2024 届研究生硕士学位论文

分类号：_____ 学校代码：10269

密　级：_____ 学　　号：51215903014

華東師範大學

East China Normal University

硕 士 学 位 论 文

MASTER'S DISSERTATION

论文题目：基于协作网络与仓库文本特征的开源软件价值评估模型研究

院　　系:	**数据科学与工程学院**
专 业 名 称:	**数据科学与大数据技术**
研 究 方 向:	**开源软件供应链**
指 导 教 师:	**王伟 教授**
学位申请人:	**吴双**

2024 年 2 月 15 日

DISSERTATION for MASTER Degree in 2024 School Code: 10269

Student Number: 51215903014

East China Normal University

East China Normal University

MASTER'S DISSERTATION

Title: Research on Open Source Software Value Assessment Model Based on Network Features and Repositories' Content Features

Department:	School of Data Science and Engineering
Major:	Data Science and Big Data Technology
Research Area:	Open Source Supply Chain
Supervisor:	Professor Wei Wang
Candidate:	Shuang Wu

February, 2024

华东师范大学学位论文原创性声明

郑重声明：本人呈交的学位论文《XXXX（你的论文题目）》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名:_____ 日期: 年 月 日

华东师范大学学位论文著作权使用声明

《XXXX（你的论文题目）》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的研究成果归华东师范大学所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和相关机构如国家图书馆、中信所和“知网”送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

（ ）1. 经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文*，于 年 月 日解密，解密后适用上述授权。

（ ）2. 不保密，适用上述授权。

导师签名:_____ 本人签名:_____

年 月 日

* “涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

硕士学位论文答辩委员会成员名单

姓 名	职 称	单 位	备 注
田春岐	副教授	同济大学	主席
胡文心	教授级高工	华东师范大学	
董启文	研究员	华东师范大学	
陈优广	高级工程师	华东师范大学	
蔡鹏	研究员	华东师范大学	

摘　要

随着开源软件在全球技术生态系统中的地位日益提升，理解和评估这些软件及其背后的商业开源软件公司对于推动技术创新和行业发展起着至关重要的作用。开源软件通过技术共享与快速迭代，不仅促进了技术共享和快速迭代，还为企业显著地优化了成本。然而，对开源软件及商业开源软件公司价值的准确评估仍然是一个复杂且挑战性的任务。本研究采用了金融学中的资产定价对比法，通过深入分析相似项目或公司的价值指标，近似拟合目标项目及公司的价值。

资产对比法侧重于通过比较相似资产的市场表现来进行价值评估，这种方法在开源软件价值评估中的应用意味着需要识别出能够代表软件项目价值的关键特征。

开源项目和商业开源软件公司的价值的多种影响因素在一定程度上可以被抽象为开发维护活跃度、当前质量及潜在影响力，这其中开发维护活跃度和潜在影响力可以通过开源社区的协作网络特征代表，而当前质量可以通过开源软件的仓库文本特征特征代表。本研究旨在通过综合考量这些因素，开发一个创新的开源软件价值评估模型。

基于以上背景，本文的主要工作包括：

1. **构建支持开源软件商业价值评估任务的基准数据集** 对开源软件商业价值的预测需要历史上开源软件被融资的信息作为对比参考。本研究通过综合运用 OCR 技术和专业商业信息采集手段，首次成功地将融资信息数据量从 748 条扩展至 1507 条，并进行了严格的手动校验。此外，利用 GitHub 官方 REST API 和行为数据集，本研究从 2015 年至今采集了所有公开活动数据，并与商业开源软件公司的核心组织链接进行整合。这一过程不仅涉及了协作网络特征数据的收集，还包括了通过自动化技术获取的商业开源软件公司项目的 README 文档。本研究构建的数据集涵盖了商业开源公司的行为、影响力及介绍文档，形成了一个包含 9 个属性的基准数据集，为开源软件商业价值的准确评估提供了全面而深入的数据支撑。
2. **提出结合开源软件的协作网络和仓库文本的核心特征模型** 本研究提出了一个创新性的模型，该模型不仅结合了开源软件的协作网络特征与仓库文本特征，还引入了先进的统计方法和数据处理技术以提高评估的准确性和效率。通过对 OpenRank 的定制调整为 Dynamic OpenRank，本模型为不同类型的开源项目设定了差异化的阻尼因子，从而精准地反映了各项目在协作网络中的

动态影响力。此外，利用 BERT 技术对仓库 README 文档进行深度文本分析，模型能够综合利用文本数据中的关键信息，增强了对项目内容质量和社区活跃度的理解。同时，为了处理和分析构建的综合数据集中的高维特征，本研究采用了主成分分析 (PCA) 和核主成分分析 (KPCA) 降维技术，识别和提取数据集中的主要变量和模式，优化了数据集的结构和可管理性，将数据经过核 PCA 降维后得到更具有结构特征的结果，有效降低了数据的复杂性，使模型能够更准确地捕捉到对商业价值评估至关重要的特征，进一步提升了模型在开源软件商业价值预测中的性能和可靠性。本研究实现了轮廓系数从 0.2737 提升至 0.3895，戴维斯-布尔丁指数从 1.1360 降低至 0.8080，该数据明确表明了聚类质量的显著提升。这一结果突出了核 PCA 降维后的数据比原始数据更适合作为模型的输入值。

3. **提出了一种基于 DCNN-ETNN 的开源软件价值评估方法** 基于上述综合性数据集及核心特征模型，本研究提出了一种创新的开源软件价值评估方法，核心采用了基于深度卷积神经网络（DCNN）和增强 Transformer 神经网络（ETNN）的模型。此方法利用了先前构建的包含协作网络特征和仓库文本特征的综合性数据集，并通过应用统计方法和核主成分分析降维技术，优化了特征的选择和处理。在模型设计方面，本研究不仅比较了多个价值评估模型，包括线性回归、随机森林、XGBoost 及基础神经网络，还特别开发了包含自建 Transformer 层的复杂多层神经网络及其优化版本。经过严格的比较和评估，最终确定了一种结合 DCNN 和增强 Transformer 层的深度神经网络作为最优模型。该模型在处理复杂的综合性数据集时展现了卓越的性能，为开源软件价值评估提供了一种高效、精确的新方法。实验结果表明，相比基线模型和 XGBoost 显示出显著的性能提升，DCNN-ETNN 模型的决定系数达到了 32.42%，

本研究借鉴了金融学中的资产定价方法——对比法，通过深入分析相似项目或公司的价值指标，定义了一套准则来识别和选择比较对象，并采用了数据处理和自然语言处理技术，以调整和比较不同项目之间的关键价值差异。这种方法的应用，允许相关从业人员从一个全新的角度出发，结合传统的量化指标和项目文档的质量分析，为开源软件的价值评估提供了一个更为精确和全面的框架。同时本研究还通过实证分析验证了对比法在开源软件价值评估中的有效性和实用性。

关键词： 开源软件，Open Source Software；资产定价，Asset Pricing；机器学习，Machine Learning

ABSTRACT

As the role of open-source software in the global technology ecosystem continues to rise, understanding and assessing the value of these software and the commercial open-source companies behind them play a crucial role in driving technological innovation and industry development. Open-source software, through technology sharing and rapid iteration, not only promotes technological exchange and swift development but also significantly optimizes costs for businesses. However, accurately assessing the value of open-source software and commercial open-source companies remains a complex and challenging task. This study adopts the asset pricing comparison method from finance, deeply analyzing the value indicators of similar projects or companies to approximate and fit the value of the target projects and companies.

The asset comparison method focuses on evaluating value by comparing the market performance of similar assets, implying that key features representing the value of software projects need to be identified for application in open-source software value assessment.

The value of open-source projects and commercial open-source software companies can be abstractly represented by several factors, such as development and maintenance activity, current quality, and potential influence. Among these, development and maintenance activity and potential influence can be represented by the collaborative network features of the open-source community, while the current quality can be represented by the textual features of the software's repositories. This study aims to develop an innovative open-source software value assessment model by comprehensively considering these factors.

Based on the above background, the main work of this thesis includes:

1. **Building a benchmark dataset for assessing the commercial value of open-source software** predicting the commercial value of open-source software requires information on historical financing of open-source software as a reference

for comparison. This study, for the first time successfully expanded the financing information data volume from 748 to 1507 through a combination of OCR technology and professional business information collection methods, followed by strict manual verification. Moreover, using GitHub's official REST API and behavior datasets, this study has collected all public activity data from 2015 to the present and integrated it with the core organization links of commercial open-source software companies. This process involved not only the collection of collaborative network feature data but also the acquisition of README documents of projects from commercial open-source software companies through automation technology. The dataset constructed by this study covers the behavior, influence, and introduction documents of commercial open-source companies, forming a complex dataset with 9 attributes, providing comprehensive and in-depth data support for the accurate assessment of the commercial value of open-source software.

2. **Proposing a core feature model combining collaborative network and repository text of open-source software** this study proposes an innovative model that not only combines the collaborative network features and repository text features of open-source software but also introduces advanced statistical methods and data processing technologies to improve the accuracy and efficiency of assessments. By custom-adjusting OpenRank to Dynamic OpenRank, this model sets differentiated damping factors for different types of open-source projects, precisely reflecting their dynamic influence within the collaborative network. Moreover, using BERT technology for deep text analysis of repository README documents, the model can comprehensively utilize key information from text data, enhancing the understanding of project content quality and community activity. Additionally, to process and analyze the high-dimensional features in the constructed comprehensive dataset, this study employed Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA) dimensionality reduction techniques, optimizing the dataset's structure and manageability. These

statistical methods not only help identify and extract the main variables and patterns in the dataset but also effectively reduce data complexity, allowing the model to more accurately capture features crucial for commercial value assessment, further enhancing the model's performance and reliability in predicting the commercial value of open-source software.

3. **Proposing an open-source software value assessment method based on DCNN-ETNN** based on the comprehensive dataset mentioned above, this study presents an innovative open-source software value assessment method, centrally utilizing a model based on Deep Convolutional Neural Networks (DCNN) and Enhanced Transformer Neural Networks (ETNN). This method leverages the previously constructed dataset containing collaborative network features and repository text features and optimizes feature selection and processing through statistical methods and kernel principal component analysis dimensionality reduction techniques. In terms of model design, this study not only compared multiple value assessment models, including linear regression, random forests, XGBoost, and basic neural networks but also specially developed complex multi-layer neural networks with custom Transformer layers and their optimized versions. After rigorous comparison and evaluation, a deep neural network combining DCNN and enhanced Transformer layers was determined as the optimal model. This model demonstrated outstanding performance in processing complex comprehensive datasets, providing an efficient and accurate new method for open-source software value assessment.

This study draws inspiration from the asset pricing comparison method in finance. By deeply analyzing the value indicators of similar projects or companies, this study defines a set of criteria for identifying and selecting comparison objects and employs data processing and natural language processing technologies to adjust and compare key value differences between different projects. The application of this method allows us to approach open-source software value assessment from a new angle, combining

traditional quantitative indicators with quality analysis of project documents, providing a more accurate and comprehensive framework for open-source software value assessment.

Keywords: *Open Source Software; Asset Pricing; Machine Learning*

目录

插图

表格

第一章　引言

1.1　研究背景与意义

随着信息技术时代的深入发展，开源软件（Open Source Software, OSS）在全球软件生态系统中的地位变得日益重要。OSS 的兴起不仅促进了技术知识的共享和创新速度的加快，还为企业和个人提供了低成本、高效率的技术解决方案 [1][2]。然而，随着开源项目数量的激增，如何准确评估开源软件及其背后商业开源公司的价值，以指导投资、开发和政策制定，成为了一个复杂而紧迫的问题。传统的价值评估方法往往无法完全适用于开源生态的特点，这要求研究者和实践者探索更加创新和全面的评估模型。

1.1.1　研究背景

在当今数字化时代，开源软件通过其易于访问和修改的特性，不仅降低了软件开发的门槛，还促进了跨界创新，已经成为推动技术创新和促进全球技术共享的重要力量。OSS 的成功不仅体现在其技术创新和广泛应用上，更在于其背后的开源社区文化和协作机制，这为软件开发带来了根本性的变革 [3]。尤其是在企业级市场，商业开源公司通过提供支持和服务，促进了 OSS 的进一步发展和商业化应用 [4]。

然而，尽管 OSS 的影响力不断扩大，对其价值的准确评估却面临着巨大挑战。传统的价值评估模型往往关注于财务表现和市场表现的定量分析，很少有工作从开源项目的内在属性如社区活跃度、贡献者行为、以及文档质量等方面进行综合评估 [5][6]。

随着开源项目数量的激增和开源社区信息的数量级过大，传统的简单的模型已无法满足需求。此外，随着自然语言处理（NLP）等文本处理技术大模型在软件工程领域的应用日益广泛 [7]，如何有效利用这些技术来提高 OSS 价值评估的准确性和效率，成为了研究的新方向。

此外，当前研究对于利用金融学中的对比法原理和先进的自然语言处理（NLP）技术在 OSS 价值评估中的应用也鲜有涉及。这些技术的结合，有潜力显著提高评估的准确性和深度，尤其是在处理和分析大规模文本数据时 [7]。

1.1.2　研究动机和目的

尽管开源软件（OSS）在全球技术生态系统中扮演着越来越重要的角色，现有的价值评估模型往往未能充分捕捉到决定其价值的关键因素。

在商业领域，公司价值的评估通常涵盖其市场影响力、运营活跃度以及公共信息的质量和可获取性。将这一逻辑应用于 OSS，本研究认为开源项目的价值同样可以通过其影响力、开发环境的活跃度以及 README 文档的质量来衡量。这一认识指向了开发一个全新的 OSS 价值评估模型的必要性，该模型能够综合这些关键因素，为开源社区提供更准确的价值判断。

本研究的动机在于填补现有评估模型在综合考量 OSS 项目特性方面的空白，特别是在量化项目的社区影响力、开发活跃度（更新频率和完善程度）以及 README 文档的易理解性和内容质量方面。本研究提出，通过精细化地分析这些维度，可以构建出一个能够全面反映开源软件价值的评估模型。这一模型不仅有助于揭示开源项目的内在价值，还可以为投资者、开发者和政策制定者提供更为科学的决策支持工具，进而促进 OSS 生态的健康发展和创新活动。

因此，本研究的主要目的是开发一个创新的 OSS 价值评估模型，该模型基于开源项目的社区影响力、开发活跃度以及 README 文档的质量进行综合评估。通过实证分析验证该模型的有效性，本研究旨在为 OSS 的评估提供一个新的理论框架和实用工具，从而帮助相关利益相关者更好地理解和利用开源软件的潜在价值。

1.2　研究现状

尽管开源软件（OSS）在全球软件生态系统中的影响日益增长，但对其价值的综合评估仍是一个复杂且未充分探索的领域。当前研究在评估 OSS 的价值时往往

侧重于单一维度，如社区活跃度、贡献者行为或项目的技术特性，忽视了这些因素之间的相互作用及其对 OSS 价值的综合影响。此外，虽然文本处理技术在提取项目文档和代码库中的关键信息方面显示出潜力，但其在评估 OSS 价值模型中的应用还处于起步阶段。本节综述了 OSS 社区影响力的量化研究、商业开源公司价值评估模型的发展，以及文本处理技术在软件工程领域的最新应用，旨在揭示这一跨学科领域的研究空白和未来机遇。

1.2.1 开源软件的社区影响力量化研究

OSS 项目的成功高度依赖于其背后社区的健康和活跃度。量化社区活跃度的研究表明，社区的参与度、贡献者多样性及其互动质量对项目的持续发展至关重要 [8][9]。然而，这些研究往往未能全面覆盖影响 OSS 项目成功的所有社会技术因素，特别是在将社区活跃度与项目的技术质量和市场表现联系起来方面还有待深入 [10]。同时，这些研究往往忽略了软件文档的质量和商业开源公司的市场表现对软件价值的影响。[11]

1.2.2 商业开源公司价值评估模型

尽管商业开源公司在软件生态中扮演着越来越重要的角色，但目前对其价值的评估模型还不够成熟。根据近期的投资行为进行分析，现有的投资研究更加集中在股票二级市场以及非开源的商业软件公司，这些研究主要依赖于财务数据和市场表现 [12][13][14]，缺乏对开源特有属性的深入考量但它们往往忽略了 OSS 项目的社区动态和开源特有的价值创造机制。

非财务指标的研究也主要是探索包括社区规模、开源许可证的选择以及开源项目使用的技术版本等最为表象的特征，通过这种信息很难为其提供更为全面的商业开源公司价值评估 [15]。

1.2.3 自然语言处理在开源软件分析中的应用

近年来，随着计算技术的发展，深度神经网络，尤其是自然语言处理（NLP）大模型技术，在开源软件分析中的应用已经从代码注释分析扩展到需求提取、文

档质量评估乃至社区讨论内容的情感分析 [16][17]，包括自动化文档生成、代码评论分析以及 bug 报告分类等方面 [16][18][19][20][21][22][23][24]。

NLP 技术能够帮助开发者和研究人员在海量的文本数据中提取有用信息，理解开源社区的动态和用户需求。此外，NLP 还可以用于自动化生成和优化软件文档，提高文档的可读性和有效性，从而降低新用户的学习成本并提高软件的可用性。

具体的，本研究有如下典型应用场景：

1. **自动化文档生成**随着软件项目的增长和复杂性提高，手动编写和维护文档变得越来越困难。NLP 技术可以自动从代码库、提交日志和 bug 报告中提取关键信息，用于生成或更新软件文档。例如，LeClair 等人提出的方法，通过从源代码中自动生成文档，显著提高了文档的生成效率和质量 [19]；使用 NLP 技术可以自动生成 API 文档，通过分析代码结构和注释内容，自动提取关键信息生成文档 [20]。

2. **代码评论分析**代码审查是开源软件开发中的一个重要环节，NLP 可以用于分析代码评论，自动识别出哪些评论是关于代码逻辑错误、性能问题或是可读性改进建议。Panichella 等人的研究展示了如何使用 NLP 技术来自动分类代码审查中的评论，从而帮助开发者更高效地处理这些评论，更有效地进行代码审查 [16][21]。

3. **Bug 报告分类**在开源项目中，用户和开发者经常会提交 bug 报告，但这些报告的质量和相关性差异很大。NLP 技术可以用于自动分类 bug 报告，区分出哪些是真正的 bug，哪些是功能请求或是无效报告。Lam 等人开展的研究，通过使用机器学习算法对 bug 报告进行自动分类，有效提升了 bug 处理的效率 [22]；同时也有研究可以对 bug 进行优先级排序，帮助开发者快速识别和解决最紧迫和重要的问题 [23]。

4. **社区交流内容的情感分析**开源社区的健康度和活跃度对项目的成功至关重要。NLP 技术可以用于分析社区交流内容的情感倾向，如 GitHub issue 讨论、邮

件列表或论坛帖子，帮助项目维护者理解社区成员的情绪和需求。[18]Novielli 等人的工作表明，通过情感分析可以揭示开源项目社区的情绪动态，为社区管理提供有价值的洞见 [24]。

这些技术为自动化提取和分析 OSS 项目中的文本信息提供了强大工具，为评估 OSS 价值的新维度打开了大门。然而，将这些技术综合应用于构建 OSS 价值评估模型，尤其是在考虑经济学和社会学理论的情况下，目前的研究成果还相对有限。现有的 NLP 技术应用最接近的研究还局限于基于 GitHub、StackOverflow 等代码技术问题交流平台构建大模型和问答系统 [25][21]。

1.2.4 国内外研究评价

尽管 OSS 价值评估的重要性在国际学术界得到了广泛认可，提出了多种评估模型和方法，但这些研究大多聚焦于评估特定的 OSS 项目或考虑有限的评估指标，例如项目的技术质量或开发用户数 [26]。这些研究虽然在各自的领域内取得了进展，但在综合评估 OSS 价值的全面性和深度上仍存在限制。此外，关于结合金融学原理和文本处理技术进行 OSS 价值评估的研究在国际上也相对缺乏，尤其是在将这些方法应用于大规模 OSS 项目数据分析方面 [27]。

国内在 OSS 价值评估方面的研究则更为有限，尽管近年来对 OSS 的兴趣和应用日益增加，但关于如何评估 OSS 及其背后商业模式的价值的系统性研究仍然不足 [28]。这一研究空白强调了开发一个全面、创新且适用于多种 OSS 项目的价值评估模型的迫切需求，以及将国际上的先进研究成果和方法引入国内，推动 OSS 价值评估研究的深入发展。

1.3 本文主要工作和贡献

本研究出发点认为，开源软件价值核心

本研究基于对开源软件社区价值特性的理解，创新性地提出了一种结合对比法和文本处理技术的开源软件价值评估模型。本研究的贡献主要包括：

1. 构建了一个综合性的商业开源公司数据集，包括被投资信息、GitHub 活动数据和 README 文档内容。
2. 开发了一种新的价值评估模型，该模型通过分析比较相似项目的社区影响力和文档质量，有效预测开源软件的价值。
3. 实证分析验证了模型的有效性，为开源软件投资决策提供了新的理论和实践工具。
4. 开发了一个面向用户的价值评估工具，提高了模型的可访问性和实用性。

通过这些工作，本研究不仅丰富了开源软件价值评估的理论基础，也为相关领域的研究和实践提供了有价值的参考。

1.4　本文组织结构

本文的组织结构如图1.1所示。

第一章介绍了研究的背景，提出了一种结合对比法和文本处理技术的开源软件价值评估模型，旨在通过构建商业开源公司数据集、开发评估模型、进行实证分析验证模型有效性，为开源软件投资决策提供新的理论和实践工具。

第二章回顾了开源软件和商业开源公司的关键概念，强调了协作网络驱动的开发模式的重要性，并详细介绍了协作网络特征和仓库文本特征的评估方法，特别是引入了自然语言处理技术提高对开源项目文档的理解和分析能力。

第三章构建了一套支持开源软件商业价值评估任务的基准数据集。此外，本章详细讨论了这些数据的收集与预处理过程，包括采集框架的设计、先进的网络爬虫技术应用、数据验证和清洗流程，以及数据治理框架的建议。从而为后续的研究和分析提供了可靠的数据基础。

第四章介绍了一个结合开源软件的协作网络特征和仓库文本特征的模型，旨在更准确地预测社区活跃度和影响力。通过融合 Dynamic OpenRank 指标和 BERT

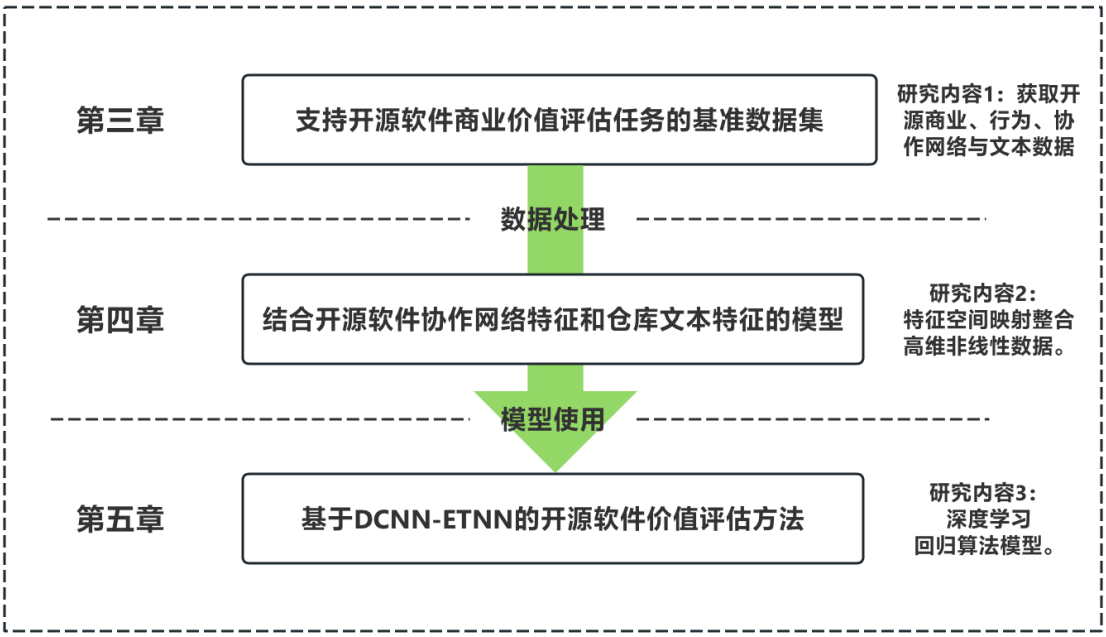

图 1.1 本文大纲结构

处理的文本特征，进行数据的标准化、降维等处理。最终，模型依据精心挑选的特征，有效反映了开源项目的社区活跃度和影响力。

第五章通过六个模型以及基线的对比实证分析验证了开源软件价值评估模型的有效性，特别指出优化的 DCNN-ETNN 模型在预测开源软件价值方面表现出众。分析基于综合数据集，展示了新模型相较于传统评估方法的优势，重点在于协作网络和仓库文本内容质量对软件价值的显著影响。该模型提供了一个实用工具，帮助投资者和决策者评估商业开源项目的潜在价值，其准确性和效率通过多个案例研究得到验证。尽管存在局限性，本章还讨论了未来研究方向，旨在提高模型的泛化能力和评估指标的完善性。

第二章　背景知识与理论基础

2.1　开源软件和商业开源公司

2.1.1　开源软件的定义与特点

开源软件（OSS）指的是其源代码可以被公众使用、查看、修改和分享的软件。与专有软件相比，开源软件强调的是开放协作和知识共享的理念 [29]。开源软件的主要特点包括透明性、可访问性和社区驱动的发展模式。这些特点不仅促进了技术的快速创新和迭代，还降低了软件开发和维护的成本 [14]。

2.1.2　商业开源公司的兴起与发展

商业开源公司的出现是对传统软件开发和销售模式的一种创新。随着开源软件在全球技术生态系统中的地位日益提升，越来越多的商业公司开始基于开源项目构建其商业模型。商业开源公司的商业模式有很多种，包括但不限于提供付费的支持与咨询服务、销售开源许可证、开发开源软件的专有扩展，以及基于开源的软件即服务（SaaS）模型 [15]。

以下是每项策略的具体例子：

1. **提供付费的支持和咨询服务**

    - **Red Hat[30]** 提供企业级 Linux 操作系统 Red Hat Enterprise Linux（RHEL）的支持和咨询服务，帮助企业实现高效的系统管理和优化。
    - **Canonical[31]** 通过其 Ubuntu Advantage 服务提供专业的技术支持和安全维护服务，确保企业用户的 Ubuntu 系统稳定运行。
    - **MongoDB Inc.[32]** 提供 MongoDB 数据库的专业支持和咨询服务，帮助企业优化数据库性能和架构设计。

