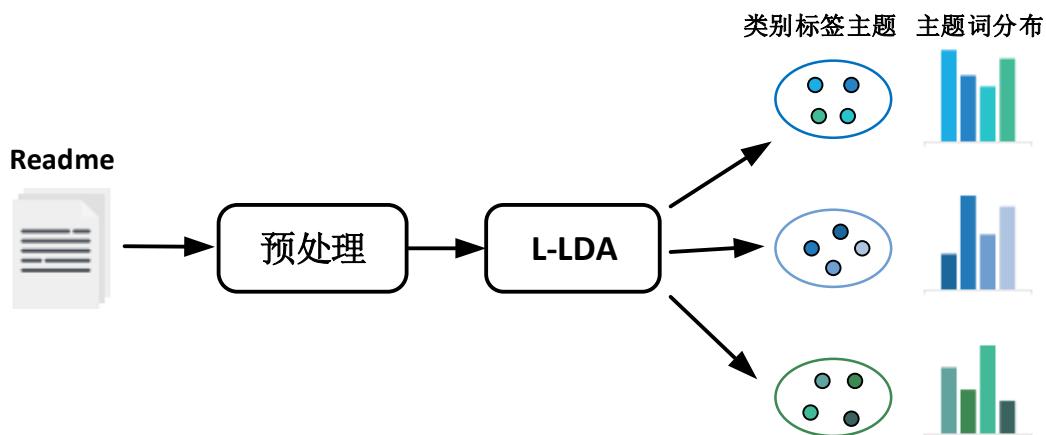


图 4.2 L-LDA 模型生成类别标签关键词过程

主题-词分布可以理解为一个主题下的带有相关性权重的词集，词对应的权重越大，该词与主题的相关性越强。由于主题与标签对应，因此主题-词分布即为关于标签的带有权重的关键词集。综上所述，本小节利用 L-LDA 模型生成类别标签的关键词集。

从主题-词分布获得启发，本文为类别标签所构建的关键词集加入了权重的概念，即类别标签下的每一个关键词都带有其对应的权重，权重值越高代表该关键词对于其所属类别标签重要性越高。这与主题-词分布中词的概率相似，主题中词出现的概率越高，那么该词与所在主题的相关性越强。因此，本文将主题-词分布与类别标签关键词集对应起来，主题即为类别标签，词分布中的词作为关键词，词对应的概率作为关键词的权重。

此外，训练样本是否被正确标记极大地影响了 L-LDA 模型的训练结果。因此，为保障模型的训练质量，本文所选取的样本遵循如下原则：作为样本的 npm 包被标记上类别标签的同时，且其描述（Description）中必须出现相同的类别标签。这样的做法保证了类别标签并不是包的作者随意标记上去的，即最大程度上确保了样本标记的正确性。

在得到训练好的主题-词分布之后，在每个主题下选择前 50 个词作为关键词集的候选词，为保证关键词集的准确性和互斥性，手动对候选词做如下筛选过程：

1. 移除无意义的公有词汇，例如：“find”，“version”，“make”等。
2. 与类别标签相同的词只允许出现在自己的关键词集中，确保了关键词间的界限。

最终，筛选结束并经过实验 RQ1 确定参数，对每个类别标签下的候选词按权重值从大到小的排序选取前 35 个构成最终的关键词集，并归一化词集的权重。

4.2.2 README 权重词集

对于待分类的 npm 包，它们的 README 文档作为分类方法的输入文本，需要进一步转化成特定的词集以用于后续的计算。由于 README 文档被分为了两个部分，简介部分相较于用法部分在功能性描述的内容上占比更大，导致同等文本长度下，简介部分对于分类方法的贡献更多。此外，单词出现的频次反应了该词的重要程度，也需要在形成的词集中体现。因此，本文依据单词出现在 README 文档不同部分的位置和频次，为单词分配权重，将待分类的 npm 包的 README 文档转化为带有权重的词集，以合理地改进算法精度。

定义 5 单词得分 S_w : 由单词 w 出现在 Readme 文档的位置和频次计算得到的分数。

$$S_w = \sum_{p \in P_r} TF_p(w) * Weight_p \dots\dots\dots (4.1)$$

其中, P_r 表示 Readme 文档简介部分和用法部分的集合, p 表示 Readme 文档中的一个部分, $TF_p(w)$ 表示单词 w 出现在 p 部分的频次, $Weight_p$ 表示 p 部分设定的权重值。由于 Readme 文档的不同部分文本长度不同, 功能性描述内容占比不同, 对分类的贡献也不同。因此设置参数 $Weight_p$, 对不同部分分配不同的权重值, 以考虑单词出现在不同位置的因素。因为参数 $Weight_p$ 影响分类的准确率, 所以两部分的最终参数值由实验确定, 实验得到的结果为简介部分的权重为 0.9, 用法部分的权重为 0.1。

得到每个单词的单词得分后, 将分数值归一化为权重, 如公式 4-2 所示。至此, 待分类 npm 的包的 Readme 文档被转化为权重词集 $\{w_1:T_1, w_2:T_2, \dots, w_n:T_n\}$ 。

$$T_w = \frac{S_w}{\sum_i^n S_i} \dots\dots\dots (4.2)$$

4.2.3 基于词移距离计算相似度

给定类别标签的关键词集和待分类 npm 包的 Readme 权重词集, 本文利用词移距离算法计算两种词集间的相似度来完成多标签分类工作。

设待分类的 npm 包 p 的 Readme 权重词集为 R_p , 类别标签 c 的关键词集为 K_c , 则 R_p 和 K_c 的词移距离计算过程描述如下:

定义 6 词移动代价 (word travel cost) c : 分属不同词集的两个单词间的欧式距离。

$$c(i, j) = \|v_i - v_j\|_2 \dots\dots\dots (4.3)$$

其中, 单词 i 取自 R_p , 单词 j 取自 K_c , v_i 和 v_j 为两词的向量表示。设 d_i 为词 i 在 R_p 中的权重值, d_j 为词 j 在 K_c 中的权重值, $T_{ij} > 0$ 表示词 i 移动到词 j 所耗费的权重值。由于词 i 可以移动到 K_c 中的任意一个词, 只需要保证移动耗费的权重和等于自身权重即可。与此同时, 词 j 也需要保证所有

移动到自己的词所耗费的权重和等于自己的权重。约束表示如下：

$$\sum_{j=1}^n T_{ij} = d_i \quad \forall i \in \{1, \dots, n\} \dots\dots\dots (4.4)$$

$$\sum_{i=1}^n T_{ij} = d'_j \quad \forall j \in \{1, \dots, n\} \dots\dots\dots (4.5)$$

定义 7 词移距离 (word mover's distance) d : 将 R_p 中的所有词移动到 K_c 的最小累积成本。

$$d_{pc} = \min_{T \geq 0} \sum_{i,j=1}^n T_{ij} c(i,j) \dots\dots\dots (4.6)$$

图 4.3 展示了本文中 WMD 算法示例。蓝色球代表了 R_p 中的词，黄色球代表了类别标签 fs 的类别关键词集中的词。图中的线表示了词的移动，线上的数值是移动所消耗的权重。不难发现，蓝色球移动消耗的权重等于自身的权重值，黄色球流入的权重也等于自身的权重值。在有限的词移动方案中，累积消耗权重最小的词移动代价和为最终的词移距离值。

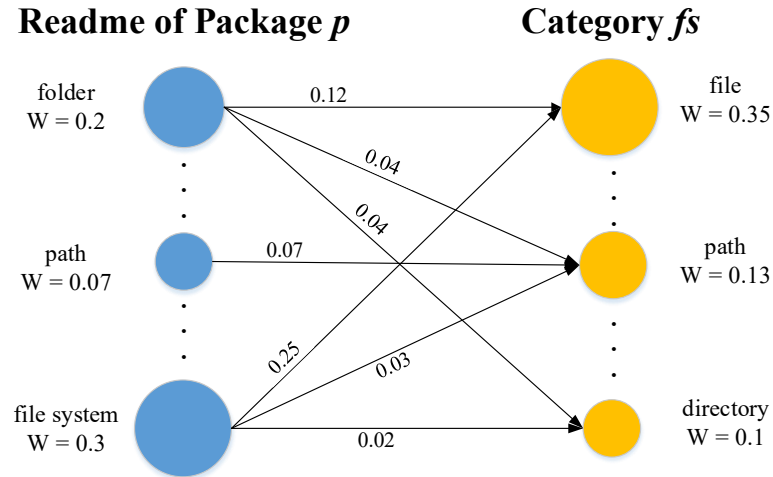


图 4.3 词移距离算法示例

词移距离展示了 npm 包的 README 权重词集与类别标签之间的相似度，词移距离值越小，两词集间的相似度越高。对于待分类的 npm 包 p ，使用 WMD 算法计算它与 35 个类别标签的相似度，并按值由小到大排序，则排名越靠前意味着包 p 与对应标签的相关度越大。

此外，本文分析了 npm 社区中所有被打上类别标签的 152,200 个包，发现有 64.9% 的包只拥有一个类别标签，33% 的包拥有两个标签，而拥有超过两个

类别标签的包的情况只占有所有情况的 2.1%。因此，本文提出的方法将只为待分类的 npm 包分配一个或两个类别标签。对待分类的包 p ，给定其词移距离排序列表 $l = \{c_1:d_{pc_1}, c_2:d_{pc_2}, \dots, c_n:d_{pc_n}\}$ ，其中 c_i 为类别标签， d_{pc_i} 是对应的词移距离值，则本文方法为其计算得到的标签结果为 Tag_p ，如公式 4-7 所示：

$$Tag_p = \begin{cases} c_1 & d_{pc_2} - d_{pc_1} > d_{pc_3} - d_{pc_2} \dots\dots\dots (4.7) \\ c_1 \text{ and } c_2 & otherwise \end{cases}$$

4.3 本章小结

npm 包的描述文档 Readme 蕴含了包的功能信息，分析 Readme 文档可以实现对 npm 包的多标签分类。由于 Readme 文档中涉及内容较多，所以本章首先根据 Readme 文档内容对其进行预处理，只保留描述功能的部分，并进行了标准的文本处理流程。同时，根据人们快速区分文档类别的方式，本章提出了一种基于关键词的多标签分类方法。首先，通过 L-LDA 主题模型为类别标签生成关键词集。其次，将待分类的 npm 包的 Readme 文档转化成带权重的词集。最终，利用词移距离算法计算两种词集间的相似度，并按相似度进行排序，从中挑选与包最相符的类别标签。

第 5 章 实验及结果分析

为了验证本文方法的有效性,本章围绕以下三个问题进行了一系列的实验研究:

RQ1: 本文方法的最佳参数设定是什么?