2. **销售开源许可证**

   - **MySQL AB[33]（现为 Oracle 公司的一部分）**通过双重许可模式销售 MySQL 数据库，一方面提供开源版本，另一方面向需要专有许可的企业用户销售商业许可。
   - **Talend[34]** 提供数据集成和数据管理解决方案，通过开源社区版和企业版的双轨模式，后者包含额外的功能和支持服务，需要购买许可证。

3. **在开源软件上构建专有扩展**

   - **Elasticsearch B.V.[35]** 开发了 Elasticsearch 开源搜索和分析引擎，同时提供 X-Pack 插件，包含安全、监控和报告等企业级功能，作为专有扩展销售。
   - **GitLab Inc.[36]** 以开源项目 GitLab 为基础，提供 GitLab Enterprise Edition，这是一个包含高级功能（如高级 CI/CD、安全测试、代码质量报告等）的专有版本。

4. **基于开源的 SaaS 模型**

   - **GitHub Inc.[37]** 提供基于 Git 的代码托管服务，虽然 Git 是开源工具，但 GitHub 通过提供托管服务、私有仓库等付费功能来盈利。
   - **Databricks[38]** 以 Apache Spark 为核心，提供一个统一的分析平台，通过其云服务模型为数据科学家和工程师提供大数据处理和机器学习服务。
   - **Nextcloud GmbH[39]** 基于开源的 Nextcloud 项目提供私有云存储和协作平台的 SaaS 解决方案，支持文件共享、在线办公和团队协作等功能。

这些模式的共同点在于它们都利用了 OSS 的核心优势——低成本、高可靠性和强大的社区支持。这些优势不仅为开源项目提供了可持续发展的资金支持，也

推动了开源软件在企业级市场的广泛应用 [40]，同时为企业提供了可持续的盈利路径。

随着开源文化的普及和开源技术的成熟，越来越多的传统软件公司也开始采纳开源模式，通过开源组件加速自身产品的开发。这种趋势不仅促进了商业开源公司的发展，也推动了整个软件产业的创新和进步。

2.2　开源价值评估相关工作

在进行资产、项目或企业价值评估时，传统的价值评估模型主要包括对比法（Comparative Approach）、成本法（Cost Approach）和收益法（Income Approach）。这些方法在商业和财务分析中被广泛应用，各有其特定的适用场景和优缺点。

- **对比法**也称为市场法，主要通过比较已知价格的相似资产来估算目标资产的价值。在实践中，这要求有足够多的相似案例作为参考，以及一个活跃且透明的市场以获取这些信息。对比法在房地产评估和二手市场中尤为常用 [41][42]。
- **成本法**基于重置或重建目标资产所需成本进行价值评估。它考虑了获取相同的功能和效用所必须的全部成本，包括材料、劳动和间接费用等。成本法适用于独特资产的评估，如特殊用途的建筑物 [43][44]。
- **收益法**依据资产未来产生的净收益进行估值，通过贴现未来收益流到现值来确定资产的价值。这种方法适用于能够产生可预测现金流的资产，如投资性房产、企业等 [45][46]。

2.2.1　对比法在开源项目价值评估中的应用

在开源项目的价值评估中，成本法和收益法的适用性受到限制。成本法难以准确评估因开源项目带来的间接收益和社区价值。收益法则面临着无法预测开源

项目未来直接收益的挑战，尤其是对于那些主要依靠社区贡献和非直接商业模式的项目。

相较之下，对比法因其灵活性和适应性，在开源项目价值评估中显得更为合适。通过比较相似的开源项目或商业开源公司，在某些维度（如社区活跃度、项目影响力等）上的表现，可以对目标开源项目的价值进行估计。然而，这需要一个合理的评价框架和准确的比较维度，以确保评估的有效性和准确性。

2.2.2　现有模型的局限性与挑战

多次典型案例证明，传统的价值评估模型如果直接应用于开源软件投资是有极大风险的。其中，由于商业生态的不同，收益法和成本法尤其不具有实用性。

具体的，针对收益法，Linux 内核是一个很好的反应其局限性的案例。Linux 内核广泛应用于服务器、嵌入式系统和超级计算机等多种环境中，其价值远超过直接通过捐赠、支持服务或赞助所获得的收入。Wheeler 的研究通过估算 Linux 内核的代码行数和开发成本来证明，如果采用商业软件的开发模式，成本将高达数十亿美元 [47]。然而，直接收益法只考虑了可直接量化的经济收入，忽略了 Linux 作为开放源代码项目对技术创新、教育和社区发展的广泛影响。

Wheeler 估算 Linux 内核包含超过 1000 万行代码，按照商业软件平均开发成本计算，总成本高达数十亿美元，远超 Linux 基金会接收的直接捐赠和支持服务收入。

类似的，针对成本法，Apache 服务器的相关案例也很好的说明了其局限性。Apache 是世界上最流行的 Web 服务器之一，其开源特性使得许多企业和个人用户可以免费使用，从而节约了购买商业软件许可的成本。然而，Hahn 指出，成本节约法难以量化开源软件带来的额外价值，如提高的安全性、灵活的定制能力和快速的漏洞修复等 [48]。

根据 Netcraft 的市场调查，Apache 服务器长期占据 Web 服务器市场的主导地位。其高安全性和稳定性是企业选择 Apache 的重要原因之一，但这些因素在成本节约法的评估中往往被忽视。

同样的，针对对比法，MySQL 数据库的相关案例展现了其局限性。MySQL 作为广泛使用的开源关系数据库管理系统，其价值不仅在于功能和性能，还在于其庞大的用户基础和活跃的社区支持。Fitzgerald 指出，市场比较法在评估 MySQL 等开源软件时，难以找到完全相似的市场产品，因为开源软件的独特价值和生态系统贡献在市场交易中难以量化 [49]。

根据 DB-Engines 的统计，MySQL 在关系数据库市场中占有显著的份额。然而，市场比较法往往无法量化用户基础的广泛性和社区的活跃度对于 MySQL 价值的贡献。

最后，Mockus 的研究通过分析 Apache 和 Mozilla 项目，展示了这些开源项目如何通过社区贡献和协作促进技术创新和标准的发展 [50]。技术影响力评估法虽然考虑了这些非直接经济效益，但由于技术影响力难以直接量化，这种评估方法往往依赖于主观判断，缺乏客观的评估标准。

Apache 和 Mozilla 项目对 Web 服务器软件和 Web 浏览器领域的技术发展产生了深远影响，包括推动了多项 Web 技术标准的制定。然而，这些贡献的价值在技术影响力评估法中难以用具体数字量化。

2.3　开源软件社区和影响力特征

开源软件项目的成功在很大程度上依赖于其背后的社区。社区成员通过报告错误、提交代码、撰写文档和参与讨论等方式贡献力量，这不仅提高了软件的质量和安全性，还加速了新功能的开发 [51]。一个活跃的社区往往意味着项目能够持续吸引和保留贡献者，这是衡量开源软件影响力的重要指标 [52]。

开源软件的影响力还体现在其能够推动技术标准的制定、促进行业内的最佳实践分享以及对全球软件开发模式的长期影响 [53]。通过分析社区的活跃度、贡献者的多样性以及项目的接受度和使用范围，可以全面评估开源软件的影响力和价值 [9]。

开源社区的健康度和项目影响力的评估是一个多维度的过程，它涉及对社区活跃度、项目质量、用户满意度等多个方面的综合考量。成功的开源项目不仅需要技术上的创新，更需要强大的社区支持和有效的项目管理，以保持项目的活力和持续发展能力。

2.3.1　开源软件的社区性

开源软件的社区性是其成功的关键要素，它体现了一种集体合作和知识共享的文化。在开源社区中，参与者不仅包括代码贡献者，还涵盖了报告问题的用户、编写文档的志愿者以及提供反馈的其他利益相关者。社区的活跃程度、多样性和协作能力直接影响到开源项目的质量、可持续性和创新能力 [52]。有效的社区管理、透明的沟通机制和包容的参与文化是构建健康开源社区的关键因素 [54]。

2.3.2　Github 作为开源软件社区的典型性

GitHub 已经成为全球最大的开源软件开发平台，提供了强大的工具和服务支持开源项目的协作开发，代表了开源软件社区的典型性。通过提供代码托管、项目管理和协作工具，GitHub 极大地促进了开源项目的发展和开源文化的传播。它支持的 Fork 和 Pull Request 机制简化了代码贡献的流程，使得参与开源项目变得更加容易和高效。GitHub 上的 Star、Fork 和 Watch 等指标，成为了衡量项目受欢迎程度和社区活跃度的重要指标 [55]。

2.3.3　社区参与度和项目影响力的量化

社区参与度和项目影响力的量化对于评估开源软件的成功至关重要。参与度可以通过贡献者数量、贡献频率、Issue 和 Pull Request 的活跃度等指标来衡量。项目影响力则可以通过项目的 Fork 数量、Star 数量、以及软件的下载量和应用范围来评估。这些量化指标不仅反映了社区的活跃程度，也指示了项目在更广泛社区或行业中的认可度和影响力 [56]。

2.3.4 OpenRank 指标的原理与应用

OpenRank 是一个由华东师范大学数据科学与工程学院 X-lab 实验室开发的量化指标，旨在综合评价开源软件项目的影响力和社区活跃度。OpenRank 算法基于社交网络模型，旨在评估并激励阿里巴巴项目中的开源贡献。该算法通过引入 OpenRank 排行榜（Open-Leaderboard），运用游戏化元素促进透明沟通、赢得权威和社区优先于代码的原则。受 PageRank 启发，该算法根据开发者在合作网络中的中心性计算贡献值。开发者对该算法的评价正面，认为其能够突出有价值的贡献并促进健康竞争，尽管也提出了对分数复杂性和潜在操纵的担忧。[57]

OpenRank 算法是基于 PageRank 算法的一种变体，专为开源项目中的合作单元如问题（Issues）和拉取请求（Pull Requests），以及开发者这些节点定制的。这个算法在 GitHub 仓库中的应用场景包括开发者围绕问题和拉取请求进行合作，展现出各种合作行为。值得一提的是，问题和拉取请求的初始价值受到开发者对它们正文中点赞表情的影响。

与 PageRank 算法相似，OpenRank 利用算法来判定问题/拉取请求和开发者的中心性，作为贡献价值的指标。然而，与 PageRank 不同，OpenRank 算法在计算每个节点的中心性时，不仅考虑了合作网络的结构，还考虑了节点自身的内在价值。这个独特的特性意味着 OpenRank 不是一个标准的马尔可夫过程，因此它的收敛需要额外的证明。

OpenRank 算法在每次迭代中对每个节点 v_i 的值的计算公式如下：

$$v_i = (1 - a_i) \sum_{j=1}^{|V|} \frac{w_{ji}}{d_{oj}} v_j + a_i v_0 \tag{2.1}$$

其中，v_0 代表节点的初始值，a_i 表示节点依赖其初始值的程度，d_{oj} 是节点 j 的加权出度，w_{ji} 是从节点 j 到节点 i 的边的权重。通过将 $\frac{w_{ji}}{d_{oj}}$ 规范化为矩阵 S，并将 a_i 作为矩阵 A 的对角线值，根据收敛证明，所有节点的 OpenRank 值将收敛到以下向量：

$$v = (E - AS)^{-1}(E - A)v^{(0)} \tag{2.2}$$

在这个公式中：

- v_i 表示在迭代过程中节点 i 的 OpenRank 值。
- v_0 是节点的初始值，可以理解为在算法开始之前，每个节点被赋予的基础价值。
- a_i 是一个系数，表明节点值多少依赖于其初始值。
- d_{oj} 是节点 j 的加权出度，即从节点 j 出发到其他节点的边的权重总和。
- w_{ji} 是边的权重，表示从节点 j 到节点 i 的关系强度。
- S 和 A 是通过特定方法从网络的结构中得到的矩阵，分别表示权重规范化后的关系和节点对初始值依赖程度的描述。
- E 是单位矩阵，代表一个在矩阵运算中不改变其他矩阵值的因子。

最终，OpenRank 值 v 通过上述公式计算得到，反映了节点在网络中的重要性和影响力。这个过程考虑了节点的内在价值和它们在网络中的相互作用，为开源项目中的合作和贡献提供了一种量化的评估方法。

算法在每次迭代中更新每个节点的值，直到达到一个稳定状态。为了确保算法的有效性，OpenRank 特别考虑了节点的初始值，这包括基于上月数据继承的价值，以及基于开发者对问题和拉取请求的正面反馈进行的调整。

在网络中，所有节点的中心性有 15% 基于它们的初始价值，而 85% 由它们在网络中的互动和连接决定。这种设计反映了 PageRank 默认阻尼因子的概念，确保了算法的平衡和有效性。

此外，算法通过考虑节点间的价值转移来确保收敛，并且每种类型节点的价值转移比例总和为 1。开发者和问题/拉取请求之间的不同合作行为—如开启、评论、审查和关闭—根据它们代表的不同努力被赋予了不同的权重。这些权重的确定依赖于层次分析过程（AHP），一个在运筹学中常用的方法，以确保算法的公正和一致性。

总之，OpenRank 算法通过综合考虑开源项目中的合作行为和参与者的贡献，提供了一个量化和评估开源社区合作和贡献价值的有效工具。

2.4　预测模型相关技术

在开源软件价值评估领域，预测模型起着决定性作用，旨在通过分析历史数据来预测项目的未来表现。这一过程涉及到复杂的数据处理和分析技术，需要精心选择合适的模型以准确捕捉项目的潜在价值。以下概述了构建预测模型时需考虑的关键环节以及在开源软件评估中的应用：

1. **数据预处理**是模型构建过程中的首要步骤，包括数据清洗、特征提取和特征选择等 [58]。在开源软件价值评估中，这可能涉及到从项目的提交记录、问题追踪和社区讨论等方面提取关键指标。
2. **模型选择**根据数据的特性和预测任务的需求选择适当的模型。常见的预测模型包括线性回归、决策树、随机森林、支持向量机和神经网络等 [59]。
3. **模型训练与验证**通过训练数据来调整模型参数，使用交叉验证等技术评估模型的泛化能力 [60]。
4. **结果评估**基于测试数据集，使用准确率、召回率、F1 分数等指标对模型性能进行评估 [61]。

在开源项目评估中，模型的选择需要综合考虑准确性、可解释性和计算效率 [62]。例如，线性回归模型虽然易于理解和解释，但可能无法处理复杂的非线性关系。相比之下，深度学习模型如神经网络能够捕获更复杂的数据模式，但其内部机制较为复杂，解释性较差 [63]。

线性回归（Linear Regression）、随机森林（Random Forest）、XGBoost（eXtreme Gradient Boosting）、以及神经网络（Neural Network）构成了现代统计学和机器学习领域的四大基石，各自在模型构建、预测准确性、以及处理复杂数据集方面展现出独特的优势和应用场景。

线性回归模型，作为预测分析的基础，依赖于线性方程来描述自变量和因变量之间的关系。该模型的参数估计通常通过最小化残差平方和来实现，适用于简单

到中等复杂度的数据关系分析。尽管在处理单一或多个自变量的情景中表现优异，线性回归在面对高维数据和非线性关系时的表现则相对受限 [59][64]。

随机森林，作为一种集成学习方法，通过构建多个决策树并采取平均预测或多数投票机制来提升预测的准确性。其独特的自助采样和特征随机选择机制有效提高了模型的泛化能力，使其在处理高维数据和非线性问题上具有显著的优势。然而，随机森林的模型解释性较弱，且在计算资源需求上相对较高 [62][65]。

XGBoost 进一步推进了梯度增强框架的发展，通过引入正则化项减少模型复杂度并避免过拟合，同时实现了特征列抽样、缺失值自动处理、深度优先的剪枝策略，及特征维度上的并行处理，显著提升了提高了模型的随机性、计算效率和泛化能力 [66]。XGBoost 在众多数据科学竞赛中的广泛应用证明了其对复杂数据集处理能力的强大性能。

神经网络，特别是全连接神经网络或多层感知机，通过引入隐藏层和非线性激活函数，极大地增强了模型捕捉复杂数据关系的能力 [67][63]。神经网络的这种灵活性使其在图像识别、自然语言处理等领域取得了重大突破。神经网络的非线性特性使其能够处理线性模型难以解决的高维和非线性问题。

总体而言，这几种模型各有优缺点，选择合适的模型需要考虑数据的特性、问题的复杂度、以及计算资源的可用性。在实际应用中，结合多种模型的集成学习方法往往能够提供更优的预测性能和模型鲁棒性。

2.5　自然语言处理

自然语言处理（Natural Language Processing, NLP）是计算机科学和人工智能的一个分支，NLP 集成了多种学科的理论和方法，专注于使计算机能够理解和处理人类语言。

NLP 的核心任务包括语言理解（从文字中提取意义）和语言生成（产生自然语言响应）。随着深度学习和大数据技术的发展，NLP 已经取得了重大进展，应用范

围也从文本分类、情感分析扩展到了机器翻译、自动摘要、问答系统等多个领域[68]。

NLP 的关键技术和方法如下：

1. **词法分析（Tokenization）**

   将文本拆分成更小的单位，如单词、短语或句子。这是 NLP 处理的第一步，为后续的语义分析奠定基础。

2. **句法分析（Parsing）**

   识别文本中的语法结构，如句子成分和依存关系。句法分析有助于理解句子的结构，为深层次的语义理解提供信息 [69]。

3. **语义分析（Semantic Analysis）**

   理解单词、短语和句子的含义，包括词义消歧、实体识别、关系抽取等。语义分析是理解人类语言真正含义的关键步骤 [70]。

4. **情感分析（Sentiment Analysis）**

   识别和提取文本中的情绪倾向，广泛应用于社交媒体分析、市场研究等领域[71]。

5. **机器翻译（Translation）**

   将一种自然语言翻译成另一种自然语言，是 NLP 的复杂应用之一。近年来，基于深度学习的神经机器翻译（NMT）模型在这一领域取得了显著成就 [72]。

深度学习为 NLP 的发展带来了革命性的变化。特别是预训练语言模型，如 BERT（Bidirectional Encoder Representations from Transformers）和 GPT（Generative Pre-trained Transformer），通过在大规模文本数据上预训练，能够捕捉到丰富的语言规律和知识，然后在特定任务上进行微调，大幅提升了 NLP 任务的性能 [73][74]。

除了 BERT 和 GPT 外，其他如 XLNet、RoBERTa 和 ELECTRA 等模型也在 NLP 的多个领域中展示了其优越性，包括文本分类、情感分析、问答系统、自然语言推

理等 [75][76][77]。这些模型进一步推动了 NLP 技术的边界，使得从复杂文本中提取有用信息变得更加高效和精确。

预训练语言模型之所以强大，部分原因在于它们采用了 Transformer 架构，该架构通过自注意力机制（Self-Attention Mechanism）有效地处理序列数据，捕获长距离依赖关系 [78]。此外，这些模型的另一个关键优势是它们的微调能力，即在针对特定任务进行少量训练后能够适应并优化任务性能 [73][74]。

2.5.1 Transformer 模型

Transformer 模型，首次由 Vaswani 等人在 2017 年的论文《Attention is All You Need》[78] 中提出，已经成为自然语言处理（NLP）领域的一个重要里程碑。Transformer 的核心思想是利用自注意力机制（Self-Attention Mechanism）来处理序列数据，摒弃了之前主流模型依赖的循环神经网络（RNN）和卷积神经网络（CNN）结构，实现了更高效的并行计算和更好的长距离依赖捕捉能力。

自注意力机制是 Transformer 的核心，它允许模型在处理序列的每个元素时，考虑到序列中的所有其他元素，从而捕获序列内各元素间的复杂关系。

由于 Transformer 完全依赖于自注意力机制，缺乏序列的顺序信息。为了补充这一信息，Transformer 在输入嵌入向量中加入位置编码，位置编码与嵌入向量有相同的维度，这样模型就能根据相对或绝对位置来获取词的顺序信息。

Transformer 模型的提出极大地推动了 NLP 领域的发展，其架构和机制为处理复杂的语言处理任务提供了强大的工具，也为后续的模型创新奠定了基础。BERT 就是其中最重要的分支之一。

2.5.2 BERT 模型

2018 年，Google 提出了一种新的 NLP 模型——双向编码器表示（BERT），它通过预训练深度双向表示从无标记文本中学习，然后对这些表示进行微调，以执行广泛的语言处理任务 [73]。BERT 模型的出现标志着 NLP 领域的一个重大突破，它

在多个 NLP 任务上都取得了显著的性能提升。在开源软件分析领域，BERT 模型可以用于理解和分析 README 文件、代码注释和开发者交流中的复杂语言模式，提取关键信息和知识，从而为项目管理、质量保证和社区建设提供支持。

BERT 模型的核心创新在于利用 Transformer 的编码器架构，预训练深度双向表示，从而在广泛的 NLP 任务上实现了显著的性能提升。BERT 的成功部分归因于其能够捕获丰富的双向上下文信息，这是之前的模型所不具备的。

BERT 模型的预训练包括两个阶段：Masked Language Model (MLM) 和 Next Sentence Prediction (NSP)。

1. **Masked Language Model (MLM)** 在 MLM 任务中，输入文本的一部分词被随机替换为一个特殊的 [MASK] 标记，模型的任务是预测这些被遮蔽的词。具体而言，给定一个句子 S，将其中的一些词替换为 [MASK]，BERT 模型输出这些 [MASK] 位置上每个词的概率分布，模型的目标是最大化被遮蔽词的对数似然概率。

2. **Next Sentence Prediction (NSP)** NSP 任务旨在预测两个句子是否是连续的文本片段。在预训练阶段，给定一对句子 A 和 B，模型需要预测 B 是否是 A 的下一句。这个任务通过在句子对前添加特殊标记 [CLS]，并在两个句子之间插入 [SEP] 标记来实现。[CLS] 标记位置的输出向量被用来预测句子对是否连续。

由于 Transformer 架构不具有捕捉序列顺序的能力，BERT 通过加入位置编码来注入位置信息。这些位置编码被加到输入嵌入向量上。其中，[CLS] 标记（Classification Token）便是一个非常重要的位置编码，尤其是在处理分类任务时。当输入序列被送入 BERT 模型进行处理时，[CLS] 标记被添加到序列的最前面。模型通过训练学习到的表示，[CLS] 标记的最终状态被用作整个输入序列的聚合表示，从而在进行分类或其他任务时捕捉到全局的上下文信息。

例如，在句子分类任务中，[CLS] 的输出向量（即 Transformer 的最后一层中 [CLS] 位置的隐藏状态）被用来预测句子的类别。这是通过在 [CLS] 的输出向量上添加一个全连接层（或多个层）和 softmax 层来实现的，从而将这个向量映射到预定类别的概率分布上。

BERT 的训练过程通过最小化预训练任务（如 Masked Language Model 和 Next Sentence Prediction）的损失函数来进行，从而学习深层次的双向表示。在预训练之后，通过微调过程，BERT 可以被适配到广泛的下游 NLP 任务上，实现出色的性能。

预训练完成后，BERT 模型可以通过微调（fine-tuning）过程适应特定的下游任务。微调阶段使用预训练的参数作为初始化，然后在特定任务的训练数据上继续训练。对于不同的任务，只需在 BERT 的输出层添加适当的任务特定层（如分类层、序列标注层等），并在此基础上进行训练即可。

2.5.3　README 文本的重要性与分析方法

README 文件作为开源项目的门面，为用户和开发者提供了项目概述、安装指南、使用说明和贡献指南等关键信息。因此，README 文件的质量直接影响到项目的可接受性和用户满意度。使用 NLP 技术分析 README 文本可以帮助评估其信息完整性、易读性和有效性。通过应用如 BERT 等深度学习模型，研究人员可以自动识别 README 文件中的信息缺失、潜在的歧义和改进空间，从而指导开发者优化文档内容 [79]。

通过对 README 文件和其他文档的深入分析，NLP 不仅可以提升软件文档的质量，还可以促进开源社区的健康发展，增强用户参与度和项目的影响力。

2.6　本章小结

本章回顾了开源软件与商业开源公司的关键概念，明确了开源软件的定义、特性，以及商业开源公司的商业模式和它们对全球技术生态系统的重大贡献。特别强调了社区驱动的开发模式对于促进技术创新和降低软件开发维护成本的重要性。

商业开源公司通过多样化的商业实践，如提供付费支持、专有扩展和基于开源的 SaaS 模型，展示了开源与商业模式的融合和创新。

在探讨开源软件社区和影响力时，本章详细介绍了社区参与度和项目影响力的重要性，并通过 OpenRank 指标来量化这些因素，展现了评估开源项目和贡献者影响力的先进方法。此外，通过引入自然语言处理（NLP）的最新技术，如 Transformer 和 BERT 模型，本章展示了如何利用深度学习提高对开源项目文档的理解和分析能力，特别是 BERT 模型中的 [CLS] 标记在理解和分类任务中的关键作用被突出讨论。[CLS] 标记作为模型理解整个输入序列的聚合表示的代表，对于执行各种 NLP 任务至关重要，体现了深度学习在提升文本分析精度方面的潜力。

在预测模型的讨论中，本章不仅覆盖了线性回归、随机森林、XGBoost 等传统预测模型，还包括了简单神经网络的介绍，准备了对 NLP 在开源软件分析中应用的理论基础。这些内容为开发基于深度学习的开源软件价值评估模型提供了必要的背景知识。

对传统价值评估方法的分析揭示了直接收益法、成本法和市场比较法在应用于开源项目评估时的局限性，强调了对比法在考虑开源特性时的适应性。然而，现有方法在量化开源项目的社区活跃度、技术影响力等方面面临挑战，指出了未来研究的方向。

通过本章的讨论，本研究为后续章节关于开源软件价值评估模型的开发提供了理论与实践基础，旨在为开源社区的管理者、开发者提供有价值的见解，以促进开源项目的健康发展和社区的积极参与。

第三章　支持开源软件商业价值评估任务的基准数据集构建

本章专注于构建一个基准数据集来支持开源软件商业价值评估任务。此任务的核心在于深入理解开源社区的行为、商业操作及其文本数据的来源、采集方法和策略，并通过这些数据对其商业价值进行评估。鉴于此，本研究关注的数据为开源软件商业、协作网络与仓库文本数据。

有效的数据采集不仅为本研究奠定了实证基础，而且为后续的分析工作提供了必要的广度和深度。具体的，本章介绍开源软件社区、行为、商业及文本数据的模型、采集途径与方案。

3.1　基础数据来源

3.1.1　开源软件公司商业数据

商业开源公司数据对于理解开源项目的商业价值和市场影响至关重要。这部分的数据采集分为两个主要部分：基础的 COSS 投资数据和开源公司的商业数据，每部分采用了不同的方法和来源，以确保数据的全面性和准确性。

对于基础的商业开源软件（COSS）投资数据，采集主要依赖于 COSS.community 的定期更新数据集。该数据集虽然提供了商业开源公司投资情况的基本信息，但存在一定的局限性，例如信息的不全面和采集的技术限制。特别地，由于 COSS.community 网站通过 Google Doc 设置的反爬虫机制及权限限制，直接下载或通过网页爬虫技术导出数据存在困难。

于是为解决这一问题，在采集过程中，本研究采用了结合录屏软件与动作助手软件截图及 OCR（光学字符识别）技术，后续通过人工校对以保证数据的准确性 [80]。此方法虽然较为繁琐，却确保了数据的高质量和可靠性，为研究提供了坚实的基础。更多关于 COSS.community 的信息可以在其官方网站查阅 COSS.community[1]

[1]网址：`https://coss.community/`

中的更新原始数据展示窗口 All Publicly-Announced Global VC Funding into COSS from January 2020。

另一方面，为补充和完善基础投资数据，也由于商业公司的更多数据主要来源于 CrunchBase 和 Wind 商业数据网站，通过这一途径可以极大地完善之前的数据。CrunchBase 作为权威的创业公司和投资信息数据库，提供了丰富的商业开源公司信息，包括但不限于公司概况、融资轮次、投资者信息及市场表现。CrunchBase 的详细信息可在 CrunchBase 网站 [2] 获得。

同时，Wind 商业数据网站则为研究者提供了更广泛的财经数据和商业信息，支持对开源公司的商业表现进行全面分析。Wind 信息覆盖了从股票市场表现到财务报表的细节，是分析开源公司商业成功和市场策略的重要工具。Wind 的更多信息可以在其 Wind 网站 [3] 查阅。

通过结合这两个部分的数据采集途径，能够从多个维度理解商业开源公司的市场表现和投资情况，为后续的分析和评估提供了全面的数据支持。这一数据采集工作不仅展示了开源项目在商业领域的潜力和影响力，也为开源社区的持续发展和商业化探索提供了实证基础。

3.1.2　开源软件行为数据

开源软件行为数据的采集依赖于三个主要途径，每个途径都针对开源软件的不同方面，提供了独特的分析视角。

GH Archive 作为 GitHub 公开活动的历史存档，为研究开源软件提供了丰富的活动数据。这些数据包括项目提交、问题讨论、拉取请求等，是理解开源社区动态和项目活跃度的宝贵资源 [81]。更多关于 GH Archive 的信息，请访问其官方网站：GH Archive[4]。

[2]网址：https://www.crunchbase.com/

[3]网址：http://www.wind.com.cn/

[4]网址：https://www.gharchive.org/

REST API 作为 GitHub 官方提供的 API 接口允许研究者直接查询特定项目的详细信息和活动记录，支持实时数据采集。这一途径特别适合于进行个案研究或追踪特定项目的发展 [82]。GitHub REST API 的详细文档可在 GitHub Developer 网站[5]查阅。

REST API 对用户的限制为 5000 次调用每小时，这一个数字实际上也会因人而异，如果是新创建的用户可能只有 3000 次每小时，如果过度频繁调用则新用户可能还会被封禁。所以在进行并行请求的过程中，需要精细地分配和定时暂停。

华东师范大学 X-lab 实验室的 Open-Digger 项目通过整理和聚合 GH Archive 数据，提供了便于分析的数据集。Open-Digger 不仅简化了数据处理的复杂度，还提升了数据分析的效率和准确性，对于理解开源项目的发展趋势、社区活跃度及网络信息具有重要价值。

如图3.1所示，Open-Digger 的数据使用 GH Archive 作为外部数据集，其流程如下：

1. 由于 GH Archive 以每小时的频率更新数据，OpenDigger 也在每个小时监测一次 GH Archive 的 API 查看是否有数据更新，如果有更新则进行抓取。
2. 将新抓取的数据在内部临时存储区域进行数据预处理，预处理过程包括：
   - 解压原始数据包；
   - 解析 json 格式为整体的文本数据以节省后续的网络 IO；
3. 将预处理后的数据以块的形式输入到裂数据库 ClickHouse 中。

最终 ClickHouse 中存储并在调用时呈现的是一条条基本的行为数据以及定期进行聚合计算的衍生数据，用户如果想查询基本的行为数据则只需要直接访问 ClickHouse。

Open-Digger 项目的更多信息可在 Open-Digger GitHub 页面[6]找到。

[5]网址：https://docs.github.com/en/rest

[6]网址：https://github.com/X-lab2017/open-digger

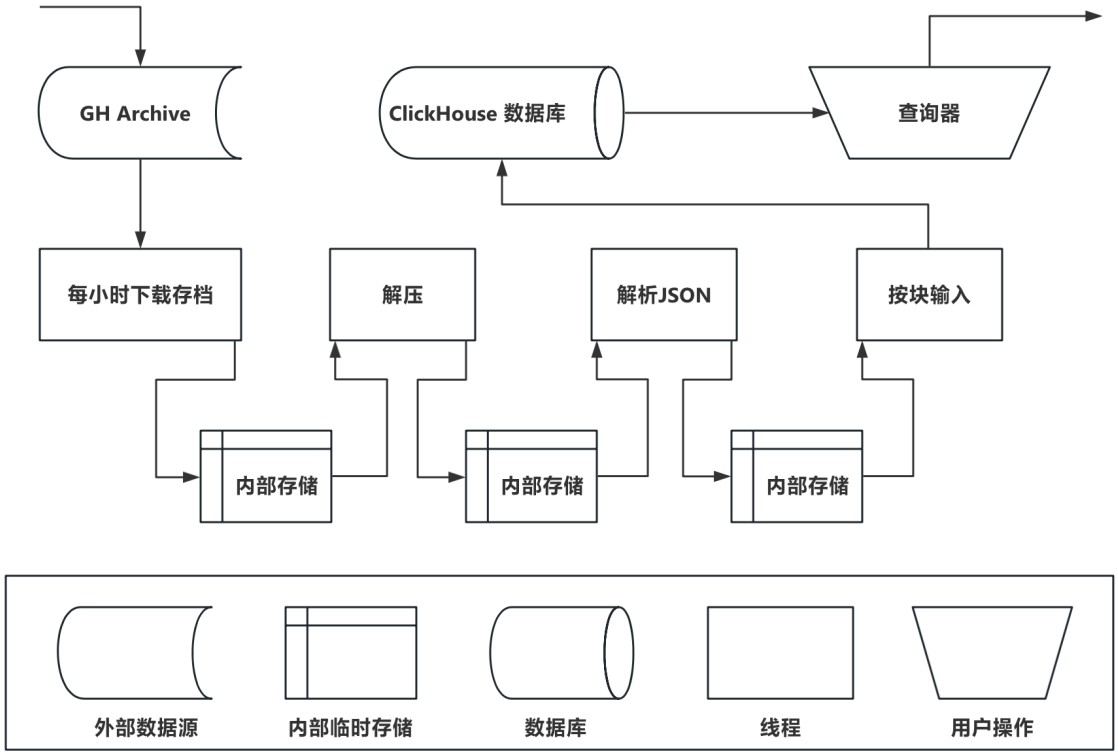

图 3.1 Open-Digger 数据处理流程

3.1.3　开源软件协作网络特征数据 - Dynamic OpenRank

在第二章中本文介绍了 OpenRank 这一网络结构特征指标，本章将引入一个 OpenRank 的变体。

由于 OpenRank 是一个类似 PageRank 的 Markov 过程算法。其对所有节点 v_i 的迭代计算公式均为

$$v_i = (1 - a_i) \sum_{j=1}^{|V|} \frac{w_{ji}}{d_{oj}} v_j + a_i v_0 \tag{3.1}$$

其中，v_0 代表节点的初始值，a_i 表示节点依赖其初始值的程度，d_{oj} 是节点 j 的加权出度，w_{ji} 是从节点 j 到节点 i 的边的权重。OpenRank 算法通过其对角线均为 $a_i > 0$ 确保其收敛性。但现实生活中，不同开源软件社区网络内部的转移概率是不同的，设置成同一个系数是一种直觉但不够细致的做法。

于是本研究对 OpenRank 算法进行改进，允许不同类型的开源软件网络子图的阻尼系数设置为不同的数。本研究命名该算法为 Dynamic OpenRank。

由于 Dynamic OpenRank 和 OpenRank 都是类似于 PageRank 的 Markov 过程算法，所以其收敛性是算法得以正常实施的关键因素。以下证明其在阻尼系数均不为 0 的情况下一定收敛：

转移矩阵 P，在 Dynamic OpenRank 算法中，表示节点间跳转的概率。对于任何页面 i 到页面 j 的转移概率是由节点 i 的阻尼因子 d_i 和链接结构决定。转移矩阵的性质直接影响到算法的收敛行为：

1. **随机性（Stochasticity）**转移矩阵 P 是随机的，意味着其每一行的和等于 1。这反映了一个页面到其他页面跳转概率的总和为 1。
2. **非负性（Non-negativity）**P 中的所有元素都非负，反映了概率的本质。
3. **稀疏性（Sparsity）**由于网页之间的链接结构，转移矩阵通常是稀疏的，即大多数元素为 0。

为了保证 PageRank 算法的收敛，转移矩阵 P 必须满足几个关键条件：

1. **不可约（Irreducibility）**如果从任一页面到另一页面都存在跳转路径，则称转移矩阵为不可约的。这保证了网络是完全连接的，从而避免了算法陷入局部最优。
2. **非周期性（Aperiodicity）**转移矩阵是非周期的，意味着从任一页面出发，返回到该页面的路径长度没有共同的周期。这是通过引入阻尼系数并保证至少存在一条路径可以直接或间接返回到自身来实现的。
3. **稳定性（Stability）**引入阻尼系数 d 并设置为小于 1 的值（通常为 0.85），是为了保证算法的稳定性和收敛性。阻尼因子引入了随机跳转的可能性，确保了即使在链接结构不完美的情况下也能收敛。

对于 Dynamic OpenRank 算法定义的转移概率矩阵 P，其元素 P_{jk} 代表了从页面 p_k 到页面 p_j 的转移概率。根据 Markov 链的理论，如果矩阵 P 是不可约且

非周期的，那么该 Markov 链将收敛到一个唯一的稳定分布，这保证了 Dynamic OpenRank 值的收敛性。

在不同区域阻尼系数不同的设置下，收敛性取决于转移概率矩阵 P 的性质，特别是其谱半径（即最大绝对值特征值）。

定理 3.1.1. *对于任意的阻尼因子配置 $\{d_i\}$，如果所有的 d_i 都满足 $0 < d_i < 1$，则转移概率矩阵 P 的谱半径小于 1，从而保证了 PageRank 值的收敛性。*

证明. 基于 Gershgorin 圆盘定理，可以证明转移概率矩阵 P 的每个特征值都落在以 d_i 为半径的圆内。因为 $0 < d_i < 1$，所有圆的中心都位于实轴上，半径小于 1，因此所有特征值的绝对值都小于 1，这保证了谱半径小于 1。 □

虽然 Dynamic OpenRank 算法在理论上保证了收敛，但不同阻尼因子的配置会影响收敛的速度。较高的阻尼因子可能会加快接近稳定分布的速度，但也可能使算法更敏感于初始 Dynamic OpenRank 值的选择。通过适当选择区域阻尼系数，可以在加速收敛和保持算法稳定性之间找到一个平衡点。

3.1.4　开源软件仓库文本数据

GitHub 的 REST API 提供了一种直观且强大的方式来访问 GitHub 上几乎所有的类型的公开数据。通过构造精确的 API 请求，研究者可以定向地获取特定开源项目的 README 文件，以及其他相关的项目信息，如贡献者列表、提交历史、issue 讨论等。README 文本数据的采集通过 GitHub 的 REST API 完成，这一途径能够高效地获取最新的 README 文本内容，对于使用自然语言处理技术分析项目文档提供了数据基础 [83]。

3.2　数据模型及特征指定

3.2.1　数据模型

针对开源软件商业、协作网络与仓库文本数据的模型构建，本研究通过图3.2展示一个围绕开源软件公司及其核心开源组织和项目的协作网络、商业关系、以及

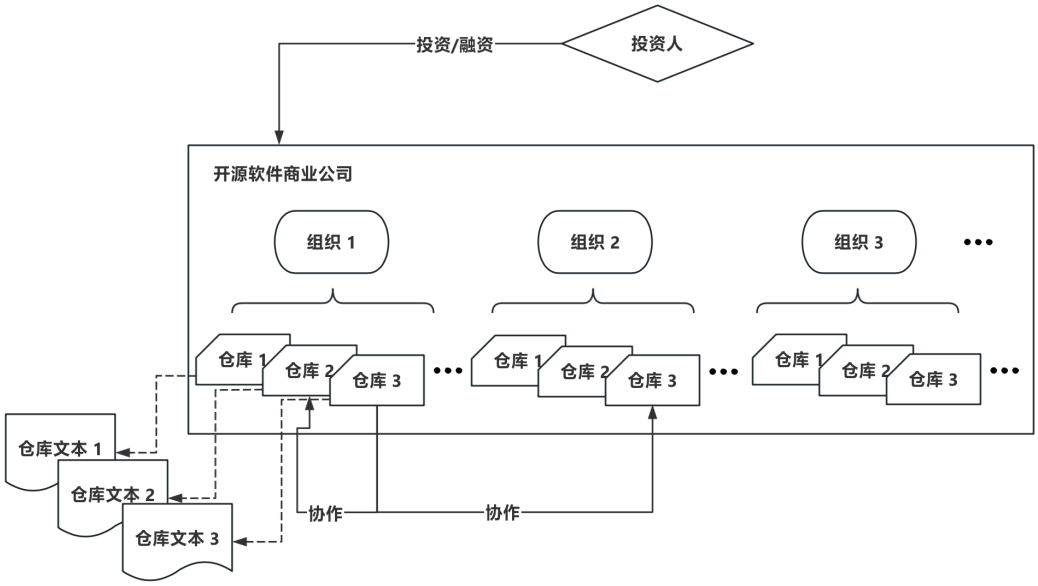

图 3.2 开源软件商业、协作网络和仓库文本模型

文本关系的数据模型。在该模型中，投资人投资商业开源软件公司；这些商业开源软件公司拥有一个或多个核心开源软件组织，每个开源软件组织包含多个开源软件项目，每个开源软件项目与另外的开源软件项目拥有协作关系，形成复杂的协作网络；每个开源软件仓库拥有一个仓库文本信息（通常为 README）。

这一综合数据模型有效地捕捉了开源项目中商业操作、社区协作和技术实践之间的复杂互动，为深入分析开源软件的商业价值和社区影响力提供了一个数据基础。

3.2.2　数据样例

本节将详细介绍商业开源公司数据的样例，这些数据样例反映了本研究在采集和分析过程中的关键发现。这些样例旨在展示如何从基础的 COSS 投资数据到开源公司的具体商业数据，以及如何通过 REST API 获取核心 GitHub 组织和项目的数据，进一步分析开源社区的活跃度和项目文档的质量。

（1） 商业开源公司数据

商业开源公司数据分为两个部分：基础的 COSS 投资数据和开源公司的商业数据。

其中，基础的 COSS 投资数据主要包括了 2020 年以后的开源软件项目和公司的投资轮次、投资金额、投资方等信息。这些数据通过录屏软件与动作助手软件结合 OCR 技术及人工审核获取，确保了数据的准确性和完整性。例如，图3.3展示了几个开源公司在网络上宣布的投融资增长轨迹，包括每轮融资的时间、金额和主要投资方。

1	© 2023 OSS Capital LLC	This data is maintained continuously by OSS Capital since January 2020. Please email sam@oss.capital to add/correct records or provide feedback.					
2	**COSS Company**	**FOSS Core(s)**	**Stage**	**Round Size (M)**	**Month Announced**	**Year Announced**	**Lead Investor**
3	KSOC	Kubernetes	Seed	$6.0	February	2022	.406 Ventures
4	Qdrant	Qdrant	Seed	$2.2	January	2022	42CAP, IBB Ventures
5	Earthly	Earthly	Seed	$2.0	October	2020	468 Capital
6	QuestDB	QuestDB	A	$12.0	November	2021	468 Capital
7	Activeloop	Hub	Seed	$5.0	November	2021	468 Capital
8	Digital Asset	DAML	D	$120.0	April	2021	7RIDGE
9	Yugabyte	YugabyteDB	C	$30.0	June	2020	8VC
10	Acryl Data	DataHub	Seed	$9.0	June	2021	8VC
11	Acryl Data	Acryl Data	A	$21.0	July	2023	8VC
12	R		A	$20.0	February	2021	A Capital
13	P		A	$15.0	September	2023	a16z
14	R		B	$40.0	December	2023	a16z

Acryl Data
acryldata.io

图 3.3 COSS.community 所展示的基础 COSS 投资原始数据

有所不同的是，开源公司的商业数据是由商业的专业数据收集公司获得。所以进一步的，通过 Wind 和 Crunchbase 获取的这些开源公司所有年份的商业数据包括了财务数据、市场表现、投资情况等详细信息。这部分数据需要付费获取，但也提供了从商业角度准确且深入理解开源公司的机会。如图3.4所示，展示了一家开源公司 PingCAP 自成立以来的所有融资情况，包括更加精确的融资规模、融资时间、主要投资方及融资轮次。而如图3.5所示，展示了一家公司 Copper 每年的融资规模和融资轮次。

（2） 开源软件数据

本节详细介绍了开源软件数据及其 README 文本数据的采集方法。采用 GH Archive、GitHub REST API 和 OpenDigger 等多种方法。

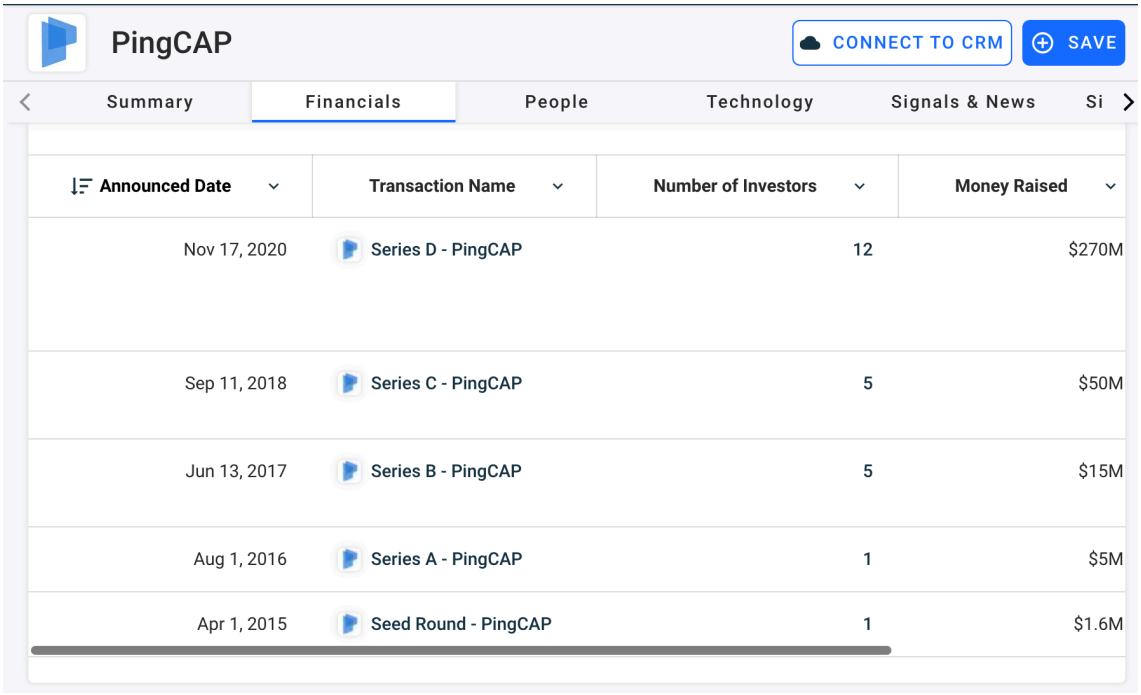

图 3.4 CrunchBase 公司商业数据样例（以 PingCAP 为例）

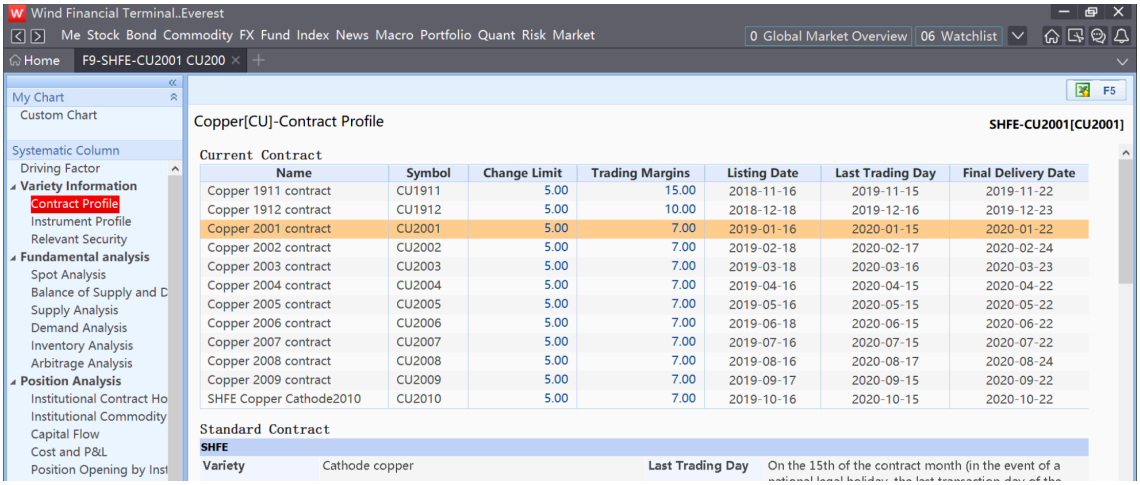

图 3.5 Wind 公司商业数据样例（以 Copper 为例）

GH Archive 是一个广泛使用的开放数据集，它记录了 GitHub 上的公开活动。这一资源对于研究开源软件项目的发展趋势、社区活跃度以及开发者行为模式等方面提供了宝贵的信息。GH Archive 通过捕获 GitHub 事件 API 提供的数据，为研究人员提供了一种方便的方式来分析 GitHub 上的公开活动数据。

表3.1所展示的是 GH Archive 的数据架构描述，它详细列出了数据集中包含的字段、类型以及相应的描述信息。这些字段包括事件类型、是否为公开活动、事件负载（以 JSON 格式呈现）以及与事件相关联的仓库、参与者和组织的信息。此架构描述对于理解数据集的结构和内容至关重要，有助于研究人员设计和实施他们的分析策略。

表 3.1 GH archive 架构描述

Name	**Type**	**Description**	**Notes**
type	STRING	Event type as per `https://developer.github.com/v3/activity/events/types/`	
public	BOOLEAN	Always true for this dataset since only public activity is recorded.	
payload	STRING	Event payload in JSON format	
repo.id	INTEGER	Numeric ID of the GitHub repository	Part of ‘repo‘ record
repo.name	STRING	Repository name	Part of ‘repo‘ record
repo.url	STRING	Repository URL	Part of ‘repo‘ record
actor.id	INTEGER	Numeric ID of the GitHub actor	Part of ‘actor‘ record
actor.login	STRING	Actor’s GitHub login	Part of ‘actor‘ record
actor.gravatar_id	STRING	Actor’s Gravatar ID	Part of ‘actor‘ record
actor.avatar_url	STRING	Actor’s Gravatar URL	Part of ‘actor‘ record
actor.url	STRING	Actor’s profile URL	Part of ‘actor‘ record
org.id	INTEGER	Numeric ID of the GitHub org	Part of ‘org‘ record
org.login	STRING	Org’s GitHub login	Part of ‘org‘ record
org.gravatar_id	STRING	Org’s Gravatar ID	Part of ‘org‘ record
org.avatar_url	STRING	Org’s Gravatar URL	Part of ‘org‘ record
org.url	STRING	Org’s profile URL	Part of ‘org‘ record
created_at	TIMESTAMP	Timestamp of associated event	
id	STRING	Unique event ID	
other	STRING	Unknown fields in JSON format	

通过这一数据架构，GH Archive 为开源社区的研究提供了一个丰富的数据源，支持从多个维度对 GitHub 上的公开活动进行深入分析和理解。

针对 REST API，本研究所采用的技术手段是 PyGithub 库，这是一个强大的

Python 库，它封装了 GitHub 的 REST API，使得从 GitHub 获取数据变得更加便捷和高效。

为了深入理解开源软件的社区活跃度及开发者参与情况，本研究采用了 PyGithub 库来获取特定仓库的开放问题（issues）。这些数据能够提供关于社区成员关注点及活跃讨论话题的重要信息。以下是获取开放问题的 Python 代码及其输出示例：

相应的，GitHub 用户的信息获取方式、代码及结果如下所示：

```
>>> repo = g.get_repo("PyGithub/PyGithub")
>>> open_issues = repo.get_issues(state='open')
>>> for issue in open_issues:
...     print(issue)
...
Issue(title="How to get public events?", number=913)
Issue(title="Prevent .netrc from overwriting Auth
   ↪ header", number=910)
Issue(title="Cache fetch responses", number=901)
Issue(title="Is suspended_users for github enterprise
   ↪ implemented in NamedUser?", number=900)
Issue(title="Adding migration api wrapper", number=899)
```

代码清单 3.1 GitHub REST API 调用示例

本研究重点关注 Open-Digger 数据库中的 events 和 OpenRank 两个表。这些数据集为开源软件社区的行为模式和网络结构特征提供了丰富的信息，支持对开源项目活动和社区动态的深入分析。

events 表记录了开源社区中发生的各种事件，如 Issues 的创建、Pull Requests 的提交等，反映了社区的活跃度和开发者的贡献模式。events 表结构包含事件类型（type）、事件参与者（actor_id、actor_login）、项目信息（repo_id、repo_name）等字段，为分析开源项目的活动模式提供了基础数据。