RQ2: 本文方法能否有效地为 npm 包打上类别标签?

RQ3: 本文方法针对各类别标签的分类效果如何?

5.1 实验数据及评价指标

由于本文所研究的是基于文本分析的 npm 包面向功能的多标签分类问题,并没有已有的数据集来进行实验。因此,本文通过对 npm 的数据库¹进行遍历,查找出高质量的拥有类别标签和 Readme 文档的 npm 包作为实验的数据集。Npm 官方提供了针对 npm 包的评分系统,主要包含三个方面:流行度(popularity),品质(quality)和维护度(maintenance)。Npm 社区综合这三个分数来为搜索页中的各个包排序,分数越高,排名越靠前,同时也意味着包的整体质量靠前。所以,在保证拥有类别标签的基础上,本文所收集的 npm 包中需保证三个指标的综合得分超过 60,最终获得了 23.8 K 个包。由于本文方法中的 L-LDA 模型为有监督模型,需要训练数据完成模型的训练。因此本文采用十倍交叉验证法,将数据集分成 10 份,其中 8 份作为训练集,2 份作为测试集。与此同时,在训练集中划分出十分之一的数据作为 RQ1 参数设定的验证集。此外,为验证本文方法对真实情况中没有类别标签的包的有效性,随机抽取了 100 个高质量的并有 Readme 文档的无标签 npm 包作为 RQ2 的部分数据集。

本文采用标准的多标签分类评价指标从不同角度来评估本文方法,具体包括:

$$Macro-F1 = \frac{1}{N} \sum_{C=1}^N \frac{2 \times TP_C}{2 \times TP_C + FP_C + FN_C} \dots \dots \dots (5.1)$$

其中, N 为类别标签个数, TP_C 表示本文方法的分类结果与真集中的标签都为标签 C 的包个数; FP_C 表示本文方法得到的标签为 C 但是真集中所对应的包不为 C 的包个数; FN_C 为真集中包的类别标签为 C 但是本文方法应用到对应的包得到的结果不是 C 的包的个数。本章在 RQ1 和 RQ2 中使用 *Macro-F1* 作为评价指标。

¹ <https://registry.npmjs.org/>

Hamming Loss 和 *Label Ranking Average Precision (LRAP)* 同样是多标签分类方法的评价指标。*Hamming Loss* 是方法分类的错误标签个数占总标签个数的比，它是一个损失函数，最优值为 0。*LRAP* 评估标签的排名，即排名较高的预测标签中有多少是正确的标签。其值在 0 和 1 之间，值越高，方法性能越好。这两个指标用于 RQ2，具体定义见第二章第三小节。

本文实验的软件环境是 windows 64 位操作系统。实验代码基于 python 语言编写，集成开发环境为 Pycharm。

5.2 参数设定 (RQ1)

4.2.1 小节中获取类别标签关键词集的关键词数量时和 4.2.2 小节中 Readme 文档转化为 Readme 权重词集时，用法部分和简介部分的权重分配都影响了本文方法的准确性。因此，在本小节使用数据集中的验证集进行实验，取得使本文方法达到最佳性能的两个参数。

每个类别标签的关键词集负责描述类别标签。本文将关键词数量的范围设定在 10 到 50 之间且倍数为 5 的值，组成 9 组。Readme 文档两部分权重和为 1，则两部分的各自权重值变化范围为 0 到 1。为此，本文设置简介部分的权重由 0 增加至 1，每次增幅为 0.1。用法部分的权重则由 1 降至 0，每次降幅 0.1，过程始终保持权重和为 1，共构成 11 组。所有参数共同影响方法的性能，所以本文对所有参数的组合进行实验以测得最佳的参数设定。如此一来，一共有 99 组组合，表 5.1 展示了 8 组代表性数据。

表 5.1 参数组合实验

关键词数量	简介部分权重	用法部分权重	Macro-F1
10	0	1	0.322
30	0.3	0.7	0.526
30	0.5	0.5	0.541
30	0.9	0.1	0.587
35	0.5	0.5	0.556
35	0.9	0.1	0.592
35	1	0	0.574
40	0.9	0.1	0.587

实验结果表明，当关键词集中关键词的个数为 35，简介部分和用法部分的权重分别为 0.9 和 0.1 时，本文提出的方法性能最佳。观察 Readme 文档的权重分配相同，关键词数量不同的组，可以发现关键词的数量变化对于方法的影响较小。对于关键词数量相同的组，Macro-F1 值随着简介部分的权重的增加而增加。然而当简介部分的权重值为 1 时，即输入文本中全部为简介部分时，准确率反而有所降低。因此，简介部分和用法部分对于本文的方法而言都是必不可少的，同时简介部分包含了比用法部分更多的关于功能类别的信息。