OpenRank 表提供了开源项目及其参与者的影响力评分，该评分基于项目活动、社区参与度等多个维度计算得出，是衡量项目重要性和影响力的重要指标。OpenRank 表结构则包括平台（platform）、仓库 ID（repo_id）、仓库名（repo_name）、用户 ID（actor_id）、用户登录名（actor_login）以及 OpenRank 评分（openrank），为评估项目及参与者的影响力提供了详细信息。

如下是 Open-Digger 的 events 的数据结构及其调用方法。

```
>>> default_table = "opensource.events"
>>> get_year_constraint = lambda x, y=None: f"created_at
   ↪ BETWEEN '{str(x)}-01-01 00:00:00' AND '{str(y or
   ↪ (x+1))}-01-01 00:00:00'"
>>> conndb.sql = f'select * FROM {default_table} where
   ↪ {get_year_constraint(2022)} LIMIT 10;'
>>> conndb.execute()
...
platform    id  type    action  actor_id    actor_login
   ↪ repo_id repo_name   org_id  org_login   ...
   ↪ commit_comment_updated_at
0   GitHub  23454517593 CreateEvent added   22234154
   ↪ noorprajuda 522440625   rivaldiheriyan27/Koma
   ↪ 0         ... None
10 rows × 141 columns
>>> list(conndb.rs.columns)
...
['platform', 'id', 'type', 'action', 'actor_id',
   ↪ 'actor_login', 'repo_id', 'repo_name', 'org_id',
   ↪ 'org_login', 'created_at', 'issue_id',
   ↪ 'issue_number', 'issue_title', 'body',
```

```
...
```

代码清单 3.2 Open-Digger 调用 events 数据及输出示例

（3） 开源软件 README 文本数据

对于开源项目而言，README 文件是项目的“门面”，它向潜在的用户和贡献者提供了项目的简介、使用说明、贡献指南等关键信息。因此，获取和分析 README 文本数据对于评估项目的可访问性和用户友好性至关重要。以下是使用 PyGithub 库获取 README 文件内容的 Python 代码及其输出示例：

```
>>> repo = g.get_repo("PyGithub/PyGithub")
>>> contents = repo.get_contents("README.md")
>>> print(contents)
...
ContentFile(path="README.md")
```

代码清单 3.3 获取 GitHub 仓库 README 文件内容的代码及输出示例

通过上述方法采集到的开源软件数据和 README 文本数据，本研究采用了标准化的数据整理流程，以确保数据的一致性和可分析性。如图3.6所示，对于 README 文本数据，将其整理为以下格式：“< 项目 ID>b’<README 内容 >’”，其中“项目 ID”用于唯一标识一个 GitHub 项目，“README 内容”为对应项目 README 文件的实际文本内容。这种标准化的格式便于在后续的研究中对数据进行查询、分析和比较。

3.3 基准数据构建

3.3.1 数据采集

需要注意的是，由于最初搭建数据集的过程中网络采集的数据量非常大且频率非常高，所以极易被检测并反爬虫机制限制。因而在初次的数据采集过程中需要多次考虑如何规避反爬虫机制。

图 3.6 整理后的项目 README 数据集，形式：< 项目 ID>b'<README 内容 >'

（1） 数据采集流程

初次搭建数据集的流程会较为繁琐，且因为每一步请求的数据量都非常大所以需要规避反爬虫机制。

1. 基础的 COSS 投资数据采集

   作为数据采集的初步阶段，首先通过录屏软件与动作助手软件结合 OCR（光学字符识别）技术进行数据截取，随后通过人工审核的方式确保数据的准确性和完整性。此步骤主要针对 COSS.community 提供的定期更新数据集，旨在获取开源软件项目和公司的基础投资信息。

2. 开源公司商业数据的获取

   接下来，利用 Wind 和 Crunchbase 这两个权威的商业数据平台深入获取开源公司的具体商业数据，包括但不限于财务数据、市场表现、投资情况等。这一步骤为后续的分析提供了丰富的商业背景信息。这一部分需要付费进行。同时，可以通过这一步骤增强原有数据集，将其规模上从只有 2020 年以后的数据扩展到 2015 年以后的数据。

3. 核心 GitHub 组织和项目的识别

   基于前两步获取的商业数据，进一步通过 GitHub 的 REST API 精确识别相关的 GitHub 组织和项目。此步骤确保了数据采集的针对性和高效性，为行为数据的采集奠定了基础。

4. 行为数据的采集

   通过 GH Archive、GitHub 的 REST API 以及华东师范大学 X-lab 实验室处理过的 Open-Digger 数据，全面收集了目标开源软件组织和项目的行为数据。这些数据涵盖了项目提交、问题讨论、拉取请求等，为理解开源社区的活跃度和项目发展提供了关键性指标。

5. README 文本数据的获取

   最后，利用 GitHub 的 REST API 专门获取了目标组织和项目的 README 文本信息。这一步骤为使用自然语言处理技术分析项目文档提供了数据基础，有助于评估项目的可接入性和文档质量。

由于 COSS.community 的每次更新都是由这个组织的工作人员手动输入的，所以每次的更新点以及更新内容质量都存疑，所以在初次搭建数据集之后的每次后续更新都必须完整的重复初次搭建流程中的第一步「基础的 COSS 投资数据采集」，而其他数据则不必完整的重复，只需要将更新扩充部分的开源软件项目和组织走第二至五步流程，进行小规模的采集即可。

而且由于采集的规模较小，所以后续的更新不再需要规避反爬虫机制。

（2） 数据采集框架

本研究的数据采集框架采用了模块化设计，每个步骤都针对特定的数据类型和来源，通过层层递进的方式逐步深入，最终形成了一个全面、系统的开源软件社区与商业数据集。如流程图3.7所示，该框架不仅考虑了数据的广度和深度，还特别强调了数据质量的控制，确保了后续分析的可靠性和有效性。

3.3.2 数据预处理

数据预处理是研究过程中至关重要的一步，其目的是通过一系列处理步骤提高数据质量，从而为后续的分析和模型构建提供可靠的基础。本研究涵盖了商业开源公司数据、开源软件数据和开源软件 README 文本数据三个主要部分的预处理工作。

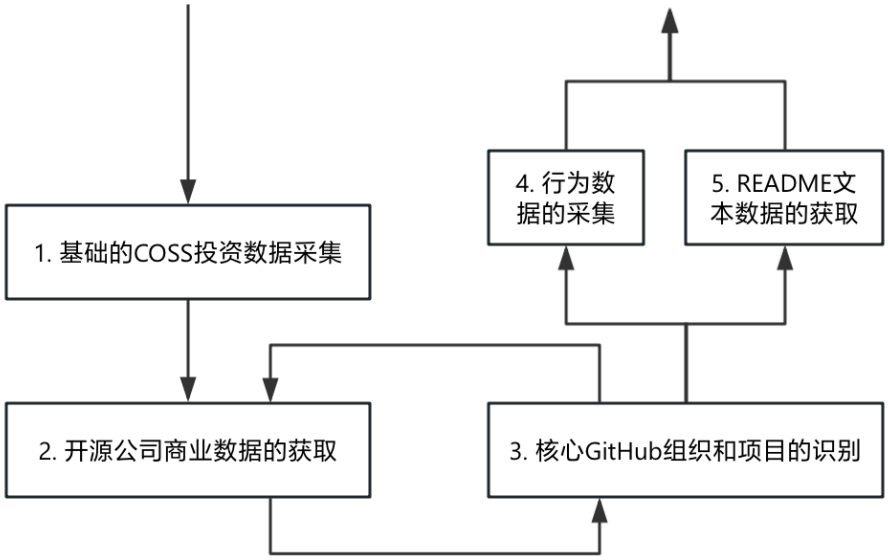

图 3.7 数据采集框架

（1） 商业开源公司数据预处理

由于商业开源公司数据分为两个部分：基础的 COSS 投资数据和开源公司的商业数据。数据预处理这一步骤将分别对其采取不同的措施流程。

针对基础的 COSS 投资数据，由于该数据来源 COSS.community 的数据本身也是由其工作人员手写输入的，所以会出现非常明显的失误，同时由于是每周更新，所以会出现前几周的输入习惯、命名方式和后几周的不一样这样的错误情况。于是，对于商业开源公司数据，预处理步骤主要包括数据清洗、缺失值处理、手动校对和数据归一化。

1. 数据清洗

   通过移除重复记录、修正明显的输入错误，确保数据的准确性和一致性。

2. 缺失值处理

   对于缺失的数据，本研究在数据为无序离散序列情况下进行众数处理，在有序序列的情况下进行平均数处理。

3. 手动校对

   由于仍然有大部分的数据准确性存疑，例如会出现融资公告撤稿；公司项目

组织改名；公司改名等情况，所以还是需要手动一一校对。

4. 数据归一化

为了消除不同规模和量纲的影响，对数值型数据进行归一化处理，使其范围统一到 [0, 1] 或 [-1, 1] 之间。

这些预处理步骤有助于提高数据分析的准确性和效率，为后续的商业分析和模型建立提供了坚实的基础。

针对开源公司商业数据，由于 Wind 和 CrunchBase 是面向多国使用的平台，所以如表3.2所示，其给出的融资规模可能会是以不同的货币计算的，所以此处需要将其全部换算成美元。

表 3.2 CrunchBase 待预处理数据

Company	Date	Round	Amount	Investors
Akeneo	Mar 15, 2022	Series D	$135M	Summit Partners
Akeneo	Sep 12, 2019	Series C	$46M	Summit Partners
Akeneo	Dec 19, 2018	Corporate Round	—	Salesforce Ventures
Akeneo	Mar 20, 2017	Series B	$13M	Partech
Akeneo	Sep 23, 2014	Seed Round	$2.3M	—
Akeneo	Jul 26, 2013	Seed Round	€100K	—
Akuity	May 16, 2022	Series A	$20M	Decibel Partners, ...
Akuity	Oct 11, 2021	Seed Round	$4.5M	Decibel Partners
Aleph Alpha	Nov 6, 2023	Series B	$500M	Bosch Ventures, ...
Aleph Alpha	Jul 18, 2023	Corporate Round	—	—
Aleph Alpha	Jun 30, 2023	Series B	€100M	—
Aleph Alpha	Jul 27, 2021	Series A	€23M	Earlybird VC, Lakestar, ...
Aleph Alpha	Jan 27, 2021	Seed Round	€5.3M	468 Capital, ...
Aleph Alpha	Jan 1, 2020	Non Equity Assistance	—	Public
ArangoDB	Oct 6, 2021	Series B	$27.8M	IRIS
ArangoDB	Mar 14, 2019	Series A	$10M	Bow Capital
ArangoDB	Jun 29, 2017	Seed Round	€4.2M	Target Partners
ArangoDB	Nov 24, 2016	Seed Round	€2.2M	—
ArangoDB	Feb 10, 2015	Seed Round	€1.9M	—

（2） 开源软件数据预处理

开源软件数据的预处理涉及数据筛选、异常值处理和特征提取等步骤。

1. 数据筛选

   根据研究目标和需求，从原始数据集中筛选出相关的事件和属性，如项目活动类型、参与者信息等。

2. 异常值处理

   识别并处理数据中的异常值，这些异常值可能是由于数据收集错误或其他外部因素造成的。

3. 特征提取

   从处理后的数据中提取出对研究有意义的特征，经提取整理出了多个特征，包括融资当时时间点涉及的所有开发人员总数；融资当时时间点涉及的所有 Stars 总数；融资当时时间点涉及的所有项目的 openrank 总和。

4. 时间截取

   将所有行为数据截止在融资之前，融资之后的数据会对融资形成因果时间矛盾的影响，所以此处将其删除。

这一阶段的数据预处理工作对于准确评估开源软件项目的特性和影响力至关重要。

（3） 开源软件 README 文本数据预处理

如图3.8所示，由于 readme 在 GitHub 是一个富文本展示方式，所以会出出现很多 HTML、URL、markdown 格式这类的与语义无关的数据。于是，对于开源软件的 README 文本数据，预处理主要包括文本清洗、分词和词性标注等步骤。

1. 文本清洗：移除文本中的 HTML 标签、URL、多余空白、非打印字符等无关内容，转换成小写，提取出干净的文本数据。
2. 分词：将连续的文本内容分割成有意义的词汇单元，为后续的文本分析和特征提取做准备。

3. 词性标注：对分词后的文本进行词性标注，识别词汇的语法属性，如名词、动词等，这对于理解文本语义和进行深入的语言分析非常重要。

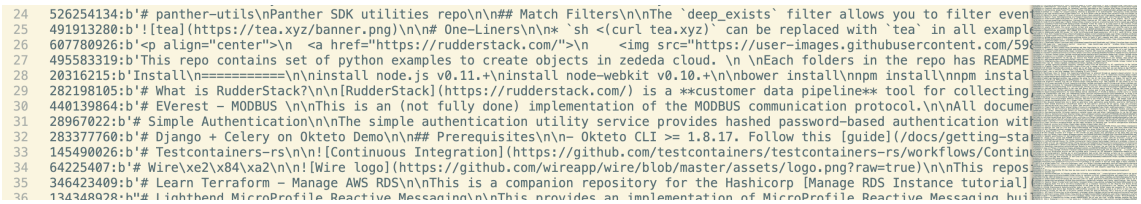

图 3.8 待预处理的 README 文本数据

通过这些细致的预处理步骤，能够有效地从 README 文本数据中提取出有用的信息，为进一步的文本分析和模型训练打下坚实的基础。

3.4　基准数据描述与可视化

如表3.3所示，预处理后的数据集包含 1057 条记录，涵盖了从开源软件中挑选的关键特征。数据集包含 1374 条记录和 10 个列。

表 3.3 预处理后的数据集特征描述

特征名称	数据类型	描述
COSS Company	object	商业开源软件公司的名称。
FOSS Core(s)	object	公司核心开源软件组织/项目。
Stage	object	项目融资阶段。
Round Size	float64	融资金额，以百万美元为单位。
Month Announced	int64	融资宣布的月份。
Year Announced	int64	融资宣布的年份。
Lead Investor	object	主要投资者的名称。
org	object	核心开源软件组织的信息列表。
user	object	核心开源软件组织的用户信息列表。
repo	object	核心开源软件组织的项目信息列表。
OpenRank	float64	OpenRank 在融资对应时间的值。
Star	float64	Star 数量在对应时间的值。
Participants	float64	参与者数量在对应时间的值。
Repositories	int64	核心开源软件组织的项目总数。
repo_readme_feature	object	基于项目 README 文件内容，使用 BERT 模型提取的特征向量。

图3.9展示了基准数据集中的各特征的分布情况。可以看出除了声明年份和生命月份，其余特征都处在一个右偏情况。

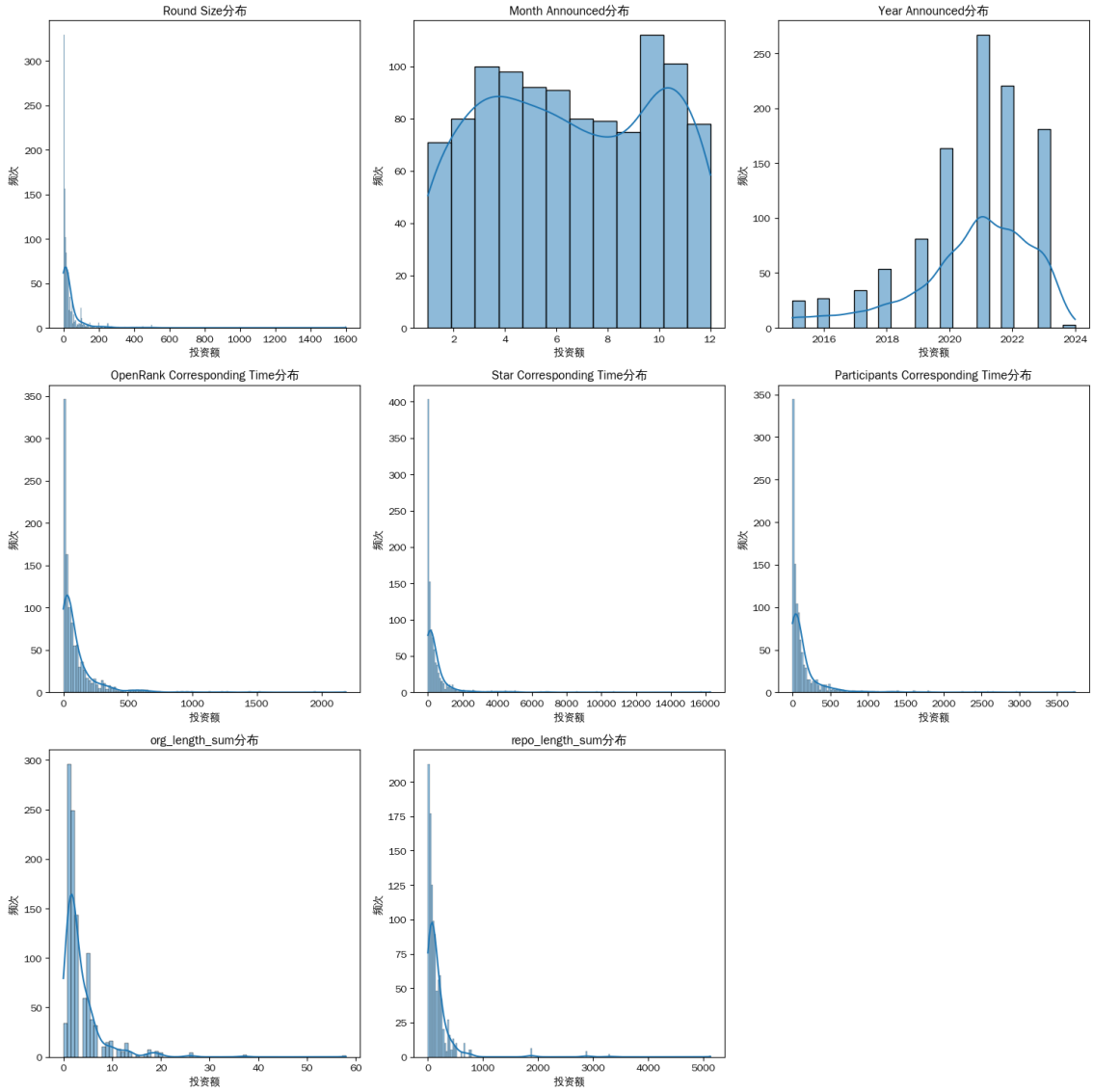

图 3.9 基准数据集各特征分布示意图

1. 融资金额（Round Size）分布

   如图3.9中第一张子图所示，融资金额范围从 0.001 万美元到 1600 万美元，平均融资金额约为 31.61 万美元。其中位数（50% 分位数）融资金额为 10.5 万美元，表明融资金额分布存在较大偏斜，大部分融资金额较低，但也存在少数较高的融资案例。图显示，融资金额主要集中在较低的区间，随着金额增加，频率迅速下降，显示出典型的右偏分布特征。

2. 融资公告年份（Year Announced）分布

   融资公告年份的分布显示了不同年份的融资案例数量。从图3.9中第一行第三张子图中可以看出，某些年份（具体年份未标出）的融资案例数较多，可能反映了那些年份开源软件行业的融资活跃度较高。

3. 融资公告月份（Month Announced）分布

   如图3.9中第一行第二张子图所示，融资公告月份的分布较为均匀，但仍有些月份的融资案例稍多于其他月份。这可能与行业特定事件或投资周期有关。

这些统计和可视化分析提供了对开源软件公司融资活动的基本理解，包括融资规模的分布特征、融资活动在不同年份的分布情况，以及全年各月份的融资活动分布。

3.5　本章小结

本章综述了开源软件的协作网络与仓库文本构建的模型的各数据采集的优化策略，强调了数据准确性与完整性的重要性，并提出了一个全面的数据采集框架。该框架包含了从录屏和 OCR 技术的初步数据采集到通过 GitHub REST API 全面收集开源项目行为数据的过程，同时利用 Wind 和 Crunchbase 等平台获取开源公司的商业数据。此外，通过获取 README 文本数据，为构建仓库文本数据奠定了基础。

为提升数据采集效率和质量，建议采用网络爬虫技术和数据存储管理平台，实施数据验证和清洗流程，利用数据可视化工具和机器学习模型来分析数据，最后，通过数据治理框架和目录确保数据的良好管理和可访问性。

研究涵盖了商业开源公司数据、开源软件数据、开源软件协作网络数据，以及开源软件的 README 文本数据的收集与预处理。商业数据的预处理包括数据清洗、处理缺失值、手动验证和数据归一化，特别是对于跨平台数据需要货币单位统一。开源软件数据预处理包括数据过滤、异常处理、特征提取和时间切片，以确保数据的相关性和一致性。README 文本数据预处理则包括文本清洗，移除不相关的格式数据以准备进一步的分析。

本章的改进策略旨在通过提高数据采集的质量和效率，为开源软件社区的研究与分析提供可靠的数据基础，支持开源生态的商业分析和决策。

第四章　开源软件的协作网络与仓库文本核心特征建模

本章致力于开发并评估一个模型，该模型集成了开源软件的协作网络特征与仓库文本特征，用于更准确地建模和预测开源软件的社区活跃度和影响力。通过整合本文针对已有的 OpenRank 改变的 Dynamic OpenRank 等协作网络特征指标与通过 BERT 处理的 README 等仓库文本特征，并在之后根据不同特征的统计性质，对第三章获取的数据进行特征处理和建模。

4.1　特征选取与描述

4.1.1　模型整体架构

构建的整体过程如图4.1所示，本研究的模型构建依赖于对开源软件的协作网络指标与 README 文件等仓库文本特征的深入理解和分析。OpenRank 作为衡量社区影响力的关键指标，提供了一个量化社区活跃度和参与度等协作网络特征的有效途径，本研究对其进行改进，使其更丰富的反应协作网络特征。此外，通过统计开源软件项目的开发人员总数、Star 总数和项目 OpenRank 总和，进一步丰富了模型的数据基础。特别地，利用 BERT 模型对 README 文件进行深度语义分析，使能够从仓库文本内容中提取出总结仓库文本特征的关键信息。

数据分为文本数据和非文本数据（数字形式数据），非文本数据建模过程如下：

协作网络特征数据在项目仓库层面，在第三章中获取的开源软件行为和网络数据为：项目的 Star 数、参与人员人数、项目的 OpenRank 值，针对这些数据，本研究做如下变换：

- OpenRank：通过图计算对该数据进行重新计算，计算出变体 Dynamic OpenRank（不同类型的开源软件项目的阻尼系数不同）；
- Star 数、参与人员人数：这些数据不变;

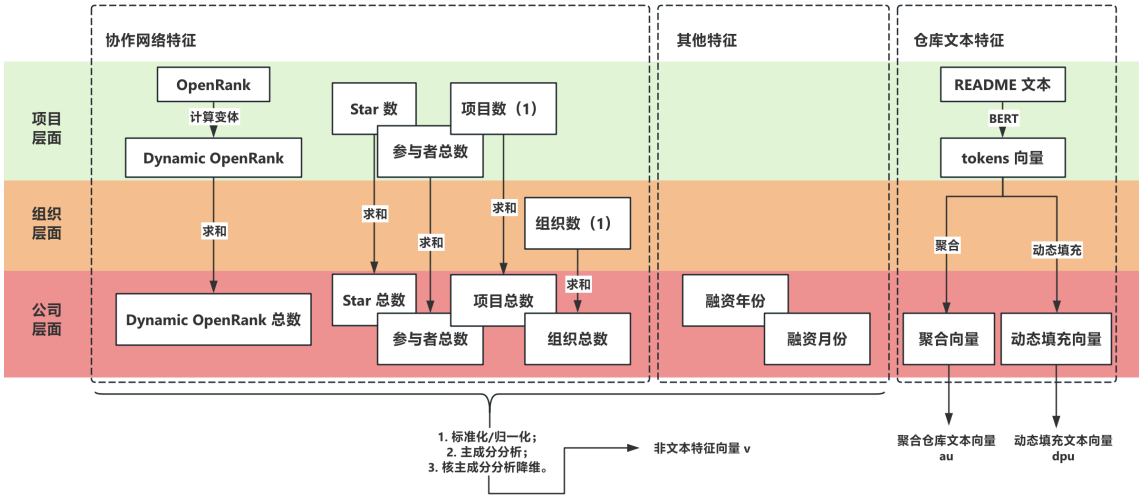

图 4.1 协作网络特征与仓库文本特征数据模型构建过程

- 项目数（1）：尽管 1 个项目仓库可以更名多次，但由于一个项目本身的项目数为 1，故这里记为 1。

在开源软件组织层面，类似的，尽管 1 个组织可以更名多次，但由于一个组织本身的组织数为 1，故这里记为 1；

在开源软件公司层面，将之前的所有特征数值求和即得到公司层面的协作网络特征数据；

其他特征融资时间直接记录下来。

根据以上操作，将会得到一个包含七个属性的列表，将上面所有的数据执行下列操作：

1. 标准化/归一化；
2. 主成分分析；
3. 核主成分分析降维。

最终得到的数据为对上述所有非文本特征进行核主成分分析降维后取重要成分的 26 个属性。

文本数据的建模过程如下：将仓库 README 的内容通过 BERT 的 tokenizer 转换为 token 向量，针对开源软件公司，需要将其所拥有的这些仓库的 token 向量组成一个列表，分别考虑两种对向量列表的处理方法：

1. 聚合：考虑使用小型模型时使用聚合方法确保模型训练有效；
2. 动态填充：考虑较大规模模型使用动态填充方法确保模型训练充分。

4.1.2　数据特征初步筛选与计算

（1）　特征选择

在构建开源软件供应链价值评估模型时，选择具有强预测能力的特征至关重要。这些特征应能够全面反映开源项目的社区活跃度和影响力。基于数据探索和预先的假设，精心挑选了融资时 COSS 组织的参与人员总数、融资时 COSS 组织的各项目 Star 总数、融资时的 Dynamic OpenRank 总和与 README 文件的处理结果总列表四个新特征，并提供了详细的原因说明和分析。

1. **融资时 COSS 组织的参与人员总数**

   开发人员总数直接反映了项目的人力资源规模，是评估项目活跃度和持续维护能力的重要指标。一个拥有众多开发人员的项目通常意味着更高的社区活跃度，更频繁的更新，以及更快的问题解决速度。此外，较大的开发团队也可能吸引更多的新贡献者加入，形成正向循环，进一步增强项目的社区影响力。开发人员总数反映了项目背后的人力资源规模。