5.3 方法有效性评估（RQ2）

为验证本文方法的有效性，本文首先与四个多标签分类方法进行了对比实验，其次在无标签 npm 包的数据集下评估了方法的分类准确率。

在对比实验中，本文所选取的四个多标签分类方法分别为：

- 二元关联 BR (Binary Relevance) [31]: BR 是一种基于标签是独立的假设的分解方法。它将每个标签视为一个单独的分类问题，并为每个标签训练一个分类器进行预测。
- 分类器链 CC (Classifier Chains) [32]: CC 是基于 BR 方法的，它也使用了 n 个二进制分类器。在 CC 中，所有分类器都链接在一个链中。链中的二进制分类器是建立在前一个分类器的预测之上的。
- L-LDA: L-LDA 作为有监督的主题模型，根据训练好的模型可以对新文档推理主题，即为多标签分类过程。
- 关键词匹配 Keyword Matching: 利用本文方法生成的类别关键词与 Readme 文档内容直接匹配，将匹配成功的词汇总按数量排序，生成排序列表，以本文方法相同的方式选取标签。

注意两个多标签分类器分别需要实例化一个基分类器，本文选择支持向量机 (SVM) 作为基分类器。以上四种方法的训练数据与本文方法相同。

表 5.2 展示了实验结果。从表中可以观察到，本文方法在三个评价指标上都优于基线方法: *Macro-F1* 平均提高了 29%，*Hamming Loss* 平均提高了 42%，*LRAP* 平均提高了 44%。

表 5.2 对比方法评估结果

方法	评估指标		
	<i>Macro-F1</i>	<i>Hamming Loss</i>	<i>LRAP</i>
BR	0.431	0.068	-
CC	0.447	0.067	-
L-LDA	0.470	0.054	0.071
Keyword Matching	0.469	0.055	0.107
本文方法	0.587	0.035	0.123

BC 和 CC 是 n 个二进制分类器的组合，它们将任务转化为多个二进制分类器。它们生成的预测结果包含更多的标记，这导致不正确的标记也相对较多。L-LDA 和关键词匹配都是统计模型，因此它们严重依赖于词频，忽略了词与词之间的语义。本文提出的方法通过词向量捕捉单词的语义，并计算 Readme 文档中的单词与类别标签的关键词之间的相似性。因此，它可以使 Readme 文档中的所有单词在语义层面上与关键词建立关系。关键词匹配只关注关键词，忽略了其他词的信息。L-LDA 虽然考虑了所有的单词，但不能捕获单词语义层面的相关性。

此外，对于一些包被归类为错误标签和遗漏了恰当标签的情况，通过观察分析这些数据，本文总结出三个主要原因：1) 这些包的 Readme 文档内容不够，使得 Readme 文档中的每个词都获得了相近的权重值。2) 一些不重要的词出现得太频繁，以至于让这些词获得了过多的权重值。3) 一些关键词被多个类别标签共享。例如，关键字“server”出现在 10 个类别标签关键词集中。

为验证本文方法对实际的没有标签的 npm 包的有效性，选用无标签包数据集进行实验。首先将该数据集输入到本文方法中，得到分类结果。其次，为判定分类结果的正确性，本文采用人工阅读包的 Readme 文档并运行相应包的样例来确定方法预测的结果是否正确。实验结果如图 5.1 所示。100 个随机抽取的 npm 包中，64 个包被预测正确了至少一个类别标签，准确率达到了 64%。这其中有 55 个包完全预测正确，9 个包部分正确，完全预测正确的比例远高于部分正确的比例。36 个包预测错误，这其中有 2 个包的真实标签不在本文所构建的类别

标签中，分别是关于数据库驱动的‘neo4j-driver’和关于 benchmarking 的‘@c4312/matcha’。然而，这两个功能在 npm 生态系统中是稀少的，它们是包的作者定制化的标签以更详细地描述它们的包。而本文方法专注于生成能够代表核心且大众的功能的标签。综上所述，本文方法能够在实际情景中为没有类别标签的包完成有效地多标签分类。

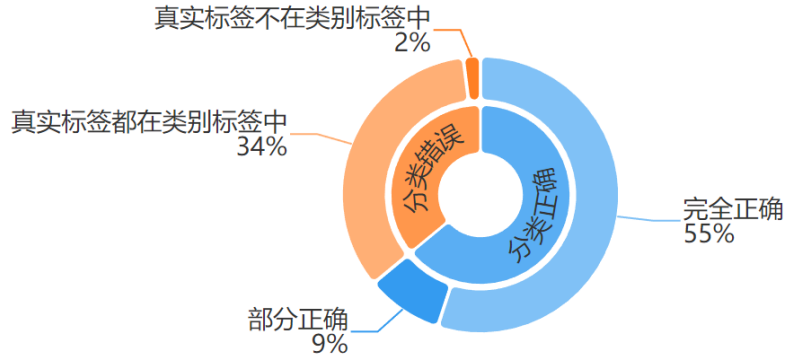


图 5.1 无标签包分类情况

5.4 类别标签间差异（RQ3）

RQ2 实验是在整体上验证方法的有效性，那么本小节将针对每一个类别标签的分类效果进行实验，探究类别标签间的差异。具体来说，在 RQ2 实验结果的基础上，以类别标签为单位，对每一个类别标签的分类结果独立地进行统计分析。采用的评价指标如下：

$$Precision_c = \frac{TP_c}{TP_c + FP_c} \dots\dots\dots (5.2)$$

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \dots\dots\dots (5.3)$$

$$F1 - measure = \frac{2 \times Precision_c \times Recall_c}{Precision_c + Recall_c} \dots\dots\dots (5.4)$$

值得注意的是，本小节所做的实验仍是在多标签分类的环境下，不是针对单一类别的二分类实验。

实验结果如图 5.2 所示，观察得到，各类别标签的分类结果差异较大。其中分类结果较好的有 cache, css, time 和 log，他们的 F1 值分别为 0.796, 0.792, 0.774 和 0.768。分类结果较差的有 network, util 和 debug，他们的 F1 值分别为 0.293, 0.303 和 0.407。对比这些标签的关键词集，分类结果较好的标签的

关键词集中，各个关键词与类别标签所描述的内容相关性较高，例如 `time` 标签的关键词集为 `time:{"date": 0.28, "time": 0.161, "format": 0.076, "picker": 0.048, "calendar": 0.043, "day": 0.036, "datetime": 0.029, "moment": 0.028, "timezone": 0.024.....}`，大多数关键词都与时间相关。对比之下，`debug` 标签的关键词集为 `debug:{"debug": 0.347, "console": 0.08, "browser": 0.041, "spare": 0.041, "development": 0.031, "variable": 0.031, "debugger": 0.031, "stringify": 0.03, "length": 0.028`，无法直观地从词义上将关键词与标签 `debug` 联系起来。上述结果表明类别标签关键词集的质量直接影响了该标签的分类结果。

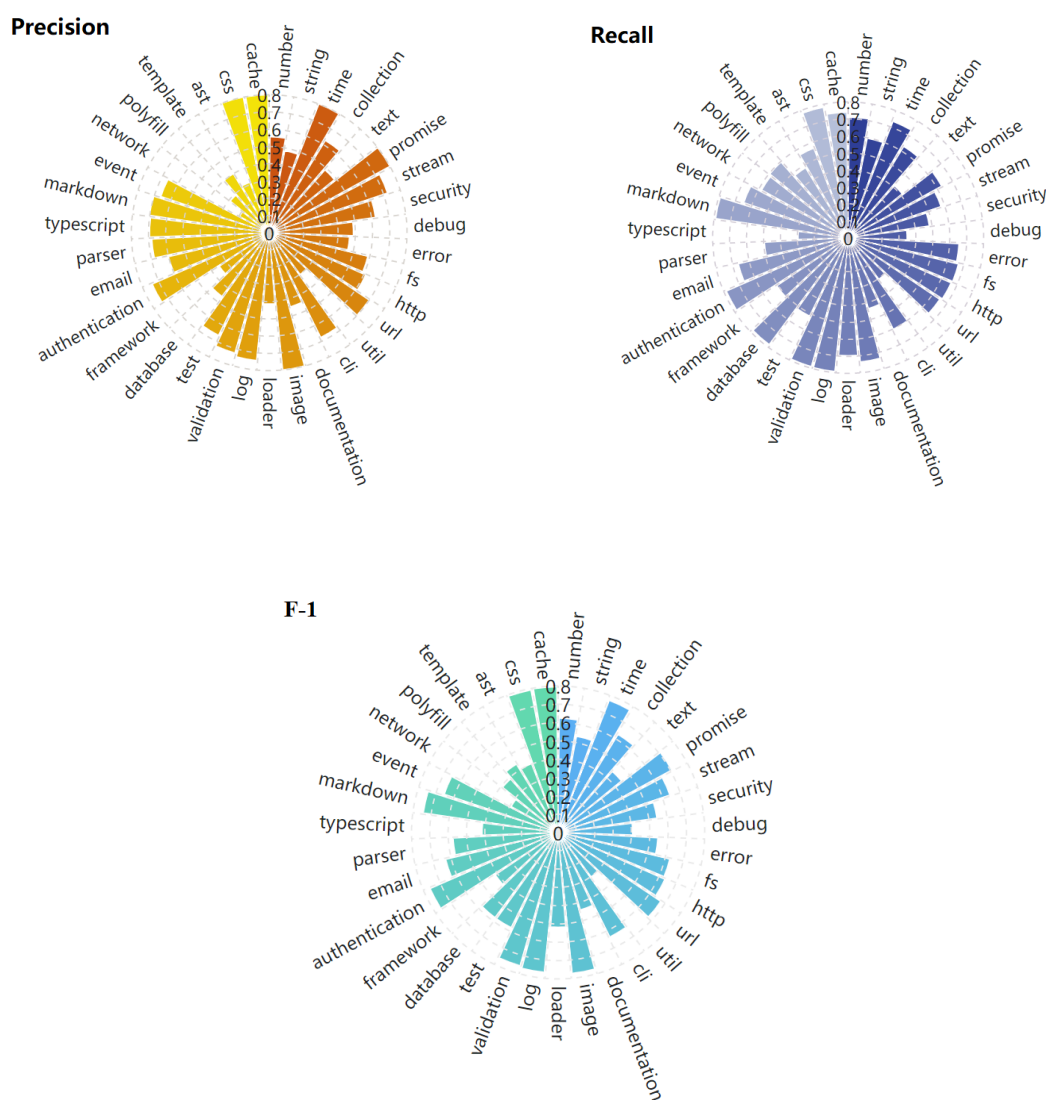


图 5.2 各类别标签下的分类情况

类别标签关键词质量不佳存在如下两个原因：一方面是有些类别标签本身的同义词，近义词和相关词就较少，而且生成关键词集的模型 `L-LDA` 为统计模型，

导致最终构成关键词的单词与类别标签的语义相关程度低。另一方面，训练文本的数量不足也是潜在原因。`debug` 标签所标记的训练数据量为 211 个，还不足 `time` 标签所标记的训练数据量的二分之一。