   从社会网络分析 (SNA) 的角度看，参与者总数也可视为网络规模的指标，较大的网络规模通常意味着更高的结构洞机会 (structural holes) 和信息流动效率，从而促进了社区的创新和活跃度。此外，根据集体行动理论，人力资源的规模直接关联到社区的集体行动能力，影响项目的持续发展和维护能力。

2. **融资时 COSS 组织的各项目 Star 总数**

GitHub Star 数量被广泛认为是衡量开源项目受欢迎程度和社区认可度的指标。一个项目的 Star 数量越多，通常意味着它拥有更广泛的用户基础和更高的可见度。在融资背景下，Star 总数反映了项目在融资时刻的社区支持和市场潜力，是评估项目吸引力和未来增长潜力的关键数据点。Star 总数作为社区认可度的量化指标，反映了项目的受欢迎程度和用户基础的广泛性。

从信息扩散理论的角度，Star 数量可以被视为信息扩散的" 种子" 数量，更多的 Star 意味着更大的信息扩散潜力。此外，根据社会认同理论，较高的 Star 总数也可能增强潜在贡献者的归属感，进一步促进社区的发展。

3. **融资时的 Dynamic OpenRank 总和**

Dynamic OpenRank 作为一个综合性的社区影响力指标，量化了项目在开源社区中的相对地位和影响力。通过计算融资时刻所有相关项目的 Dynamic OpenRank 总和，能够获得一个关于组织整体社区影响力的综合视图。这不仅反映了项目本身的社区活跃度，还考虑了项目与社区其他成员的互动程度，为评估项目的整体社区影响力提供了一个更全面的指标。

项目的 Dynamic Openrank 总和 (DO_{total}) 综合反映了组织在开源社区中的影响力和地位。其数学表示为：

$$DO_{total} = \sum_{i=1}^{k} DO_i \tag{4.1}$$

其中，k 是组织中项目的总数，DO_i 是第 i 个项目的 Dynamic OpenRank 值。

Dynamic OpenRank 结合了项目的活跃度、参与度和社区反馈等多个维度，从多个角度全面反映了项目的社区影响力。根据网络中心性理论，高 OpenRank 的项目通常处于社区交流的核心位置，能有效促进知识的流动和创新。

4. **README 文件的处理结果总列表**

README 文件是项目对外展示的“门面”，它提供了项目概述、使用说明、贡献指南等关键信息。通过利用 BERT 模型对 README 文件进行深度语义分析，能够从中提取出反映项目特性、社区期望和参与动机等深层次语义信息。这些信息对于理解项目的吸引力、用户友好度以及潜在的社区参与度具有重要意义。处理后的 README 文本特征，结合其他量化指标，能够为模型提供丰富的语境信息，增强其预测开源软件社区活跃度和影响力的能力。

利用自然语言处理技术，尤其是深度学习模型如 BERT 的 [CLS] token 向量，可以从文本中提取高级语义特征，如主题一致性、情感倾向等，这些特征对于理解项目的吸引力、维护水平和社区活跃度具有重要价值。同时从语用学的角度分析 README 文本还可以揭示项目社区的交流模式和文化，进一步增强对社区活跃度和影响力的理解和预测。

本研究将一个 COSS 对应的所有的仓库的 README 内容存储为一个列表，但后对这个大列表进行处理和分析。处理方式有两种：聚合（Aggregate）和动态填充（Dynamic Padding）。以下是这两种方法的详细说明：

5. **聚合（Aggregate）的文本特征向量**

在聚合方法中，将一组 README 文档 $\{D_1, D_2, \ldots, D_n\}$ 聚合成一个单一的表示，然后作为整体进行处理和分析。这种方法的核心优势在于它能够提供宏观的视角，揭示出跨项目的共同主题和趋势，这对于理解组织的总体目标和社区文化具有极大的价值。例如，通过聚合分析，可以识别出整个社区普遍关注的技术问题、共享的价值观，或者在多个项目中重复出现的合作模式。

数学上，聚合表示可以通过计算所有文档 [CLS] 向量的平均值得到：

$$\mathbf{h}_{agg} = \frac{1}{n} \sum_{i=1}^{n} BERT(D_i) \tag{4.2}$$

其中，$\mathbf{h}_{agg}$ 是聚合后的文档表示，代表了整个项目或组织的总体特性。

然而，聚合方法也有其局限性。最显著的是，它可能掩盖了单个项目内部的独特信息和差异。在聚合的大海中，那些只在少数项目中出现的重要细节可

能会丢失，这些细节可能对理解特定项目的社区参与和贡献动态至关重要。因此，虽然聚合方法适用于揭示广泛的趋势和主题，但它需要与能够捕捉项目特定信息的方法相结合，以获得全面的分析视角。

6. **动态填充（Dynamic Padding）的文本特征向量**

与聚合方法不同，动态填充方法针对每个文档独立处理，并根据其内容动态调整输入模型的长度，以确保模型能够有效捕捉每个文本的细节。这种方法的显著优势在于其对项目特异性的敏感性。通过保留每个项目的独立信息，动态填充方法能够揭示那些可能在聚合过程中被忽略的独特模式和趋势，如特定项目的技术创新、社区互动风格或贡献者行为模式。对于每个文档 D_i，首先将其转换为 BERT 模型的输入格式，并计算其 [CLS] 标记的特征向量 $\mathbf{h}_{[CLS],i}$。然后可以通过一个可学习的权重矩阵 $\mathbf{W}$ 将这些特征向量映射到一个共同的特征空间：

$$\mathbf{h}_{dp,i} = \mathbf{W} \cdot \mathbf{h}_{[CLS],i} \tag{4.3}$$

其中，$\mathbf{h}_{dp,i}$ 是经过动态填充处理的文档 D_i 的表示。这种方法允许模型捕获每个项目的独特性，同时通过权重矩阵 $\mathbf{W}$ 实现特征空间的统一。

然而，动态填充方法也面临着挑战，尤其是在处理大量项目时的计算资源需求更高，且在整合和比较不同项目的分析结果时可能需要更复杂的方法。此外，保持每个项目分析的独立性可能导致在寻找跨项目共性时出现困难。

（2） 特征计算

融资时 COSS 组织参与人员总数 融资时 COSS 组织参与人员总数的计算采用算法1进行。算法输入包括组织的所有仓库（$repositories$）以及融资事件的时间点（$fundingTime$），输出为在融资时间点之前的参与人员总数。该算法首先初始化参与人员总数为 0，然后遍历组织下的每一个仓库。对于每个仓库，通过 CountParticipants 函数计算在融资时间点之前的开发人员数量，并累加到参与人员

总数中。最终返回的参与人员总数为在融资时间点之前，该组织所有项目的参与人员累计数。

Algorithm 1 Calculate Participants for Open Source Software Value Assessment

Input: $repositories$, $fundingTime$

Output: Features calculated considering data up to $fundingTime$

1: **Function** CalculateParticipantsTotal($repositories$, $fundingTime$)
2: $participantsTotal \leftarrow 0$
3: **for** each $repository$ in $repositories$ **do**
4: $participantsTotal \leftarrow participantsTotal+$ CountParticipants($repository$, $fundingTime$)
5: **end for**
6: **return** $participantsTotal$
7: **End Function**

此外，此算法的复杂度主要取决于仓库的数量以及每个仓库中参与者的统计效率，适当的优化和并行处理可以有效提高其执行效率。

融资时 COSS 组织各项目 Star 总数 融资时 COSS 组织参与人员总数的计算采用算法2进行。算法输入包括组织的所有仓库（$repositories$）以及融资事件的时间点（$fundingTime$），输出为在融资时间点之前的 Star 总数。该算法首先初始化参与人员总数为 0，然后遍历组织下的每一个仓库。对于每个仓库，GetStarsCountAtTime 函数计算在融资时间点之前的开发人员数量，并累加到 Star 总数中。最终返回的 Star 数为在融资时间点之前，该组织所有项目的 Star 累计数。

Algorithm 2 Calculate Stars for Open Source Software Value Assessment

Input: $repositories$, $fundingTime$

Output: Features calculated considering data up to $fundingTime$

1: **Function** CalculateStarsTotalAtFunding($projects$, $fundingTime$)
2: $starsTotal \leftarrow 0$
3: **for** each $repository$ in $repositories$ **do**
4: $starsTotal \leftarrow starsTotal+$ GetStarsCountAtTime($repository$, $fundingTime$)
5: **end for**
6: **return** $starsTotal$
7: **End Function**

此外，此算法的复杂度主要取决于仓库的数量以及每个仓库中参与者的统计效率，适当的优化和并行处理可以有效提高其执行效率。

融资时 COSS 组织各项目 Dynamic OpenRank 总和 本算法3通过遍历每个仓库，并调用 FetchDynamicOpenRank 函数来累加在融资时刻之前的 Dynamic OpenRank 值。通过这种方法能够得到一个准确的 Dynamic OpenRank 总和，为评估项目的社区影响力和潜在价值提供了基础。

Algorithm 3 Calculate Dynamic OpenRank for Open Source Software Value Assessment

Input: $repositories$, $fundingTime$

Output: Features calculated considering data up to $fundingTime$

1: **Function** CalculateDynamicOpenRankTotalAtFunding($repositories$, $fundingTime$)
2: $openRankTotal \leftarrow 0$
3: **for** each $repository$ in $repositories$ **do**
4: $openRankTotal \leftarrow openRankTotal+$ GetOpenRankAtTime($repository$, $fundingTime$)
5: **end for**
6: **return** $openRankTotal$
7: **End Function**

Dynamic OpenRank 总和的计算基于一个核心假设，即 Dynamic OpenRank 能够全面反映一个项目在开源社区中的影响力。这一假设背后是对 Dynamic OpenRank 计算方法的信任，该方法通常结合了项目的活跃度、用户参与度以及其他关键指标。

算法的复杂度取决于处理的仓库数量和获取 Dynamic OpenRank 数据的效率。为了提高算法的执行速度和准确性，由于 Dynamic OpenRank 具体值是事先计算并存储于 ClickHouse 的成果，所以这里直接查询并抓取相应的数据即可。

README 文件的处理结果总列表 本算法4首先遍历开源项目的仓库列表，针对每个项目的 README 文件，执行以下步骤：

1. **文本提取** 基于融资时间点，提取该时刻之前的最新 README 文本。这一步骤确保分析的内容是投资决策时可获得的信息。

2. **BERT 处理** 将提取出的 README 文本输入 BERT 模型，获取深度语义特征向量。BERT 模型的双向编码能力使得从文本中提取的特征能够全面反映其语义内容。

3. **特征集成** 根据实际应用场景，选择适当的方法（聚合或动态填充）来集成各个项目的 README 特征向量，形成一个综合的特征列表。

在应用此算法时，需要注意的是，BERT 模型处理大量文本数据时的计算资源消耗。为了提高效率，本文使用 GPU 加速计算。

Algorithm 4 Process README with BERT for Open Source Software Value Assessment

Input: $repositories$, $fundingTime$

Output: Features calculated considering data up to $fundingTime$

Function ProcessREADMEWithBERT($repositories$, $fundingTime$)
$readmeFeaturesList \leftarrow []$
for each $repository$ in $repositories$ **do**
    $readmeText \leftarrow$ FetchREADMEText($repository$, $fundingTime$)
    $featureVector \leftarrow$ BERTModelProcess($readmeText$)
    Append $featureVector$ to $readmeFeaturesList$
end for
Choose between Aggregate or Dynamic Padding for $readmeFeaturesList$
return $readmeFeaturesList$
End Function

聚合（Aggregate）的文本特征向量 本算法5通过计算所有项目 README 特征向量的算术平均值，形成一个代表整个社区或组织的聚合特征向量。

动态填充（Dynamic Padding）的文本特征向量 本算法6通过填充或截断操作，将所有 readme [cls] 标记的内容链接的特征向量调整至统一的长度 $maxLen$，这样不仅保证了模型输入的一致性，也保留了每个 README 文件的独特信息。动态填充方法的一个关键优势是它能够适应不同长度的文本数据，使得模型训练和推理更加灵活有效。

然而，选择合适的 $maxLen$ 值是一个挑战，需要在保留足够信息和计算效率之间做出平衡。过长的 $maxLen$ 可能会导致计算资源的浪费，而过短则可能丢失

Algorithm 5 Aggregate README Features

Input: $readmeFeaturesList$

Output: Aggregated feature vector

Function AggregateFeatures($readmeFeaturesList$)
$aggregatedFeature \leftarrow$ Initialize to zero vector of appropriate size
$count \leftarrow 0$
for each $featureVector$ in $readmeFeaturesList$ **do**
    $aggregatedFeature \leftarrow aggregatedFeature + featureVector$
    $count \leftarrow count + 1$
end for
if $count > 0$ **then**
    $aggregatedFeature \leftarrow aggregatedFeature/count$
end if
return $aggregatedFeature$
End Function

重要信息。因此，根据数据集的具体特征和模型要求，合理设定 $maxLen$ 是动态填充方法成功应用的关键。

4.2　数据核心特征建模

4.2.1　数据特征空间映射

开源项目的复杂性，包括其技术实现、社区协作模式、以及商业发展策略，构成了对其进行有效评估的主要挑战。特别是，这些项目产生的数据通常是高维的、非线性的。且不谈文本信息，单指其中的非文本信息，便有高维和低维的差距且难以通过线形方式调和。这些特点使得传统的线性数据处理和分析方法难以适用。

（1）　开源软件项目的高维非线性数据与低维线形数据的整合问题

此处旨在解决开源软件项目评估中遇到的一个关键问题：如何有效整合和分析开源软件项目产生的高维非线性特征数据与低维线性特征数据，并探索这两类数据之间的相互作用及其对项目价值和社区影响力评估的影响。

Algorithm 6 Dynamic Padding of README Features

Input: $readmeFeaturesList$, $maxLen$

Output: Padded feature list

Function DynamicPadding($readmeFeaturesList$, $maxLen$)
$paddedFeaturesList \leftarrow []$
for each $featureVector$ in $readmeFeaturesList$ **do**
  if length($featureVector$) < $maxLen$ **then**
    $paddedVector \leftarrow$ Pad $featureVector$ to $maxLen$ with zeros
  else
    $paddedVector \leftarrow$ Truncate $featureVector$ to $maxLen$
  end if
  Append $paddedVector$ to $paddedFeaturesList$
end for
return $paddedFeaturesList$
End Function

在开源软件项目的数据集中，协作网络特征、仓库文本特征等高维非线性数据，以及一些基础的统计数据如参与者数量、Star 数量等低维线性数据，共同构成了项目的全貌。这些数据不仅在维度和性质上存在显著差异，而且它们之间可能存在复杂的相互作用和因果关系，这些作用和关系对于项目的总体评估至关重要。

将问题拆分开来，可以得到两个原子问题：

1. **数据整合的挑战** 如何设计一个有效的方法来整合高维非线性数据和低维线性数据，以便于后续的处理和分析？
2. **相互作用分析的挑战** 如何识别和量化这些不同类型数据之间的相互作用及其对开源软件项目价值和社区活跃度评估的影响？

（2） 基于核主成分分析的特征空间映射方法

为解决上述问题，本研究提出了一个统一的数据处理框架，该框架首先对数据集进行正态性检验，以确定适用的数据分析方法。检验结果如果是符合正态分布的，则采用多因子分析技术；而如果是非正态分布的数据，则采用非参数重要性

检验来评估其各特征的影响力。如果某特征影响力较低，则判定为不明显特征，将该特征删去。

随后，利用主成分分析来处理低维线性数据，以减少数据维度并提取主要变异来源。对于高维非线性数据，采用核主成分分析，通过非线性映射剩下的主要特征，揭示数据的深层结构，并在累积重要性达到一定阈值（一般为 90%）后认定特征空间已被有效表达，其余部分将被删去。核主成分分析的应用是本研究的核心，它不仅能够有效整合并分析复杂的开源软件项目数据，还揭示了高维非线性特征与低维线性特征之间的相互作用和潜在因果关系。

正态性检验　数据的正态性决定其是否适用于最基础的统计和预处理方法，所以这里首先通过 Shapiro-Wilk 检验确认数据是否符合正态分布。

Shapiro-Wilk 检验是一种统计测试，用来判断一个样本集是否来自正态分布。这个检验由 Samuel Shapiro 和 Martin Wilk 在 1965 年提出，并且通常被认为是检验正态性的最有力的方法之一，特别是对于小样本数据（通常 n < 50）。

Shapiro-Wilk 检验的假设检验可以描述如下：

- **零假设（H0）**样本数据来自正态分布。
- **备择假设（H1）**样本数据不来自正态分布。

检验统计量 (W) 的计算公式如下：

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{4.4}$$

其中: - $x_{(i)}$ 是第 i 个最小值，即第 i 小的样本值。- a_i 是系数，由样本的顺序统计量和正态分布的期望值的协方差矩阵决定。- $\bar{x}$ 是样本的平均值。

计算得出的 W 值是一个介于 0 和 1 之间的值。W 值接近 1 通常意味着数据与正态分布匹配得较好。然而，是否拒绝零假设并不仅仅取决于 W 值，还需要查看与该 W 值相对应的 P 值。

P 值是在零假设为真的前提下，观察到的样本或更极端的样本出现的概率。如果 P 值很小（通常小于显著性水平 α，如 0.05），则拒绝零假设，认为样本不是来自正态分布。

非参方法重要性测试 由于之前得出结论数据各属性均不符合正态分布且通过变换也均无法符合，故此处无法适用单因素方差分析（ANOVA），只能使用非参方法对属性重要性进行评估。

Kruskal-Wallis 检验是一种非参数的方法，用于判断三个或三个以上独立样本的总体中位数是否存在差异。它是单因素方差分析的非参数替代方法，适用于数据不满足正态分布假设的情况。Kruskal-Wallis 检验分为以下步骤：

1. **排名**首先将所有群组的观测值合并在一起，按照值的大小给予排名（从 1 开始，相同数据取平均排名）。
2. **计算统计量** Kruskal-Wallis 统计量 H 通过下面的公式计算：

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{g} \frac{R_i^2}{n_i} - 3(N+1) \tag{4.5}$$

其中，N 是所有群组中观测值的总数，g 是群组的数量，R_i 是第 i 个群组的排名和，n_i 是第 i 个群组中的观测值数量。

置换检验（Permutation Test）是一种非参数统计测试，用于检验两个或多个样本之间是否存在差异。它通过重复随机重新分配样本标签来生成参照分布，并计算原始数据下的统计量在该分布中的位置。其过程如下：

1. **原始统计量**首先计算原始数据下的统计量 T_{obs}，本研究此处使用随机森林。
2. **置换**通过随机打乱样本标签并重新计算统计量 T_{perm} 来生成置换分布。
3. **计算 P 值**统计在所有置换中，有多少次 T_{perm} 至少和 T_{obs} 一样极端，以此来计算 P 值。

随机森林的特征重要性是衡量每个特征对模型预测性能的贡献大小。特征重要性的计算公式如下：

$$\text{Importance}(j) = \frac{1}{N_{\text{trees}}} \sum_{t=1}^{N_{\text{trees}}} \Delta i(s_t, j) \tag{4.6}$$

其中，N_{trees} 是森林中树的数量，$\Delta i(s_t, j)$ 是在第 t 棵树中，通过特征 j 进行分裂导致的不纯度减少量。

低维特征重要性分析 主成分分析（Principal component analysis, PCA）是一种多变量统计方法，其目的是将一个由多个相关变量组成的数据集转化为一组线性无关变量，即主成分。主成分分析通过提取数据的主要变异来源，在丢失最少信息的前提下简化数据的复杂性。

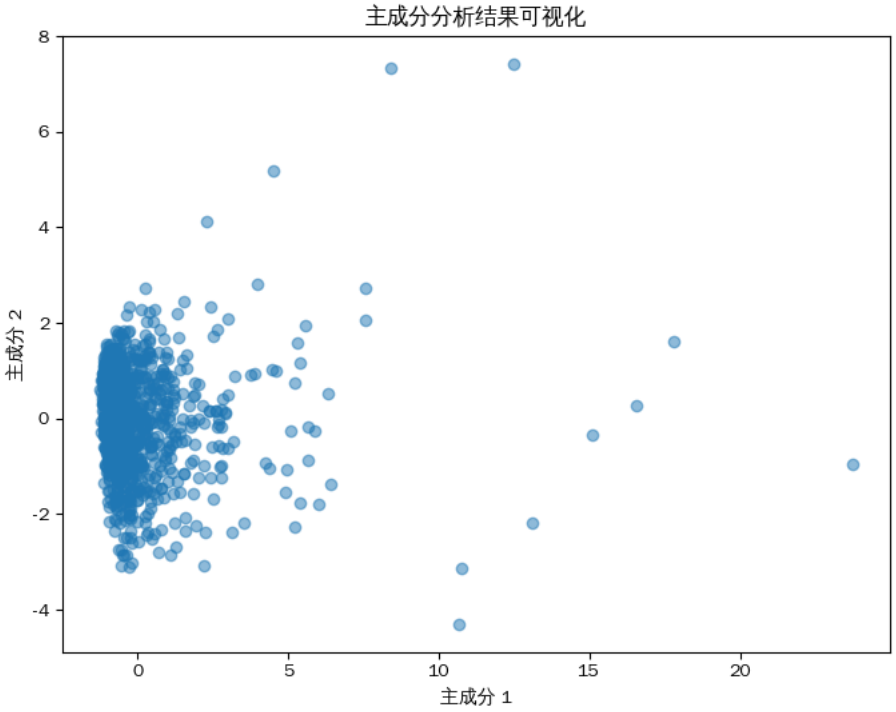

图 4.2 主成分分析图

特征空间映射 核主成分分析（Kernel Principal component analysis, PCA）是传统主成分分析的一种扩展，它使用核技巧来有效地找到在高维特征空间中的主成分，而不需要显式地计算这个高维空间中的点。核主成分分析特别适用于非线性数据集。

核主成分分析的基础是核方法，其核心思想是通过一个非线性映射 ϕ 将原始数据 $\mathbf{x}$ 映射到一个高维特征空间 F：

$$\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \tag{4.7}$$

在这个特征空间中，可以计算数据点的内积，但直接计算通常是计算代价很高的。核技巧通过定义核函数 k 来避免这个问题，核函数直接计算原始空间中任意两点 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 在映射后的内积：

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{4.8}$$

由于数据中存在复杂的非线性关系，为了能够有效地将这些非线性特征映射到一个更高维的空间以更容易区分数据点，本研究使用径向基函数核（Radial Basis Function Kernel or Gaussian Kernel）：$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$ 在计算了核矩阵 K 后，核主成分分析的目标是找到特征空间 F 中数据的主成分。这可以通过对中心化后的核矩阵进行特征分解来实现。

隔离森林（Isolation Forest） 针对核主成分分析的降维结果，为了查看在更高维度的特征空间中是否存在离群值，本研究使用隔离森林模型对数据点进行筛查。

隔离森林是一种基于集成学习的异常检测算法。它的主要优势是计算效率高，特别适合处理高维数据集，并且不需要先验知识关于数据的分布。其核心思想是异常点通常是稀少的并且在特征空间中的行为与正常点显著不同，因此异常点相对容易被“隔离”。

隔离森林由以下几个关键步骤组成：

1. **构建隔离树（iTree）**从原始数据集中随机抽取一个样本子集，然后随机选择一个特征，并在该特征上随机选择一个切分值来分割数据，生成两个子节点。这个切分过程递归进行，直到满足某个停止条件，如子集中只剩下一个点，或者达到预设的树的深度。

2. **路径长度**路径长度表示从树的根节点到隔离某个点的终端节点的路径边数。异常点通常有更短的路径长度，因为它们在较少的切分中就能被隔离。
3. **构建森林**重复构建多个隔离树，形成隔离森林。每棵树都是独立构建的，并且都使用不同的随机样本子集。
4. **异常分数（Anomaly Score）**异常分数是基于点在所有隔离树上的平均路径长度计算的。分数越接近 1，表示点越可能是异常的。
5. **标记异常值**对于给定的数据点，如果它的异常分数超过了一个预先定义的阈值（本研究使用 0.5），它就可以被标记为异常值。

4.3　建模结果与分析

4.3.1　实验结果分析

（1）　正态性检验

最终的检验结果如图4.3所示，对于所有变量，数据点都在某种程度上偏离了这条直线，这表明数据并不遵循正态分布。特别是在两端，这种偏离更加显著，这表明极端值的存在。

表 4.1 Shapiro-Wilk 正态检验结果

Variable	Test Statistic	P-Value
Round Size	0.315	2.20×10^{-52}
Month Announced	0.939	2.36×10^{-20}
Year Announced	0.894	2.60×10^{-26}
Dynamic OpenRank	0.523	1.31×10^{-46}
Stars	0.374	6.43×10^{-51}
Participants	0.454	9.68×10^{-49}
Organizations	0.615	2.33×10^{-43}
Repositories	0.384	1.18×10^{-50}

在对数据集中的八个变量进行 Shapiro-Wilk 正态性检验后，结果表明所有的样本分布均不符合正态分布。如表 4.1 所示，所有变量的 P 值均小于 0.05，强烈拒

绝了正态分布的假设。如图4.3所示，其 QQ 图（Quantile-Quantile 图）也展现出数据点均在其右侧发生非线性偏移，也即高值区域呈现偏态。

针对数据进行对数变换（如图4.4所示）、平方根变换（如图4.5所示）、Box-cox 变换（如图4.6所示），发现变换后的结果依然均不符合正态分布。

这一结果，使本研究在之后需要采用非参数检验来分析数据。

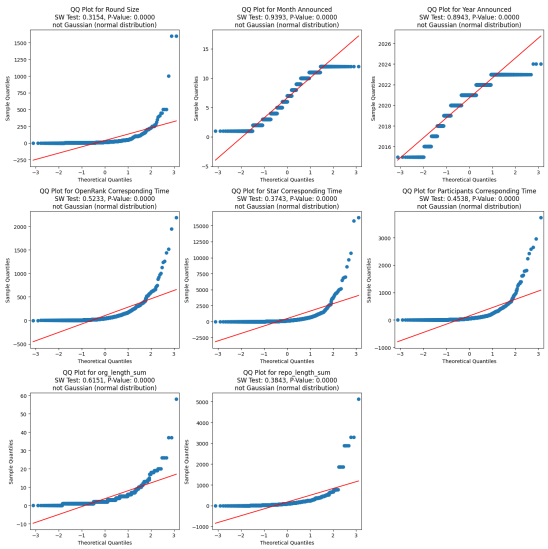

图 4.3 数据各属性 QQ 图

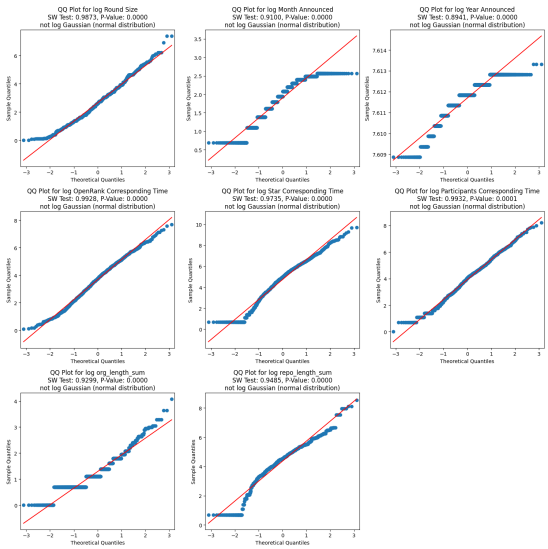

图 4.4 log 变换后数据各属性 QQ 图

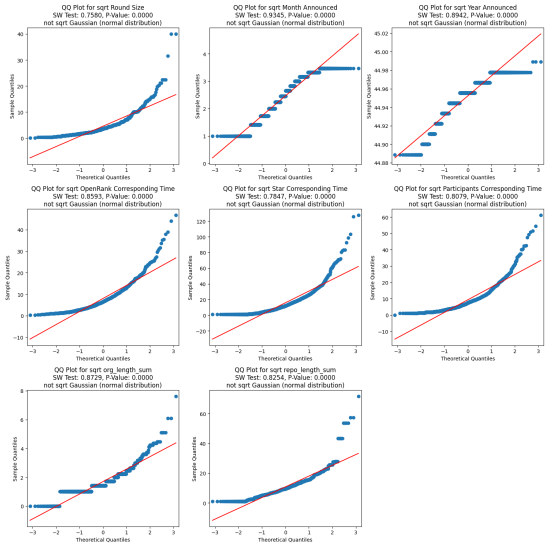

图 4.5 sqrt 变换后数据各属性 QQ 图

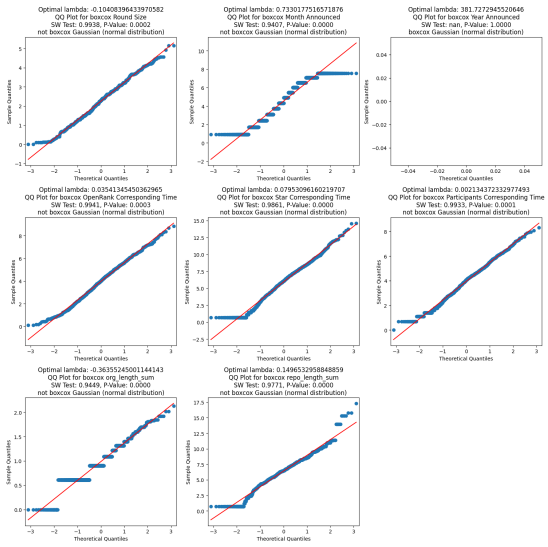

图 4.6 Box-cox 变换后数据各属性 QQ 图

（2） 非参统计结果

如表4.2所示，'Year Announced'、“参与者数量”（Participants）、“组织数”（Organizations）和 “仓库数”（Repositories）是影响因变量的显著特征。特别是 “组织数”（Organizations）和 “仓库数”（Repositories）在 Kruskal-Wallis 检验中显示了非常显著的差异，并且在特征重要性评分中也表现出较高的值，这表明这些特征与因变量之间可能存在强关联。

此外，尽管 “OpenRank 总数”（OpenRank）和 “Star 数”（Star）在 Kruskal-Wallis 检验中没有显示出统计学上的显著差异，但它们在随机森林特征重要性评分中排名较高，可能是因为它们在随机森林模型中的预测表现较好。

另一方面，“公告月份”（Month Announced）在 Kruskal-Wallis 检验中没有显示出显著差异，并且在特征重要性评分中也相对较低，这表明它可能对因变量的影响较小。

于是针对数据各属性，本研究可选择使用非线性模型或基于径向基函数核的核主成分分析处理后的数据进行线性模型。由于核主成分分析的过程会损失信息，本研究更倾向使用非线性模型进行预测。

（3） 低维特征重要性分析结果

如图4.2所示，散点图将数据可视化为由第一和第二主成分（PC1 和 PC2）定义的降维空间。观测点主要沿第一主成分轴分布，第一主成分解释的方差明显大于第二主成分，第一主成分解释了 37.763% 的方差，而第二主成分解释了 13.309% 的方差。聚集在第一主成分左侧的点表明这些数据点可能有共同点或模式，这些是第一主成分有效捕捉到的。尽管不太显著，但沿着第二主成分的分布表明此组分捕获了额外的有价值的方差。

最终的结果如表4.3所示，主成分的特征向量表示在多维空间中最大化投影数据方差的方向。这些向量的元素是相应特征对于每个主成分的权重。在这个案例

表 4.2 Kruskal-Wallis 检验、置换检验及随机森林特征重要性统计结果

Feature	Kruskal-Wallis Statistic	P-Value
Month Announced	17.19	0.102
Year Announced	39.38	9.84e-06
OpenRank	1018.86	0.332
Star	559.16	0.256
Participants	407.68	0.003
Organizations	92.91	1.09e-10
Repositories	406.12	6.59e-14
Permutation Test:		Importance ± Std
Year Announced		3.80% ± 1.40%
OpenRank		3.10% ± 1.00%
Repositories		3.10% ± 0.60%
Organizations		1.10% ± 0.30%
Feature Importance from Random Forest		
OpenRank		20.85%
Repositories		19.27%
Star		19.14%
Year Announced		11.99%
Participants		11.35%
Month Announced		10.86%
Organizations		06.54%

表 4.3 主成分特征列表

Feature	PC1_Weight	PC2_Weight
Participants	0.5100	-
OpenRank	0.4938	-
Repositories	0.4486	0.2054
Organizations	0.4162	0.2301
Star	0.3399	0.3694
Year Announced	-	0.7398
Month Announced	-	0.4383
Total Variance	37.763%	13.309%

中，本研究有两个对应于第一主成分和第二主成分的向量。每个元素的绝对值表示该特征在主成分中的重要性，绝对值越高意味着重要性越大。

这个主成分分析表明数据集中最显著的变化模式与“参与者数量”（Participants）、“OpenRank 总数”（OpenRank）、“仓库数”（Repositories）、“组织数”（Organizations）和“Star 数”（Star）有关。同时，像“公告年份”（Year Announced）和“公告月份”（Month Announced）等时间特征也很重要，表明与时间相关的模式在数据中扮演着关键角色，可能指示数据的周期性。

鉴于每一个属性都很重要，本研究在这里便不再删去任何属性。

需要注意的是，由于之前计算得出这些属性的分布均不符合正态分布，尽管主成分分析可以在数据不是完全正态分布时使用，但如果数据的分布严重偏斜，主成分分析结果可能不稳定或不可解释。于是这里本研究补充性的使用核主成分分析进行进一步分析。

（4） 特征空间映射结果

图4.7显示了使用核主成分分析降维后的数据点在第一主成分和第二主成分构成的二维空间中的分布，该方法使用径向基函数核揭示数据的非线性关系。可以发现数据点分布在图中表现出一定的流形结构，这表明核主成分分析成功地捕捉到了数据的内在特征，并在新的特征空间中展现出了数据的非线性结构。

值得注意的是，数据点在两个主成分上的分布呈现出一种流形结构。这表明核主成分分析成功地捕捉到了数据的非线性关系。对该降维后的数据集进行聚类（K均值算法），得出如图4.8所示结果。其中每种颜色代表一个由聚类算法在降维空间中识别出的簇。簇的空间分离显示了核主成分分析在该数据集上的有效性。

图4.10显示了核主成分分析降维后的累计解释方差，作为分析工具用于确定通过核主成分分析降维后保留的信息量。可以发现曲线在前几项便渐近的趋近于 1，表明前几项特征解释了大部分的数据总方差。为了在之后的研究中简化模型，这里判断在累积解释方差达到 90% 时曲线开始平缓的点最适合作为一个合适的截止

点，在此之后的额外成分对方差的贡献极小，因此带来的信息回报递减。经过计算得出该点的索引为 26，也即数据集经过核主成分分析降维后的数据在第 26 个属性之后的属性将被删去不在用于之后的计算当中。

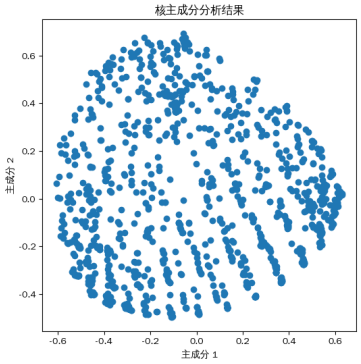

图 4.7 核主成分分析降维后数据点分布

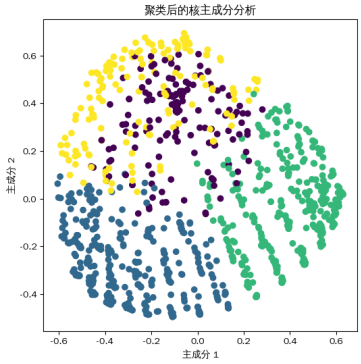

图 4.8 核主成分分析降维后数据的聚类情况

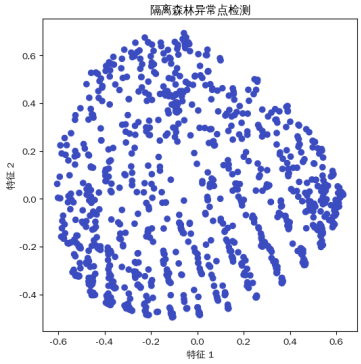

图 4.9 核主成分分析降维后数据的异常点检测

（5） 聚类质量评估

对核 PCA 降维处理后的数据集进行聚类质量评估，由于其展现的形态是流形状态，不能直接使用聚类算法进行比较，所以经过 t-SNE 处理后采用了轮廓系数和戴维斯-布尔丁指数作为评估指标。轮廓系数测量了聚类的凝聚度和分离度，值越接近 1 表示聚类质量越高。戴维斯-布尔丁指数则衡量了聚类的分离度，值越小表示聚类之间的区分度越明显。

在数学和统计学领域，聚类质量指标和降维技术是理解和解释高维数据集结构的关键工具。特别地，轮廓系数、戴维斯-布尔丁指数以及 t-分布随机邻域嵌入（t-SNE）方法在数据科学和机器学习领域内被广泛应用于评估聚类性能和可视化高维数据。

轮廓系数是一种评价聚类性能的度量，其值的范围在-1 到 1 之间。对于任意给定的样本，轮廓系数是基于样本与其同类别中其他样本的平均距离（凝聚度）和样本与最近的不同类别样本的平均距离（分离度）计算得出的。数学上，给定样本 i，其轮廓系数 $s(i)$ 定义为：

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4.9}$$

其中，$a(i)$ 是样本 i 到同一聚类中其他样本的平均距离（即凝聚度），$b(i)$ 是样本 i 到最近一个聚类中所有样本的平均距离（即分离度）。轮廓系数接近 1 表示样本很好地匹配到其自身的聚类同时不匹配到邻近的聚类，值接近-1 表示样本更适合分配给邻近的聚类。

戴维斯-布尔丁指数是另一种常用于评估聚类算法性能的指标。该指数基于聚类内部的紧密性和聚类之间的分离度，其值越小，表示聚类效果越好。对于 k 个聚类，戴维斯-布尔丁指数 DB 定义为：

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \tag{4.10}$$

其中，c_i 和 c_j 分别是聚类 i 和 j 的中心，σ_i 和 σ_j 是聚类内部样本到各自聚类中心的平均距离，$d(c_i, c_j)$ 是聚类中心之间的距离。这个比值的最大值被视为聚类 i 的得分，而 DB 指数是所有聚类得分的平均值。

t-SNE 是一种高效的非线性降维技术，特别适用于高维数据的可视化。t-SNE 通过优化低维空间中的数据点位置，使得原高维空间中相近的数据点在低维空间中也相近，而远离的点在低维空间也远离。具体而言，t-SNE 首先在高维空间中计算数据点间的相似度，转化为条件概率表示数据点间的邻近程度。然后，在低维空间中对每对数据点执行类似的计算。

表4.4展示了原始数据与经过核主成分分析处理并利用 t-SNE 降维后数据在聚类质量指标上的对比。从表中可以看出，经过核主成分分析处理的数据在轮廓系数上从 0.2737 提升到了 0.3895，表明聚类内部的凝聚度增加，聚类之间的分离度也得到了改善。同样，戴维斯-布尔丁指数从 1.1360 降低到 0.8080。这一结果充分证明了核主成分分析降维后数据的聚类结构更加明显，聚类间的分离度得到了显著提升，从而指示出核主成分分析能够有效地揭示数据的内在非线性结构并增强其聚类性能。

以上聚类质量评估的结果进一步强调了核主成分分析作为一种有效的非线性降维方法，在处理复杂数据集时的重要价值。通过揭示数据的非线性关系和内在

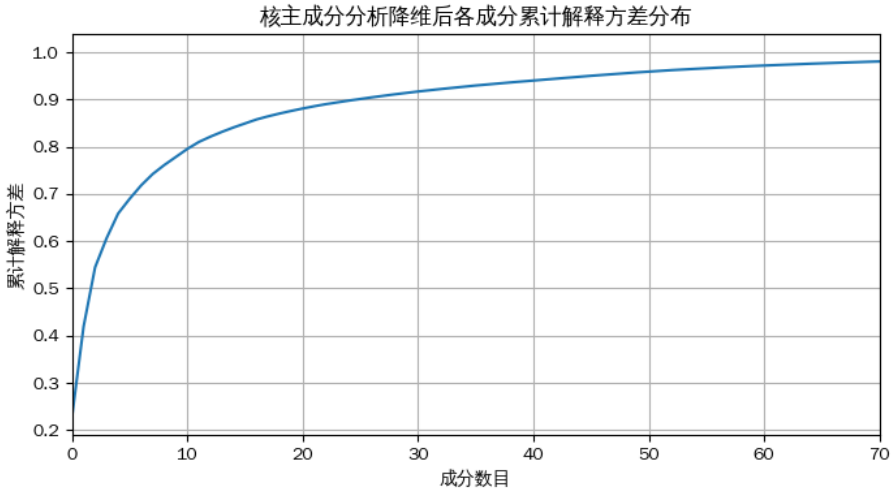

图 4.10 核主成分分析降维后各成分累计解释方差分布

结构，核主成分分析不仅提高了数据的可解释性，也为后续的聚类分析和模式识别等任务提供了更加准确的数据基础。

综上所述，本研究将使用核主成分分析降维并删去不重要属性后的数据作为非文本格式的输入，并更偏向使用非线形结构对数据集进行后续的分析和训练。

表 4.4 原始数据与核主成分分析处理后数据的聚类质量指标对比

Data Type	Silhouette Coefficient	Davies-Bouldin Index
Original Data	0.2737	1.1360
Kernel PCA + t-SNE	0.3895	0.8080

（6） 隔离森林结果

本研究使用隔离森林识别核主成分分析降维后的离群点，如图4.9所示，可以发现数据中高维度情况下没有计算出离群点，并不对核主成分分析降维后的流形形状造成影响，这说明核主成分分析的特征空间下对数据有较好的区分效果。

4.4　本章小结

第四章深入探讨了预测开源软件社区活动水平和影响力的关键因素。通过综合分析从 GitHub 等平台收集的大量数据，并利用自然语言处理技术解读文本信息，明确了影响开源项目活跃度和影响力的核心特征。

特征工程过程揭示了四个关键特征：参与者数量、Stars 数量、OpenRank 得分以及 README 文件内容参与者数量作为项目活动水平的直观指标，反映了项目的参与度。星标数量则提供了一个项目在社区中认可度和受欢迎程度的量化指标。OpenRank 得分综合了项目活跃度的多个方面，包括提交次数、拉取请求和问题数量，为项目的综合影响力提供了量化的评估。README 文件内容的分析则从另一个维度提供了项目吸引社区参与和维持活跃度的文本证据。

结合在前文中分析的统计数据和图形，尤其是 QQ 图和 Shapiro-Wilk 检验的结果，可以发现数据分布呈非正态性，这可能包括数据转换或应用非参数统计方法，以适应模型对数据正态性的假设。根据这一结论，

通过采用核主成分分析结合 t-分布随机邻域嵌入（t-SNE），本研究实现了轮廓系数从 0.2737 提升至 0.3895，戴维斯-布尔丁指数从 1.1360 降低至 0.8080，量化数据明确表明了聚类质量的显著提升。这一结果突出了核 PCA 降维后的数据比原始数据更适合作为模型的输入值。

基于上述发现，本研究将开发一个机器学习模型，利用这些核心特征来预测社区活动水平和影响力。该模型的目标是提供全面的洞见，关于推动社区参与和开源项目成功的动因。通过对影响开源软件项目成败的因素进行细致的分析，不仅能够更好地理解这些因素，还能够为开源项目的建设者和参与者提供实用的指导，帮助他们增强项目的活跃度和社区影响力。

第五章　基于 DCNN-ETNN 的开源软件价值评估方法

本章详细介绍了基于深度学习的开源软件价值评估方法的开发过程，其中着重探讨了六种不同的算法设计。这些算法旨在综合利用开源软件的协作网络特征和仓库文本特征，以提供对软件价值的深入评估。

本章最终形成一种结合了深度卷积神经网络和增强 Transformer 层的深度神经网络模型（DCNN-ETNN），此方法在第三章形成的数据集上最终的表现优于其他模型。本研究以此模型为基础构建了一个开源软件价值评估方法。

5.1　问题描述与定义

5.1.1　评估方法概述

针对回归型任务有比较常用的算法，本文构建了六种算法进行比较和概述，分别包括：

1. **线性回归（Linear Regression）**线性回归不包含文本 BERT 数据，主要依赖于传统的数值型和分类型特征。这种方法有助于评估在不考虑复杂文本信息时，其他特征对软件价值评估的贡献程度。
2. **随机森林（Random Forest）**同样采用文本 BERT 数据的聚合（Aggregate）处理方式，随机森林通过构建多个决策树并进行集成学习，提高模型的泛化能力。该方法在处理大规模数据集时表现良好，适用于包含聚合的高维文本特征。
3. **XGBoost** 这一梯度提升算法也是在文本 BERT 数据采用聚合方式处理的基础上，通过优化执行速度和模型性能，为开源软件价值提供精确的预估。XGBoost 在处理聚合特征时效率高，能快速处理大量数据。

4. **基础神经网络**基于文本 BERT 数据的聚合处理，简单神经网络能够学习到特征之间的非线性关系，从而提升评估的准确度。此方法适合于数据维度较高但模型复杂度不需过高的场景。

5. **增加了 Transformer 层的深度神经网络**此模型设计包含了自建的 Transformer 层，用于处理以动态填充（Dynamic Padding）方式处理的文本 BERT 数据，即包含了全部的 BERT 文本 [cls] tokenizer 信息。使用动态填充主要是为了优化 GPU 占用率，同时保留了更丰富的文本信息，以适应网络的复杂性。

6. **自建的根据数据形式优化过的复杂 Multi 神经网络（包含自建的 Transformer 层）**此模型进一步根据数据的具体形态进行了优化，同样采用动态填充处理文本 BERT 数据。这种优化旨在提升模型处理大规模文本数据的能力，同时减少计算资源的消耗。

在这六种方法中，之所以采用聚合方式或者不包含 BERT 文本数据，是因为在网络结构较简单时，处理大量的高维文本数据可能导致训练效果不佳。简化数据处理方法有助于模型集中学习其他关键特征的影响，而不是由于数据规模过大而难以训练。另一方面，包含自建的 Transformer 层并采用动态填充的方法，则是为了在不牺牲文本信息丰富性的前提下，优化模型的计算效率。通过动态调整输入数据的长度，这种方法能够有效管理 GPU 资源，适应更复杂的网络结构，从而在维持高效计算的同时，捕获深层次的文本语义信息

自建的根据数据形式优化过的复杂 Multi 神经网络，即基于 DCNN-BERT 的方法，被选为最终的价值评估模型。该模型结合了深度卷积神经网络（DCNN）和 BERT 语言模型的优势，通过自建的 Transformer 层进一步优化，以适应开源软件数据的特殊形式。本章将详细介绍该模型的设计理念、实现过程、以及为何它在所有测试方法中表现最佳。

通过对不同算法的比较和分析，本章不仅展示了最优模型的设计和优化过程，还阐述了如何根据特定的数据形式和需求，设计和调整深度学习模型以达到最佳

的评估效果。此外，还将讨论模型在实际应用中的潜在价值和对未来研究方向的启示。

5.1.2　评估问题的具体挑战

在开源软件价值评估的领域中，核心问题可以归纳为如何有效地整合和分析来自开源软件项目的多源异构数据，以准确预测其价值。

针对非文本特征，本研究对其进行核主成分分析降维，由于核主成分分析的非线性映射，数据的分布呈现非正态和非凸而是流形聚类的特性。这对于回归任务提出了以下挑战：

1. 非线性特征关系挖掘：由于核主成分分析揭示的降维后的特征空间具有高度的非线性，评估模型需要能够识别和利用这些非线性关系。深度学习模型，尤其是那些设计用来处理复杂数据结构的模型，如深度卷积神经网络（DCNN）和增强 Transformer 网络（ETNN），能够在这种非线性特征空间中学习到有效的特征表示。

2. 非正态分布适应：核主成分分析降维后的特征可能不再遵循正态分布，这影响了传统统计模型的有效性。因此，评估方法需要对非正态分布数据有良好的适应性，能够从这些分布中准确提取价值评估相关的信息。这要求模型能够在没有明确分布假设的情况下工作，深度学习模型由于其强大的自适应能力，自然适合处理此类问题。

3. 非凸聚类特征的处理：非凸聚类特性意味着数据点可能围绕多个局部最优分布，这对寻找全局最优的评估模型构成挑战。在这种情况下，评估模型需要具备探索数据内在结构的能力，以识别和利用这些复杂的聚类模式。利用深度学习模型，特别是那些可以通过层次表示学习数据结构的模型，可能是解决这一挑战的关键。

4. 高维数据的有效利用：尽管通过核主成分分析进行了降维，数据仍然处于高维空间中。有效利用这些高维特征，同时避免“维度的诅咒”，是模型设计中的一个重要考虑。利用深度学习的分层特征提取能力，可以从高维数据中提取出最有价值的特征表示，同时通过网络的深层结构逐步减少特征维度，最终实现有效的价值预测。

针对文本特征，本研究遇到深层次文本信息挖掘问题：开源项目的仓库中包含大量的文本信息，这些文本信息蕴含着项目的潜在价值。传统方法往往忽视了这些文本信息的深层次语义分析，如何有效挖掘这部分信息成为评估工作中的一个关键挑战。

针对文本特征与非文本特征的融合，本研究遇到如下问题：

1. 协作网络结构分析问题：开源软件的开发依赖于广泛的社区协作，项目间的协作关系构成了复杂的网络结构。如何分析这种协作网络结构，并从中提取有助于价值评估的特征，是另一个需要解决的问题。

2. 高维数据处理和模型泛化问题：开源软件项目的特征维度高，数据量大，传统的评估模型往往面临着处理高维数据的困难，以及在新项目上泛化能力不足的问题。

5.2　DCNN-ETNN 模型的设计

5.2.1　本研究使用评估方法

本研究根据商业资产定价上的对比法选用统计和机器学习方法来做回归预测。传统的评估方法往往侧重于财务指标或代码质量，忽视了社区活跃度、文档质量等非量化因素的影响。为了克服这些限制，本研究提出了一种融合了最新深度学习技术的评估模型，即基于结合深度卷积神经网络和增强 Transformer 层的神经网络模型（DCNN-ETNN）的方法。

（1） 核心理念

本方法的核心理念在于综合利用开源软件的协作网络指标和 README 文件的仓库文本特征。通过深度学习模型，不仅能够捕捉到传统指标的量化信息，还能深入理解文本数据中蕴含的复杂语义信息，从而实现对开源软件价值的全面评估。

（2） 设计原则

1. **数据驱动**所有评估方法均基于实际数据进行设计和测试，确保模型的实用性和普适性。
2. **深度整合**通过深度卷积神经网络捕捉包含 Dynamic OpenRank 等指标的协作网络呢指标的深层特征，同时利用 BERT 模型理解和分析 README 的仓库文本数据。
3. **模型优化**采用贝叶斯优化、K 折交叉验证和 Adam 优化器等技术，细致调整模型参数，以求达到最佳性能。

（3） 实施框架

本研究的实施框架包括数据准备、模型设计、性能评估和结果分析四个阶段。首先，通过系统的数据采集和预处理，构建了包含社区指标和 README 文本的综合数据集。其次，设计并实现了七种不同的评估方法，包括传统的线性回归，以及基于深度学习的随机森林、XGBoost、基础神经网络、增加了 Transformer 层的深度神经网络和根据数据维度高低判断后优化的结合深度卷积神经网络和增强 Transformer 层的神经网络（DCNN-ETNN）模型。然后，通过交叉验证和特定文本数据集的评估，全面测试了各模型的性能。最后，对比分析各方法的优缺点，确定了基于 DCNN-ETNN 的模型为最优评估方法。

5.2.2 根据数据形式优化过的结合深度卷积神经网络和增强 Transformer 层的深度神经网络（DCNN-ETNN）

由于非文本特征较为高维，尽管在第四中本研究已将其通过核这成分分析的方式降维，但相较于文本信息仍处于较为高维且数据量较小的情况，故该模型通过：1. 给 Transformer 层前添加一个深度卷积神经网络；2. 增加 Transformer 层的隐藏层数以及 3. 缩减数值处理的隐藏层这三个步骤进一步优化了基于增加 Transformer 层的神经网络这一方法，通过自建的 Transformer 层和深度卷积网络精细调整模型结构，以适应开源软件数据的特殊形式：

$$V = \text{Optimized-Transformer}(\text{DCNN}(X_{\text{BERT}})) \oplus \text{Optimized-DNN}(X_{\text{Community}}) \quad (5.1)$$