此外，本文对分类结果表现较差的几个类别进一步进行分析，发现 `network`，`polyfill` 和 `util` 的 $Precision_c$ 值很低，分别为 0.196，0.291 和 0.302；而 `typescript` 和 `debug` 类别的 $Recall_c$ 值很低，分别为 0.296 和 0.348。对于前一种情况，各类别下的 FP_c 值较高，即预测结果为 c ，实际情况不为 c 。以 `network` 类别为例，真集中标有 `network` 类别标签的共有 36 个包，而所有预测结果中被预测为 `network` 的包数量为 107 个，其中预测正确的 TP_c 为 21，而 FP_c 则达到了 86。观察预测错误的样本，发现他们的 `Readme` 中确实存在很多与 `network` 关键词语义相似的词语，导致计算词移距离结果小，`network` 类别标签排名靠前。实际上，在所有 86 个 FP_c 情况的包中，只有 27 个包被预测为单标签，而 59 个包都是多标签，意味着在 FP_c 的样本中 `network` 大多数是以多标签的形式被预测错误的。

$Recall_c$ 值低的类别中， FN_c 值较高，即真实情况包的类别标签为 c ，但预测结果中不包含 c 。以 `typescript` 类别标签为例，真集中被标记为 `typescript` 的包有 645 个，而预测结果中被标记为 `typescript` 的包只有 275 个，其中预测正确的数量为 191 个。所以有 370 个包没有被本文方法预测出来，原因在于 `typescript` 类别描述的是程序语言，在 `Readme` 文档中与之相关的描述词语较少，而在代码中更能直观地体现。

综上所述，本文方法在不同类别标签上的表现差别较大，这其中有类别标签自身特点的原因，也体现了本文方法的优势与不足。结合实验结果，在保持优势的前提下，如何弥补不足，提升方法对于各个类别标签的准确率将是接下来重要的研究方向。

5.5 本章小结

本章介绍了本文方法的实验过程以及实验结果分析。本章对方法的参数设定，方法的有效性和方法中各类别标签的分类差异进行实验，实验结果表明本文方法可以有效地给予 `npm` 包描述功能类别的类别标签。

第6章 总结与展望

6.1 工作总结

npm 社区作为面向 JavaScript 编程语言的开源软件社区，拥有数量超百万的 npm 包以供开发者共享和重用代码。随着开源软件的蓬勃发展，npm 包凭借丰富多样的功能吸引了越来越多的开发者进行使用，同时 npm 社区优秀的交流环境也促进了更多的开发者进行创作，导致 npm 包的数量呈现爆发式增长。海量的软件成为丰富资源的同时，也为 npm 社区带来了软件管理与定位的难题。

解决上述问题的有效途径是对软件资源进行分类，而 npm 社区通过标签方式来实现分类所达到的组织和检索资源的效果。标签用简短的描述信息来区分 npm 包的功能，将相关的包关联起来，对海量的包进行了划分。在检索过程中，使用标签不仅缩小了检索的范围，而且展示了软件的特征，便于搜索者快速掌握信息。然而，大量的包缺失标签，标签缺乏统一规范使得 npm 社区的标签机制无法完全起到软件分类的作用，也不足以实现软件管理与检索的目标。因此，本文借助 npm 社区已有的标签机制，对 npm 包分类问题进行研究，主要包含以下几个方面：

(1) 为充分发挥标签在 npm 包功能方面所起到的分类作用，同时消除原有标签机制中缺乏一致性的问题，本文从包功能的角度对 npm 社区中的所有标签进行分析，生成了统一的描述 npm 包功能的 35 个类别标签。由于描述同一功能的标签共同出现的概率远高于其他标签，则标签间的关联关系反映了标签作为类别的属性。因此，本文使用关联关系挖掘算法挖掘标签间的关联关系，构建标签关联关系图，并利用社区检测算法对标签关联关系图中的标签聚类成标签关联社区，使得一个标签关联社区中的所有标签都描述同一功能。为挑选出代表社区的标签，本文制定了一套类别标签识别机制，通过人工识别和标签在社区中的影响力，确定了最终的能够作为功能类别的类别标签。

(2) 为实现海量 npm 包的自动化分类，本文提出了一种面向 Readme 文档的多标签文本分类方法。Npm 包的 Readme 文档中包含了丰富的描述信息，本文通过对 Readme 文档的挖掘分析，巧妙地利用关键词实现自动化分类。首先，将所有已被类别标签标记的包的 Readme 文档作为训练文本，利用 L-LDA 主题模型为每个类别标签构建关键词集。其次，对于待分类的包，将其 Readme