这种优化实现了对开源软件价值的精确评估，展现了卓越的性能。通过深度学习技术的创新应用，该模型在所有测试方法中表现最佳，为开源软件价值评估提供了新的视角和方法论。

如图5.1所示，本模型是一个综合了深度卷积神经网络（DCNN）和自定义 Transformer 层的复合神经网络，旨在处理和分析开源软件项目的特征数据。模型结构详述如下：

1. **增强型深度卷积神经网络（Enhanced DCNN）**此模块以文本特征作为输入，通过一系列的卷积层对文本数据进行特征提取。首先，输入特征经过一个卷积层，使用 256 个过滤器和大小为 5 的卷积核，步长为 1，padding 设置为 2，并应用 ReLU 激活函数进行非线性变换。接着，通过另外两个卷积层进一步提取特征，第二层和第三层的过滤器数量分别为 512 和 1024，其他参数保持不变。这三个连续的卷积层构成了增强型 DCNN，旨在捕捉文本数据中的深层次特征。

2. **增强型 Transformer 层** DCNN 的输出接着被送入自定义的 Transformer 层进行处理。该层包含四个头的多头自注意力机制，以及前馈神经网络。Transformer 层通过自注意力机制学习序列内部的依赖关系，增强模型对文本序列

的理解。此外，每个 Transformer 块后接一个层归一化和一个小的 Dropout，以促进模型的泛化能力。

3. **数值特征处理器**与文本特征并行，数值特征通过一个单独的处理器进行处理。该处理器包含一个线性层，将数值特征映射到 128 维空间，并通过层归一化和 ReLU 激活函数进行处理。

4. **特征组合与全连接层**将 Transformer 层和数值特征处理器的输出合并后，通过一系列的全连接层进行特征融合和最终预测。首先，合并的特征通过一个 1024 维的全连接层，接着是层归一化、ReLU 激活函数和 50% 的 Dropout。随后，特征经过第二个全连接层降维到 512 维，再次应用层归一化、ReLU 和 Dropout。最后，通过一个输出维度为所需输出数量的全连接层得到最终的预测结果。

该模型通过结合 DCNN 和 Transformer 的优点，有效地处理了文本和数值特征，提高了对开源软件项目价值的预测准确度。DCNN 部分捕捉局部文本特征，而 Transformer 层能够学习长距离的依赖关系，两者的结合使模型能够全面理解和分析文本数据。同时，数值特征的并行处理确保了模型能够综合利用所有可用信息进行准确的预测。

5.3　基于 DCNN-ETNN 模型的价值评估方法

5.3.1　优化方法

为了进一步提升基于 DCNN-ETNN 模型的性能，本研究采取了以下几种优化措施，以确保模型在开源软件价值评估中的准确性和稳健性。

由于协作网络数据的数据量较小且已经维度较高，本研究主要的优化需求集中在较低维度的仓库文本特征信息上。

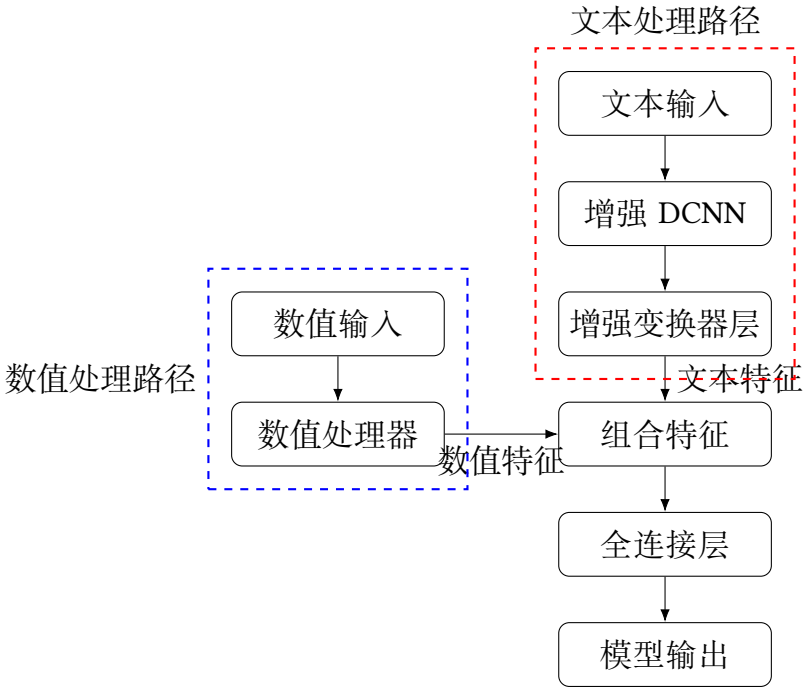

图 5.1 根据数据形式优化过的结合深度卷积神经网络和增强变换器（Transformer）层的深度神经网络（DCNN-ETNN）结构

（1） 贝叶斯优化

模型的超参数选择对其性能有着显著影响。贝叶斯优化作为一种高效的超参数优化方法，通过构建超参数的概率模型，预测给定超参数下模型的性能，从而引导下一步的参数选择。与传统的网格搜索或随机搜索相比，贝叶斯优化能够更加高效地找到性能最优的超参数组合 [84]。

贝叶斯优化的核心思想是利用已有的观测数据来更新对目标函数的认知，从而指导下一步的参数选择。具体来说，它由以下两个关键组成部分构成：

先验分布贝叶斯优化通过定义一个先验分布（通常是高斯过程，GP）来表示对目标函数的初始假设。高斯过程是定义在连续输入空间上的随机过程，任意有限集合的联合分布都是多元高斯分布。对于目标函数 $f(x)$，其先验可以表示为：

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{5.2}$$

其中，$m(x)$ 是均值函数，表示在点 x 上的期望输出；$k(x, x')$ 是协方差函数（核函数），表示输入点 x 和 x' 之间的相关性。

后验更新当获得新的观测数据 $(x, f(x))$ 后，贝叶斯优化使用贝叶斯规则来更新对目标函数的后验分布。后验分布结合了先验知识和观测数据，提供了对目标函数的更新估计。通过最大化采集函数（如期望改进 EI、贝叶斯优化算法常用的上置信界 UCB）来选择下一个评估点，这些采集函数基于当前的后验分布来平衡探索（在不确定性高的区域采样）和利用（在预期性能高的区域采样）。

对于本次研究中使用的所有神经网络的超参数优化问题，均采用了贝叶斯优化来动态调整学习率、批处理大小以及卷积层的配置。具体来说，采用高斯过程作为先验分布，通过观测模型在验证集上的性能来更新后验分布。采集函数的选择是期望改进（EI），它在每一步都会指导选择新的超参数配置，以期最大化性能提升。

贝叶斯优化能够以更少的迭代次数找到性能最优或接近最优的超参数组合，显著提升了模型性能。此外，贝叶斯优化提供的是一种概率性的优化框架，使能够量化超参数不确定性，进一步提高了模型选择的可靠性和鲁棒性。

综上所述，贝叶斯优化为评估模型的超参数调优提供了一种高效且理论上坚实的方法，显著提高了模型在开源软件价值评估任务上的性能表现。

（2） 批量归一化（Batch Normalization）

批量归一化通过规范化每个批次数据的分布，有助于缓解训练过程中的内部协变量偏移问题，从而加速模型的收敛速度并提升模型性能。以下是将批量归一化策略融入动态填充过程中的深入分析和实施建议：

批量归一化是一种广泛应用于深度学习领域的技术，它通过对每个批次的数据进行归一化处理，使得数据分布的均值为 0，标准差为 1。这种处理方式能够稳定数据分布，减少训练过程中参数更新的波动，从而提高模型的训练效率和泛化能力。

在深度学习模型中，批量归一化的数学推导和实现是基于减少内部协变量偏移（Internal Covariate Shift）的目的，从而加快模型训练过程并提高模型的泛化能力。

特别的，将批量归一化应用于 README 文件的 BERT [CLS] tokenizer 的特征计算过程，特别是在动态填充策略的实现中，可以进一步提升模型对文本特征的处理能力：

1. **批次数据归一化**在动态确定 $maxLen$ 后，对于每个批次中的 README 文件，先将通过 BERT 模型提取的特征向量进行批量归一化处理。这一步骤可以在特征向量填充或截断操作之前完成，以确保每个特征向量在进入模型前都具有统一的规模和分布。

2. **归一化参数学习**在批量归一化过程中，模型会学习到数据分布的尺度（scale）和平移（shift）参数，这些参数在模型训练过程中动态调整，以适应数据分布的变化，进一步增强模型的适应性和泛化能力。

提高模型稳定性：通过对特征向量进行批量归一化，可以有效降低模型对输入数据分布的敏感度，减少训练过程中的梯度消失或爆炸问题，从而提高模型训练的稳定性和收敛速度。

（3） 早停机制

在深度学习模型的训练过程中，过拟合是一个常见且棘手的问题，尤其是在处理高维度数据如文本和图像时。为了缓解这一问题，本研究在模型优化策略中引入了早停机制（Early Stopping）。早停机制是一种正则化手段，旨在防止模型在训练集上过度拟合，通过在验证集上监控模型性能并在性能不再提升时停止训练来实现。

早停机制的核心思想是在每个训练周期（Epoch）结束时评估模型在一个独立的验证集上的性能。这要求在数据预处理阶段将原始数据集分割为三部分：训练集、验证集和测试集。训练集用于模型的学习，验证集用于评估模型的泛化能力，而测试集则保留用于最终的性能评估。

在模型训练过程中，监控验证集上的损失函数值。如果在连续多个训练周期中验证集上的损失不再显著下降，即可认为模型开始过拟合训练数据。此时将停止训练并保存最优的模型状态。这一过程可以形式化为以下步骤：

1. **初始化**设置一个计数器 patience，用于记录验证集损失未改善的连续周期数，以及一个阈值 min_delta，表示损失减少的最小幅度，用于判断性能是否显著改善。
2. **训练与验证**在每个 Epoch 结束后，评估模型在验证集上的性能。如果当前 Epoch 的损失相比之前最低损失减少了超过 min_delta，则更新最佳模型状态并重置 patience 计数器。否则，patience 计数器加一。
3. **早停判断**如果 patience 计数器达到预设的阈值，即连续多个 Epoch 性能未显著提升，则终止训练过程，并恢复到最佳模型状态。

早停机制的引入为模型训练提供了一种有效的正则化方法，帮助模型在保持泛化能力的同时达到较高的训练效率。通过避免无谓的训练周期，早停还能节省宝贵的计算资源和时间。

然而，早停机制的效果受到 patience 和 min_delta 这两个超参数的影响。设置不当可能导致模型过早停止训练，未能充分学习数据中的模式，或者过晚停止，导致过拟合问题。

本研究通过多次试验探索这些超参数的最优值，以确保模型在最大程度上学习数据中的有用信息，同时避免过拟合问题。通过引入早停机制，本研究的模型在验证集上展现了良好的泛化能力，为开源软件价值评估提供了准确可靠的预测。

（4） K 折交叉验证

为了确保模型评估的稳健性和可靠性，本研究采用了 K 折交叉验证方法。通过将数据集分割为 K 个互斥的子集，在 $K-1$ 个子集上训练模型，并在剩余的一

个子集上进行测试，重复此过程 K 次。这种方法不仅充分利用了有限的数据资源，还通过多次评估减少了模型性能的估计偏差 [60]。

本研究特别关注模型在处理开源软件价值评估任务时的准确性和泛化能力。K 折交叉验证不仅能够充分利用有限的数据资源，而且还能减少模型评估过程中的偏差和方差，从而获得更为稳健和可靠的模型性能估计。

对于复杂的神经网络模型（包含了文本 BERT 数据），K 折交叉验证的实施特别重要，因为它考虑了模型在多个数据划分上的性能，反映了模型对于不同开源项目 README 文本数据的处理能力。此外，K 折交叉验证在超参数调优过程中发挥了重要作用，帮助识别出最适合开源软件价值评估任务的模型配置。

综上所述，K 折交叉验证在本研究中不仅增强了模型评估的稳健性，还提升了模型泛化能力的评估准确性，为基于 DCNN-BERT 的开源软件价值评估方法提供了坚实的验证基础。

（5） 使用 Adam 优化器

Adam 优化器（Adaptive Moment Estimation）由 Kingma 和 Ba 于 2014 年提出，是深度学习中广泛使用的一种自适应学习率优化算法 [85]。Adam 结合了 Momentum 和 RMSProp 优化器的优点，通过计算梯度的一阶矩（即均值）和二阶矩（即未中心化的方差）的指数移动平均值，实现了对学习率的自适应调整，特别适合处理高维空间、稀疏梯度及非静态目标的优化问题。

本研究选用 Adam 优化器来优化神经网络评估模型的参数，目的是在开源软件价值评估任务上实现更快的收敛速度和更高的模型性能。通过使用 Adam 优化器，模型能够自适应地调整每个参数的学习率，充分利用其内部一阶矩和二阶矩的信息，从而在稀疏梯度和非平稳目标函数的情况下保持稳定和高效的优化性能。

综上所述，Adam 优化器的引入为神经网络评估模型在处理复杂的开源软件价值评估任务提供了强大的支持，证明了其在深度学习模型训练中的有效性和实用性。通过细致的超参数调整和模型优化，确保了模型在本研究中的最优表现，为后续相关领域的研究提供了重要的参考和启示。

5.3.2　评估方法

（1）　均方误差

均方误差（Mean Squared Error, MSE）是衡量回归模型性能的关键指标之一，它提供了一个量化模型预测误差大小的手段。均方误差衡量的是模型预测值与实际观测值之间差异的平方和的平均值，这一度量不仅反映了模型预测的准确性，同时还帮助研究者识别模型预测中的系统性偏差。在回归分析中，均方误差的低值指示模型预测值与实际观测值之间差异小，代表模型具有较高的预测准确度。均方误差不仅重要于模型的评估阶段，其最小化也常作为回归模型训练过程中的优化目标。尽管均方误差提供了一个直观的衡量模型预测能力的方式，但它也有局限性，特别是在处理具有异常值的数据时，均方误差对于这些异常值非常敏感，可能会导致对模型性能的评估产生偏差 [59]。因此，在使用均方误差作为评估标准时，需结合模型的泛化能力进行考量。

均方误差的公式表示为：

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.3}$$

其中，y_i 是第 i 个观测的实际值，$\hat{y}_i$ 是模型预测的值，n 是样本总数 [86]。

（2）　决定系数

决定系数（R-squared）作为衡量模型解释能力的指标，在回归模型评估中占有重要地位。决定系数衡量的是模型解释的变异量占总变异量的比例，反映了模型拟合的紧密程度。它的值范围从 0 到 1，决定系数值越接近 1，表示模型对数据变异的解释程度越高，拟合效果越好 [87]。然而，决定系数并不衡量模型预测的绝对准确性，而是模型解释数据变异的能力。因此，在使用决定系数作为评价标准时，需要结合其他指标，如均方误差，而绝不应该单独用来评判模型的优劣，以获得对模型性能更全面的了解。值得注意的是，决定系数在用于评价复杂模型（如包含多

个自变量的模型）时可能会出现误导性，因为添加更多的变量到模型中往往会自然而然地提高决定系数值，而不一定意味着模型的预测性能有实质性的提升 [59]。

决定系数的公式表示为：

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5.4}$$

其中，$\bar{y}$ 是观测值的平均值 [87]。

在开源软件项目价值预测的回归模型评估中，综合考虑均方误差和决定系数指标，有助于全面理解模型的预测能力和拟合程度，确保模型既能准确预测项目价值，又具有良好的泛化能力。

5.3.3　评估过程

（1）　交叉验证

交叉验证是一种广泛应用于统计学和机器学习领域的模型评估方法，它通过将数据集分割为多个较小的子集，来评估模型对未见数据的泛化能力。

为了评估模型的稳健性和预测能力，本研究采用了 K 折交叉验证方法。在这种方法中，数据集被随机分成 K 个子集。每次一份子集被用作测试集，而其他的 $K-1$ 份子集合并作为训练集。这个过程重复 K 次，每次选择不同的子集作为测试集，最终的模型性能是这 K 次评估结果的平均值 [60]。这不仅提高了评估的可靠性，还允许有限的数据资源被最大化的利用。

该方法的优势在于其能够减少评估结果因数据划分不同而引入的偏差，提供对模型预测性能更为稳健和全面的估计。

（2）　特定文本数据集的评估

除了传统的交叉验证之外，本研究还引入了一个专门的测试数据集进行模型评估，该数据集由真实世界的开源项目影响力特征、社区活跃度特征及 README

文本组成。这一步骤旨在评估模型在处理实际应用场景时的表现，特别是模型对于自然语言文本数据的处理和预测能力。

通过对这一特定文本数据集的评估及验证集的评估结果进行对比，能够以此为依据判断模型泛化能力。考虑以下多中情况：

- 验证集结果及测试集结果都较差，则模型没有被充分训练或模型有问题。
- 验证集结果及测试集结果都较好，则模型训练成功。
- 验证集结果优于测试集结果，则模型出现过拟合现象；
- 验证集结果差于测试集结果，则模型出现欠拟合现象。

5.4　方法比较与评估

5.4.1　设计的五种对比价值评估方法

（1）　线性回归

线性回归模型通过简单的数学表达式来预测开源软件的价值。尽管线性回归模型提供了一个基础的评估框架，但它未能包含 README 文本数据的复杂自然语言特征。

（2）　随机森林

随机森林通过组合多个决策树来减少模型的方差。该模型能够处理高维数据和非线性关系，使其在包含聚合处理的 BERT 文本数据的情况下表现良好。

（3）　XGBoost

XGBoost 通过加强版的梯度提升决策树实现高效的模型训练和预测，并通过优化计算和正则化技术，有效地利用了含有聚合处理的 BERT 文本数据的特征。

（4） 增加 Transformer 层的神经网络

之前的神经网络使用仓库文本特征信息是通过对文本信息的 token 列表进行聚合操作，这种方法会有一定的信息损失。于是本研究考虑使用动态填充方法确保信息无损，而此时基础的网络结构和大小都已无法满足实验，故使用增加了 Transformer 层的神经网络通过结合自建的 Transformer 层来处理序列数据，优化网络的表示能力：

$$V = \text{Transformer}(X_{\text{BERT}}) \oplus \text{DNN}(X_{\text{Community}}) \tag{5.5}$$

其中，$\oplus$ 表示特征融合操作，X_{BERT} 是仓库文本特征，$X_{\text{Community}}$ 是协作网络特征，DNN 是深度神经网络部分。

（5） 根据数据形式优化过的结合深度卷积神经网络和增强 Transformer 层的深度神经网络（DCNN-ETNN）

由于非文本特征较为高维，尽管在第四中本研究已将其通过核这成分分析的方式降维，但相较于文本信息仍处于较为高维且数据量较小的情况，故该模型通过：1. 给 Transformer 层前添加一个深度卷积神经网络；2. 增加 Transformer 层的隐藏层数以及 3. 缩减数值处理的隐藏层这三个步骤进一步优化了基于增加 Transformer 层的神经网络这一方法，通过自建的 Transformer 层和深度卷积网络精细调整模型结构，以适应开源软件数据的特殊形式：

$$V = \text{Optimized-Transformer}(\text{DCNN}(X_{\text{BERT}})) \oplus \text{Optimized-DNN}(X_{\text{Community}}) \tag{5.6}$$

这种优化实现了对开源软件价值的精确评估，展现了卓越的性能。通过深度学习技术的创新应用，该模型在所有测试方法中表现最佳，为开源软件价值评估提供了新的视角和方法论。

5.4.2　对比结果

（1）　线性回归

如图5.2, 本研究最终采用的神经网络模型包含一个输入层、两个全连接层以及一个输出层，具体结构如下：

1. 输入层：输入层接受的特征维度为 774，即每个输入向量有 774 个特征。
2. 第一全连接层：紧随输入层之后的是第一个全连接（Fully Connected, FC）层，它具有 128 个神经元。全连接层意味着输入层中的每个神经元都与这 128 个神经元相连。这一层的目的是从输入数据中提取特征并进行非线性变换。
3. ReLU 激活函数：第一全连接层后接一个 ReLU（Rectified Linear Unit）激活函数。ReLU 函数提供了非线性变换，使得模型能够捕捉输入数据中的复杂关系。ReLU 函数的优点是计算简单且有效，有助于缓解梯度消失问题，它的公式是 f(x) = max(0, x)。
4. 第二全连接层：第一个 ReLU 激活函数后是第二个全连接层，它包含 64 个神经元。这一层进一步对第一全连接层的输出进行变换和特征提取。
5. 第二个 ReLU 激活函数：第二全连接层后同样接一个 ReLU 激活函数，为模型引入更多的非线性变换能力，提升模型的表达能力。
6. 输出层：最后是输出层，它只有一个神经元，用于输出模型的最终预测结果。在回归任务中，输出层的这一个神经元表示预测值。

如图5.3所示, 线性回归模型最终的结果均方误差为 980.89, 决定系数为-14.68%。

（2）　随机森林

本研究最终使用的随机森林即为集成上文中使用的线性回归的随机森林。

如图5.4所示，最终的结果均方误差为 766.33，决定系数为 10.40%。

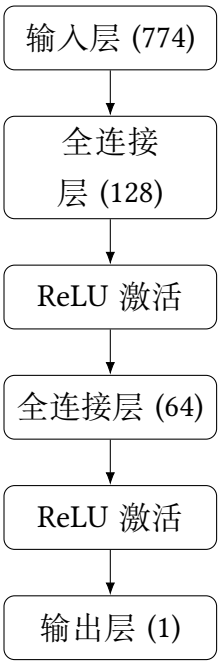

图 5.2 线性回归结构

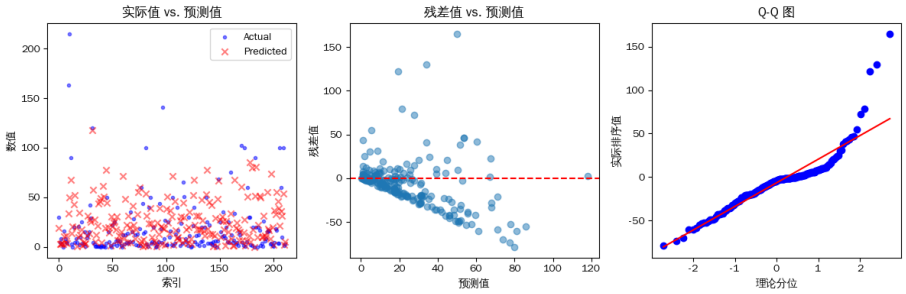

图 5.3 回归模型预测数据与结果数据对比

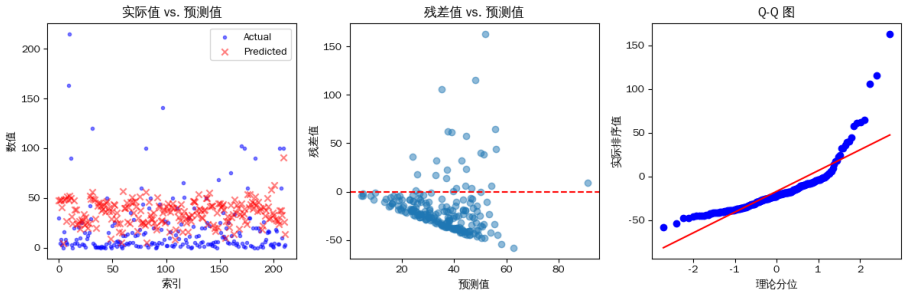

图 5.4 随机森林预测数据与结果数据对比

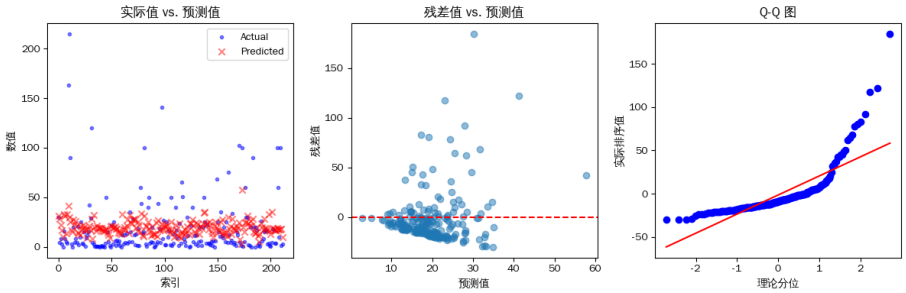

图 5.5 XGBoost 模型预测数据与结果数据对比

（3） XGBoost

本研究最终使用的 XGBoost 即为集成上文中使用的线性回归的随机森林。

如图5.5所示，随机森林模型最终的结果均方误差为 735.37，决定系数为 14.02%。

（4） 基础神经网络（文本 BERT 数据以聚合方式处理）

基础神经网络通过多层非线性变换来学习复杂的数据表示。神经网络能够捕捉深层次的数据特征，包括聚合处理的 BERT 文本数据。

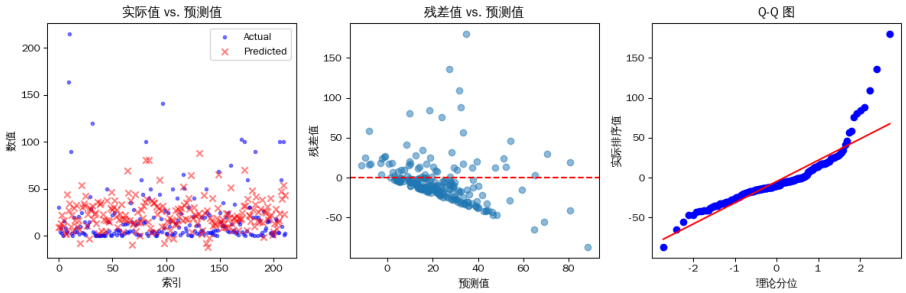

图 5.6 基础神经网络模型预测数据与结果数据对比

如图5.6所示，XGBoost 最终的结果均方误差为 827.30，决定系数为 3.27%。

（5） 增加 Transformer 层的神经网络

如图5.7所示，本文使用的结构是一个深度神经网络，专为处理具有多输入特征的开源软件项目价值评估而设计。网络结构细分为以下几个关键部分：

1. **自定义 Transformer 层**紧随 DCNN 之后的是一个自定义的 Transformer 层，这一层主要用于处理 DCNN 的输出。通过多头自注意力机制（MultiheadAttention），模型能够捕获文本特征之间的长距离依赖关系。Transformer 层还包括前馈网络、层归一化（LayerNorm）、和 Dropout，共同构成了一个能够处理序列数据的强大模块。

2. **数值特征处理器**对于数值型特征，模型采用了一个简单的前馈网络，包括一个全连接层（Linear），后接层归一化、ReLU 激活函数和 Dropout。这一设计旨在对数值特征进行标准化和非线性变换，提升模型处理数值特征的能力。