文档转换为权重词集。最后，利用词移距离算法将权重词集与每个类别标签关键词集进行相似度的计算，取相似度高的标签作为分类结果。

(3) 本文通过实验验证了提出方法的准确性和有效性。由于本文方法中涉及多个参数，因此首先通过参数设定的实验确定了使方法达到最佳准确率的参数组合。为验证方法准确性，本文将四个多标签分类方法作为基线进行对比实验，实验结果表明本文方法的准确性均大幅高于四个基线方法。其次，为验证方法有效性，本文对实际没有类别标签的 npm 包做了实验验证，结果表明本文方法可以有效地完成 npm 包的多标签分类工作。此外，本文还探究了各类别标签的具体分类情况，从结果的差异上进行归纳总结，为后续研究打下基础。

6.2 未来展望

本文在研究过程中仍有不足之处，未来将针对以下两个方面做进一步研究：

(1) 不同类别标签在分类结果上的表现差异较大，个别标签的分类准确率并不理想，这其中虽然有标签自身特点的原因，但也反映了本文方法的不足。因此，下一步计划针对这些表现不好的类别标签进行深入研究，弥补方法的不足，提升相应标签的分类准确率。

(2) 本文通过分析 npm 包的 Readme 文档有效地实现了多标签分类的目标，并达到了不错的准确率。虽然基于文本分析的方法出色地完成了任务，但是 npm 包还有更多相关的数据并没有被本文方法所利用，例如包的源代码，包之间的依赖关系等。如果能在方法中对这些相关的数据做挖掘分析，分类的精度将会进一步提升。因此，下一步的计划中将对 npm 包中除了 Readme 文档外的数据进行探究，挖掘它们与类别标签的联系，帮助方法在分类性能上做进一步提升。

参考文献

- [1] Wirfs-Brock A, Eich B. JavaScript: the first 20 years[J]. Proceedings of the ACM on Programming Languages, 2020, 4 (HOPL): 1-189.
- [2] Matveyeva S J. A role for classification: the organization of resources on the Internet[J], 2002.
- [3] Liu J, Zhou P, Yang Z, et al. FastTagRec: fast tag recommendation for software information sites[J]. Automated Software Engineering, 2018, 25 (4): 675-701.
- [4] Treude C, Storey M-A. How tagging helps bridge the gap between social and technical aspects in software development[C]. 2009 IEEE 31st International Conference on Software Engineering, 2009: 12-22.
- [5] Wang S, Lo D, Vasilescu B, et al. EnTagRec++: An enhanced tag recommendation system for software information sites[J]. Empirical Software Engineering, 2018, 23 (2): 800-832.
- [6] Chatzidimitriou K, Papamichail M, Diamantopoulos T, et al. Npm-miner: An infrastructure for measuring the quality of the npm registry[C]. 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), 2018: 42-45.
- [7] Zimmermann M, Staicu C-A, Tenny C, et al. Small world with high risks: A study of security threats in the npm ecosystem[C]. 28th USENIX Security Symposium (USENIX Security 19), 2019: 995-1010.
- [8] Madsen M, Tip F, Lhoták O. Static analysis of event-driven Node.js JavaScript applications[J]. ACM SIGPLAN Notices, 2015, 50 (10): 505-519.
- [9] Kikas R, Gousios G, Dumas M, et al. Structure and evolution of package dependency networks[C]. 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017: 102-112.
- [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, 2017: 314-319.
- [11] Di Sipio C, Rubei R, Di Ruscio D, et al.: A Multinomial Naïve Bayesian (MNB) Network to Automatically Recommend Topics for GitHub Repositories[J]. Proceedings of the Evaluation and Assessment in Software Engineering, 2020.
- [12] Zhang Y, Xu F F, Li S, et al. Higitclass: Keyword-driven hierarchical classification of github repositories[C]. 2019 IEEE International Conference on Data Mining (ICDM), 2019: 876-885.
- [13] Velázquez-Rodríguez C, De Roover C. MUTAMA: An Automated Multi-label Tagging Approach for Software Libraries on Maven[C]. 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2020: 254-258.

- [14] Izadi M, Heydarnoori A, Gousios G. Topic recommendation for software repositories using multi-label classification algorithms[J]. *Empirical Software Engineering*, 2021, 26 (5): 1-33.
- [15] Cai X, Zhu J, Shen B, et al. Greta: Graph-based tag assignment for github repositories[C]. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016: 63-72.
- [16] Zhou J, Chen W, Wu G, et al. SemiTagRec: a semi-supervised learning based tag recommendation approach for Docker repositories[C]. *International Conference on Software and Systems Reuse*, 2019: 132-148.
- [17] Yin K, Chen W, Zhou J, et al. STAR: a specialized tagging approach for Docker repositories[C]. *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018: 426-435.
- [18] Wang T, Wang H, Yin G, et al. Tag recommendation for open source software[J]. *Frontiers of Computer Science*, 2014, 8 (1): 69-82.
- [19] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. *Journal of machine Learning research*, 2003, 3 (Jan): 993-1022.
- [20] Ramage D, Hall D, Nallapati R, et al. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora[C]. *Proceedings of the 2009 conference on empirical methods in natural language processing*, 2009: 248-256.
- [21] Moon T K. The expectation-maximization algorithm[J]. *IEEE Signal processing magazine*, 1996, 13 (6): 47-60.
- [22] Robert C P, Casella G, Casella G. Monte Carlo statistical methods[M]. 2. Springer, 1999.
- [23] Jones K S. A statistical interpretation of term specificity and its application in retrieval[J]. *Journal of documentation*, 1972.
- [24] Robertson S E, Walker S, Jones S, et al. Okapi at TREC-3[J]. *Nist Special Publication Sp*, 1995, 109: 109.
- [25] Charikar M S. Similarity estimation techniques from rounding algorithms[C]. *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, 2002: 380-388.
- [26] Landauer T K, Foltz P W, Laham D. An introduction to latent semantic analysis[J]. *Discourse processes*, 1998, 25 (2-3): 259-284.
- [27] Mikolov T, Chen K, Corrado G S, et al. Efficient Estimation of Word Representations in Vector Space[C]. *ICLR*, 2013.
- [28] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation[C]. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014: 1532-1543.
- [29] Kusner M, Sun Y, Kolkin N, et al. From word embeddings to document distances[C]. *International conference on machine learning*, 2015: 957-966.
- [30] Zhang M-L, Zhou Z-H. A review on multi-label learning algorithms[J]. *IEEE transactions on knowledge and data engineering*, 2013, 26 (8): 1819-1837.

-
- [31]Zhang M-L, Li Y-K, Liu X-Y, et al.Binary relevance for multi-label learning: an overview[J].Frontiers of Computer Science,2018, 12 (2): 191-202.
- [32]Read J, Pfahringer B, Holmes G, et al.Classifier chains for multi-label classification[J].Machine learning,2011, 85 (3): 333-359.
- [33]Elisseeff A, Weston J.A kernel method for multi-labelled classification[J].Advances in neural information processing systems,2001, 14.
- [34]Zhang M-L, Zhou Z-H.ML-KNN: A lazy learning approach to multi-label learning[J].Pattern recognition,2007, 40 (7): 2038-2048.
- [35]Dembczyński K, Waegeman W, Cheng W, et al. Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss[C].Joint European conference on machine learning and knowledge discovery in databases,2010: 280-295.
- [36]Schapire R E, Singer Y.BoosTexter: A boosting-based system for text categorization[J].Machine learning,2000, 39 (2): 135-168.
- [37]Agrawal R, Srikant R. Fast algorithms for mining association rules[C].Proc. 20th int. conf. very large data bases, VLDB,1994: 487-499.
- [38]Blondel V D, Guillaume J-L, Lambiotte R, et al.Fast unfolding of communities in large networks[J].Journal of statistical mechanics: theory and experiment,2008, 2008 (10): P10008.
- [39]Newman M E.Modularity and community structure in networks[J].Proceedings of the national academy of sciences,2006, 103 (23): 8577-8582 .
- [40]Prana G A A, Treude C, Thung F, et al.Categorizing the content of github readme files[J].Empirical Software Engineering,2019, 24 (3): 1296-1327.

作者简介及在学期间取得的学术成果

作者简介：

王禹，男，1996 年 6 月出生于辽宁省阜新市，2019 年毕业于吉林大学物联网工程专业，2019 年至 2022 年于吉林大学计算机科学与技术学院计算机软件与理论专业攻读硕士学位，研究方向为软件工程。研究生期间发表 CCF-C 类会议论文一篇。

发表论文：

1.Wang, Yu, Huaxiao Liu, Shanquan Gao and Shujia Li. “Categorizing npm Packages by Analyzing the Text Information in Software Repositories.” 2021 28th Asia-Pacific Software Engineering Conference (APSEC) (2021): 53-60.

致谢

时光飞逝，转眼间已经走到了读研的第三个年头。在毕业前夕，回顾我的研究生生涯，有太多难忘的经历，而这些珍贵的经历中最值得铭记的是那些给予我帮助和关怀的人们，在此我表示由衷地感谢。

首先感谢的人是我的导师，刘华琥老师。在读研期间，刘华琥老师无论是在学术上还是在生活中都给予了我莫大的帮助。想起初入实验室时自己对科研一无所知，是他耐心地领我入门，帮我成长。那时我们会频繁地交流新想法，探究新问题，想解决思路，共同面对挫折。就这样一步一步地帮助我在科研的道路上前进着，直到收获了我人生中的第一篇学术论文。在生活中，他的生活智慧总是会给我一些启发，无论是面对论文被拒的挫折时，还是在选择工作的犹豫不决时，他都能及时地给我鼓励和建议。老师，感谢您对我的指导和帮助，师恩难忘。

我还要感谢刘宇舟师兄和郜山权师兄。他们在我的科研道路上给予了非常多的建议，让我少走了很多弯路。他们成功的科研经历也时刻鼓舞着我，让我一路坚持了下来。山权师兄在我修改论文时帮了很大的忙，现在还能想起在他的工位上帮我改句子的身影。师兄们，感谢你们对师弟的关怀和帮助，我很荣幸能遇见你们。

最后，感谢我的家人们。感谢我的妈妈对我生活上的支持，谢谢她在我背后无私的奉献。

感谢我的女朋友在生活中的陪伴，在我失意的时候鼓励着我，和我分享着我的喜悦。我爱你们。

感谢所有帮助过我的人！