3. **特征融合与输出**将文本特征（经 Transformer 层处理）和数值特征经过处理后，通过拼接（torch.cat）合并为一个统一的特征向量。之后，该融合特征向量通过一个更深层次的前馈网络，包含多个全连接层、层归一化、ReLU 激活函数和 Dropout，最终输出项目价值的预测结果。

整体而言，这个增强型多输入模型通过结合深度卷积网络、自定义 Transformer 层和数值特征处理器，有效地处理了开源软件项目的复杂特征集合。通过这种结构设计，模型不仅能够从文本中提取深层语义信息，还能够综合考虑项目的数值型指标，为开源软件项目的价值评估提供了一种全面而深入的方法。

如图5.8所示，增加 Transformer 层的深度神经网络模型最终的结果均方误差为 758.37，决定系数为 11.33%。

（6） 根据数据形式优化过的结合深度卷积神经网络和增强 Transformer 层的深度神经网络（DCNN-ETNN）

如图5.9所示，最终的结果均方误差为 577.97，决定系数为 32.42%。

5.4.3 基线比较

由于没有之前的可比的模型，本研究使用最基础的基线（Baseline）进行比较。这是一种衡量模型性能相对于简单预测策略的有效方法。基线比较的目的是为模

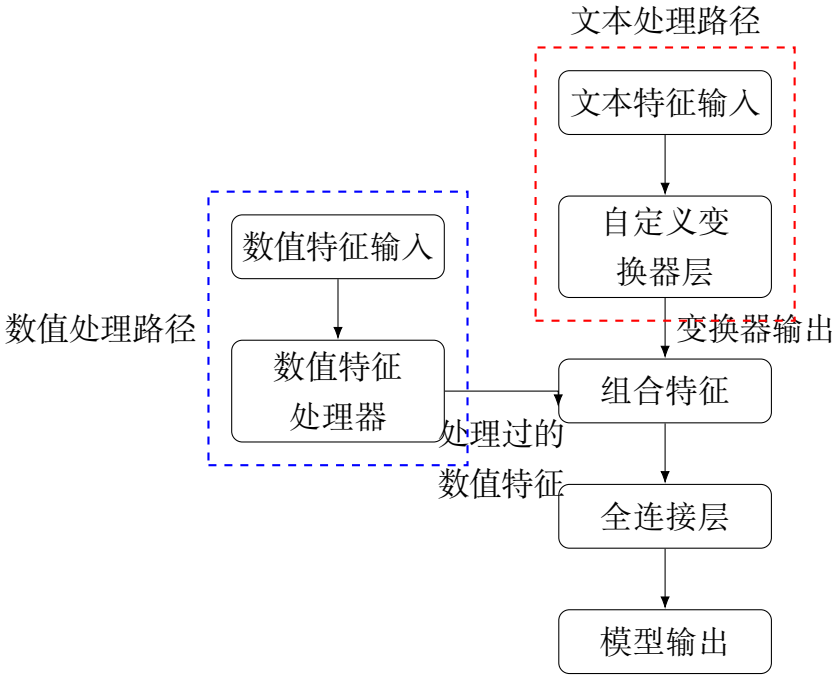

图 5.7 增加 Transformer 层的深度神经网络结构

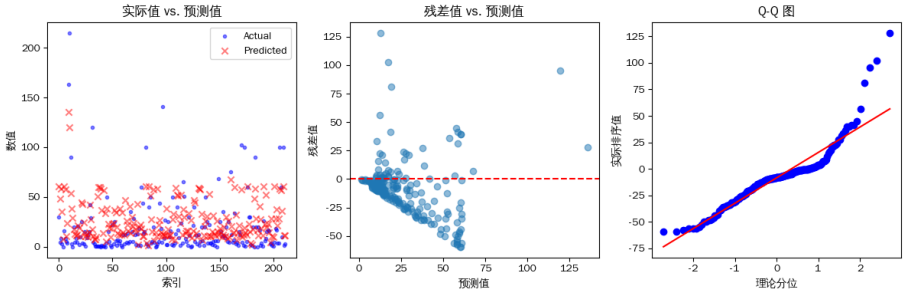

图 5.8 增加 Transformer 层的深度神经网络预测数据与结果数据对比

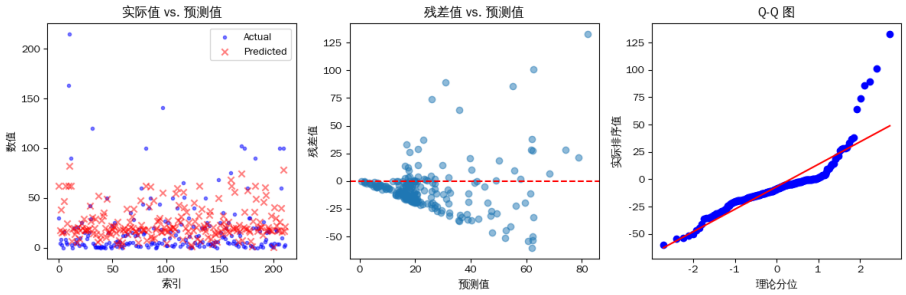

图 5.9 DCNN-ETNN 模型预测数据与结果数据的对比

型评估提供一个参考点，以此来判断模型的预测能力是否真正超越了基本的预测方法。

在进行模型评估时，除了使用均方误差和决定系数等标准评估指标外，本研究还采用了基线比较作为衡量模型性能的关键手段。基线模型采用了一种非常简单的预测策略：对于任何给定的输入，它总是预测目标变量的训练样本均值。这种方法不考虑输入特征的任何信息，仅仅使用目标值的平均值作为所有预测的基准。

（1） 基线模型的构建与评估

1. **计算训练样本的目标变量均值**首先，计算训练数据集中目标变量的均值 $\bar{y}$，该值将作为基线模型的预测输出。

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5.7}$$

2. **使用均值进行预测**在评估阶段，不论测试样本的特征如何，基线模型都将 $\bar{y}$ 作为预测值。

3. **计算基线模型的均方误差**接下来，计算基线模型在测试集上的均方误差和决定系数，作为评估其性能的标准。其中，m 是测试集中样本的数量，y_i 是第 i 个测试样本的实际目标值。

（2） 基线比较的重要性

基线比较提供了一种评估模型相对于最简单预测策略的性能优势的方法。通过与基线模型的均方误差进行比较，可以清晰地看到：

- 如果模型的决定系数显著高于基线模型的决定系数，这表明模型能够利用输入特征有效地预测目标值，具有较强的预测能力和实用价值。
- 如果模型的决定系数接近甚至低于基线模型的决定系数，这可能意味着模型未能捕捉到数据中的有用模式，或者模型存在过拟合等问题。

5.5　实验结果与分析

本研究的实验环境如表5.1所示。

在对比不同模型在开源软件价值评估方面的性能时，观察到如表5.2所示，各模型均在一定程度上优于基线模型。尤其是 XGBoost、增加 Transformer 层的深度神经网络和 DCNN 结合增强 Transformer 层的深度神经网络模型都表现出了较好的效果，表明这几个模型在解释数据变异上具有较好的性能。作为参考，本研究在表5.3中截取了部分数据及不同模型的预测结果对比作为示例，并使用图5.10、图5.11和图5.12更直观地展示对比。其中 Reg 表示 Regression 模型；RF 表示随机森林；XGB 表示 XGBoost 方法；NN 表示基本神经网络；NN w/ T 表示增加了 Transformer 层的神经网络；DCNN-T 表示结合了深度卷积神经网络和增强的 Transformer 层的深度神经网络。

综上所述，虽然基本神经网络在均方误差上取得了较低的值，但从数据解释能力来看，结合深度卷积神经网络和增强 Transformer 层的深度神经网络模型在评估中显示出更为优异的性能。这一结果强调了在开源软件价值评估任务中，考虑复杂网络结构以及有效整合文本数据处理机制的重要性。

表 5.1 实验环境

Component	Specification	Details
Operating System	Ubuntu	22.04 64-bit
CPU	Cores	16-core (vCPU)
	Model	Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz
	Architecture	x86_64
Memory	RAM	120 GiB
GPU	Number of GPUs	1
	Model	NVIDIA P100S-PCIE-32GB
	Driver(CUDA)	550.54.14(12.4)

5.6　本章小结

本章深入探讨了开源软件价值评估方法的研究，特别是基于深度卷积神经网络和增强 Transformer 层神经网络模型（DCNN-ETNN）的开发与应用。

表 5.2 模型表现对比

Model	MSE	R^2
Baseline	1361.43	-59.18%
Regression	980.89	-14.68%
Random Forest	766.33	10.40%
XGBoost	735.37	14.02%
Simple Neural Network	827.30	3.27%
Neural Network with Transformer	758.37	11.33%
DCNN-Enhanced Transformer	577.97	32.42%

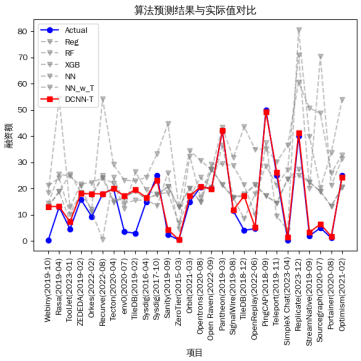

图 5.10 不同算法预测结果与实际目标结果对比折线示意图（部分行）

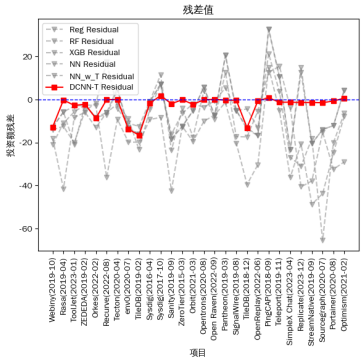

图 5.11 与实际目标结果对比残差折线示意图（部分行）

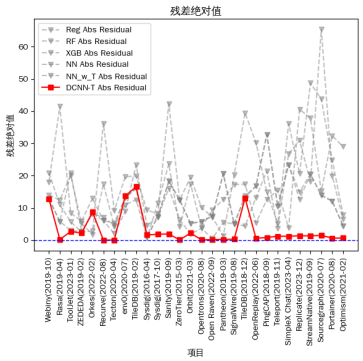

图 5.12 与实际目标结果对比残差绝对值折线示意图（部分行）

本章介绍了六种不同的评估方法：线性回归、随机森林、XGBoost、基础神经网络、复杂多层神经网络和深度神经网络结合增强 Transformer 层的神经网络。每种方法的设计和实现都经过了详细讨论，并利用交叉验证、早停等技术对其泛化能力进行了保留。

本研究以训练集指标的平均数作为基线模型，并规定出一个单独的测试集，以此对模型结果可以进行更为客观合理的比较。

最终得到结果，深度卷积神经网络结合增强 Transformer 层的深度神经网络模型（DCNN-ETNN）在评估中显示出更为优异的性能，具有最高最接近 1 的决定系数 32.42%。作为对比，基线模型的决定系数为 -59.18%，XGBoost 的决定系数为 11.33 %。

表 5.3 不同算法预测结果与实际目标结果对比（部分行）

COSS(Time)	Actual	Reg	RF	XGB	NN	NN w/ T	DCNN-T
Webiny(2019-10)	0.347	21.19	13.21	13.21	14.37	18.36	13.12
Rasa(2019-04)	13.000	23.99	18.79	18.79	25.46	54.46	13.21
ToolJet(2023-01)	4.600	10.29	25.53	7.62	24.75	12.93	7.29
ZEDEDA(2019-02)	15.90	18.40	18.54	18.54	21.04	21.80	18.09
Orkes(2022-02)	9.300	12.08	17.52	17.52	11.09	22.30	18.02
Recurve(2022-08)	18.000	54.19	24.05	25.03	0.62	24.06	18.01
Tecton(2020-04)	20.000	29.15	14.70	15.36	24.25	21.99	19.97
env0(2020-07)	3.500	23.18	16.66	16.66	14.46	12.41	17.19
TileDB(2019-02)	3.000	22.91	19.03	19.03	15.51	26.32	19.57
Sysdig(2016-04)	15.00	24.27	16.08	16.08	15.74	19.49	16.63
Sysdig(2017-10)	25.0	33.28	17.73	17.73	13.47	17.86	23.17
Sanity(2019-09)	2.4	44.76	20.71	20.71	26.02	18.50	4.33
ZeroTier(2015-03)	0.455	12.93	12.94	12.94	4.58	6.82	0.53
Orbit(2021-03)	15.0	34.51	20.16	20.16	32.40	16.91	17.21
Opentrons(2020-08)	20.600	30.73	14.82	14.82	16.60	24.24	20.69
Open Raven(2022-09)	20.000	27.12	27.44	27.44	29.28	20.54	19.86
Pantheon(2019-03)	42.000	43.31	21.41	21.41	29.37	36.55	42.25
SignalWire(2019-08)	11.500	31.73	16.53	16.53	28.74	16.62	11.79
TileDB(2018-12)	4.100	43.56	17.26	17.86	21.46	8.45	17.18
OpenReplay(2022-06)	4.700	34.99	21.46	21.46	10.05	17.93	5.28
PingCAP(2018-09)	50.00	35.18	17.27	17.27	37.39	28.64	49.18
Teleport(2019-11)	25.00	30.23	14.50	14.50	9.49	21.36	26.22
SimpleX Chat(2023-04)	0.37	36.51	23.70	23.70	4.11	27.21	1.59
Replicate(2023-12)	40.00	60.61	27.33	25.06	80.58	70.94	41.34
StreamNative(2019-09)	2.00	50.71	22.47	22.47	39.85	20.18	3.34
Sourcegraph(2020-07)	5.00	48.71	18.93	18.93	20.21	70.36	6.47
Portainer(2020-08)	1.200	26.01	13.30	13.30	33.60	21.17	1.84
Optimism(2021-02)	25.000	32.83	20.55	20.55	53.97	31.39	24.35

第六章　结论与未来工作

6.1　研究总结

本研究围绕开源软件社区的价值评估问题，提出了一种创新的评估模型，通过结合对比法和文本处理技术，旨在为开源软件投资决策提供理论和实践上的新工具。首先概述了研究背景，明确了通过构建商业开源公司数据集、开发评估模型和进行实证分析验证模型有效性的目标，并开发了用户工具以提高模型的可访问性和实用性。

其次，本文深入探讨了开源软件和商业开源公司的核心概念，强调了社区驱动的开发模式及其在促进技术创新和降低成本方面的重要性。通过详细介绍协作网络特征和仓库文本特征的评估方法，以及自然语言处理技术在提升开源项目文档理解和分析能力中的应用，本章为开源软件价值评估模型的开发奠定了理论和实践基础。

之后，本文构建了一套支持开源软件商业价值评估任务的基准数据集。通过详细讨论该数据集中各项数据（包括商业开源公司数据、开源软件数据和 README 文本数据）的收集与预处理，为后续研究和分析提供了可靠的数据基础。

之后，本文展示了一个先进的模型，该模型通过综合开源软件的协作网络指标和经 BERT 处理的仓库文本数据，以精确评估开源社区的活跃度和影响力。该模型利用改良的 Dynamic OpenRank 和文本特征分析，经过一系列的数据预处理，包括归一化和核主成分分析降维，以优化特征集。通过这种方法，研究成功地构建了一个能够全面反映开源项目社区动态和影响的评估框架。

最后，本文通过多个评估方法和基线方法的对比实证分析验证了新开发的优化的 DCNN-BERT 模型的有效性，特别强调了这些模型在预测开源软件价值方面的较为优异的表现。

6.2　未来研究方向

尽管取得了显著的研究进展和成果，但在未来的研究中，以下方向值得进一步探索：

1. 模型泛化能力的提升：由于当前的被投资开源软件公司多属于系统、基础建设和 AI 这些类别，因而当前研究多基于这些特定数据集进行模型训练和测试，未来研究应探索模型在更多不同类型的开源软件公司及组织上的适用性和稳定性，以提高模型的泛化能力。

2. 评估指标的完善：虽然本研究考虑了社区活跃度和 README 文件内容质量等因素，但仍有其他潜在因素可能影响开源软件的价值，如代码质量、项目的技术成熟度以及社区的多样性等。未来研究应进一步探索和整合这些指标，完善评估模型。

3. 深度学习技术的进一步应用：随着深度学习技术的快速发展，未来研究可以探索更先进的模型结构和算法，如 Transformer、GPT 等，以进一步提升模型在处理自然语言处理任务方面的性能。

4. 跨社区验证：未来的研究可以将本研究的模型和方法应用于其他社区（如 Bitbucket、GitLab 和 Gitee）的开源项目，验证模型的适用性和有效性。

5. 实时数据监控和动态评估：考虑到开源软件社区的动态性，未来研究可以开发实时数据监控系统和动态评估模型，以提供更及时准确的开源软件价值评估。

通过在这些方向上的深入研究，可以进一步提高开源软件价值评估模型的准确性和实用性，为开源软件社区的发展和管理提供更强大的支持工具，促进开源文化的进一步发展和繁荣。

6.3　研究的实践意义与应用前景

6.3.1　实践意义

1. 投资决策支持：通过结合先进的神经网络模型对开源软件的社区活跃度和影响力进行评估，本研究提供了一个更为精准的评估工具，帮助投资者和企业更好地识别有潜力的开源项目进行投资，降低投资风险。

2. 项目管理与优化：开源软件项目的负责人可以利用本研究提供的评估模型来监测和评估项目的健康状况，识别项目发展中的潜在问题，并据此制定策略优化社区管理和项目推广，提高项目的影响力和价值。

3. 促进开源文化的发展：本研究通过提供一种新的评估开源软件价值的方法，鼓励更多的个人和组织参与到开源项目的贡献和投资中来，从而促进开源文化的进一步发展和繁荣。

6.3.2　应用前景

1. 跨领域的广泛应用：本研究的方法和模型不仅适用于软件开发领域，也可以扩展到其他依赖于社区贡献和合作的开源项目，如开源硬件、开源数据集、开源教育资源等，具有广泛的应用前景。

2. 集成到开源平台和工具中：本研究开发的评估模型和用户工具可以集成到现有的开源项目管理平台和工具中，如 GitHub、GitLab 等，为用户提供实时的项目评估和分析功能，增强这些平台的服务能力和用户体验。

3. 推动开源项目的可持续发展：通过提供一种科学的评估方法，本研究有助于促进开源项目的可持续发展，为开源项目的长期成功提供支持，包括资源的合理分配、社区的健康管理等。

4. 政策制定和社会影响：政府和非营利组织可以利用本研究的成果来评估和支持开源项目，制定相关政策和措施，促进技术创新和社会进步。

综上所述，本研究不仅在学术上贡献了新的理论知识，而且在实践中提供了有力的工具和方法，为开源软件的投资、管理和发展提供了新的支持，展现了广阔的应用前景和深远的社会影响。

附录 A　附录

参考文献

[1] WANG X, DONG C, ZENG W, et al. Survey of Data Value Evaluation Methods Based on Open Source Scientific and Technological Information[J/OL], 2019 : 172 – 185. http://dx.doi.org/10.1007/978-981-15-0118-0_14.

[2] TASSONE J, XU S, WANG C, et al. Quality Assessment of Open Source Software: A Review[J/OL]. 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), 2018 : 411 – 416. http://dx.doi.org/10.1109/ICIS.2018.8466436.

[3] KRITIKOS A, STAMELOS I. Open source software resilience framework[C] // Open Source Systems: Enterprise Software and Solutions: 14th IFIP WG 2.13 International Conference, OSS 2018, Athens, Greece, June 8-10, 2018, Proceedings 14. 2018 : 39 – 49.

[4] SINGH J, GUPTA A, KANWAL P. The vital role of community in open source software development: A framework for assessment and ranking[J]. Journal of Software: Evolution and Process, 2023 : e2643.

[5] ADEWUMI A, MISRA S, OMOREGBE N, et al. FOSSES: Framework for open-source software evaluation and selection[J]. Software: Practice and Experience, 2019, 49(5) : 780 – 812.

[6] ZHAO Y, LIANG R, CHEN X, et al. Evaluation indicators for open-source software: a review[J/OL]. Cybersecurity, 2021, 4. http://dx.doi.org/10.1186/s42400-021-00084-8.

[7] MADAEHOH A, SENIVONGSE T. OSS-AQM: An Open-Source Software Quality

Model for Automated Quality Measurement[C] // 2022 International Conference on Data and Software Engineering (ICoDSE). 2022 : 126 – 131.

[8] CROWSTON K, HOWISON J. Assessing the health of open source communities[J]. Computer, 2006, 39(5) : 89 – 91.

[9] ALEXANDER HARS S O. Working for free? Motivations for participating in open-source projects[J]. International journal of electronic commerce, 2002, 6(3) : 25 – 39.

[10] STEWART K J, GOSAIN S. The impact of ideology on effectiveness in open source software development teams[J]. Mis Quarterly, 2006 : 291 – 314.

[11] KOYYADA S P, DESHMUKH D, BADAMPUDI D, et al. Towards automated open source assessment - An empirical study[J/OL]. ArXiv, 2022, abs/2212.00087. http://dx.doi.org/10.48550/arXiv.2212.00087.

[12] AKATSU S, FUJITA Y, KATO T, et al. Structured analysis of the evaluation process for adopting open-source software[J/OL], 2018 : 1578 – 1586. http://dx.doi.org/10.1016/j.procs.2018.08.131.

[13] WASSERMAN A I. Software engineering issues for mobile application development[C] // Proceedings of the FSE/SDP workshop on Future of software engineering research. 2010 : 397 – 400.

[14] FITZGERALD B. The transformation of open source software[J]. MIS quarterly, 2006 : 587 – 598.

[15] RIEHLE D. The commercial open source business model[C] // SIGeBIZ track of the Americas Conference on Information Systems. 2009 : 18 – 30.

[16] PANICHELLA S, DI SORBO A, GUZMAN E, et al. How can i improve my app? classifying user reviews for software maintenance and evolution[C] // 2015 IEEE

international conference on software maintenance and evolution (ICSME). 2015 : 281 – 290.

[17] GUZMAN E, BRUEGGE B. Towards emotional awareness in software development teams[C] // Proceedings of the 2013 9th joint meeting on foundations of software engineering. 2013 : 671 – 674.

[18] BIRD S, KLEIN E, LOPER E. Natural language processing with Python: analyzing text with the natural language toolkit[M]. [S.l.] : " O'Reilly Media, Inc.", 2009.

[19] ALQAIMI A, THONGTANUNAM P, TREUDE C. Automatically generating documentation for lambda expressions in java[C] // 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). 2019 : 310 – 320.

[20] KLEIN D, MANNING C D. Accurate unlexicalized parsing[C] // Proceedings of the 41st annual meeting of the association for computational linguistics. 2003 : 423 – 430.

[21] BACCHELLI A, BIRD C. Expectations, outcomes, and challenges of modern code review[C] // 2013 35th International Conference on Software Engineering (ICSE). 2013 : 712 – 721.

[22] LAM A N, NGUYEN A T, NGUYEN H A, et al. Combining deep learning with information retrieval to localize buggy files for bug reports (n)[C] // 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2015 : 476 – 481.

[23] LAMKANFI A, DEMEYER S, GIGER E, et al. Predicting the severity of a reported bug[C] // 2010 7th IEEE working conference on mining software repositories (MSR 2010). 2010 : 1 – 10.

[24] NOVIELLI N, CALEFATO F, LANUBILE F. A gold standard for emotion annotation in stack overflow[C] // Proceedings of the 15th international conference on mining software repositories. 2018 : 14 – 17.

[25] NOVIELLI N, CALEFATO F, LANUBILE F. Towards discovering the role of emotions in stack overflow[C] // Proceedings of the 6th international workshop on social software engineering. 2014 : 33 – 36.

[26] GEZICI B, ÖZDEMIR N, YILMAZ N, et al. Quality and success in open source software: a systematic mapping[C] // 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). 2019 : 363 – 370.

[27] YILMAZ N, KOLUKISA TARHAN A. Quality evaluation models or frameworks for open source software: A systematic literature review[J]. Journal of Software: Evolution and Process, 2022, 34(6) : e2458.

[28] RASHID M, CLARKE P M, O' CONNOR R V. A systematic examination of knowledge loss in open source software projects[J]. International Journal of Information Management, 2019, 46 : 104 – 123.

[29] RAYMOND E S. The Cathedral and the Bazaar, in 'The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary', O' Reily & Associates[J]. Inc., Cambridge, 1999 : 27 – 78.

[30] Red Hat, Inc.. Red Hat: Open source solutions for enterprise servers, cloud, virtualization, storage, middleware - `https://www.redhat.com`[H]. 2023.

[31] Canonical Ltd.. Canonical: The company behind Ubuntu - `https://www.canonical.com`[H]. 2023.

[32] MongoDB, Inc.. MongoDB: The most popular database for modern apps - `https://www.mongodb.com`[H]. 2023.

[33] Oracle Corporation. MySQL - https://www.mysql.com[H]. 2023.

[34] Talend Inc.. Talend: Data integration and data integrity - https://www.talend.com[H]. 2023.

[35] Elasticsearch B.V.. Elastic: Search. Observe. Protect - https://www.elastic.co[H]. 2023.

[36] GitLab Inc.. GitLab: The DevOps platform - https://www.gitlab.com[H]. 2023.

[37] GitHub, Inc.. GitHub: Where the world builds software - https://www.github.com[H]. 2023.

[38] Databricks Inc.. Databricks: The Data and AI company - https://www.databricks.com[H]. 2023.

[39] Nextcloud GmbH. Nextcloud: The self-hosted productivity platform - https://www.nextcloud.com[H]. 2023.

[40] FELLER J, FINNEGAN P, KELLY D, et al. Developing Open Source Software: A Community-Based Analysis of Research[C] // TRAUTH E M, HOWCROFT D, BUTLER T, et al. Social Inclusion: Societal and Organizational Implications for Information Systems. Boston, MA : Springer US, 2006 : 261 – 278.

[41] INSTITUTE. A. The Appraisal of Real Estate (13th ed.).[M]. [S.l.] : Appraisal Institute., 2008.

[42] PRITCHARD A. Comparative Method in Software Valuation[J]. Journal of Software Value Management, 2017, 1(1) : 34 – 45.

[43] RIGGS J, WEST T. Financial Management for the Construction Industry[M]. [S.l.] : William Morrow, 1986.

[44] HITCHNER J R. Financial Valuation,+ Website: Applications and Models[M]. [S.l.] : John Wiley & Sons, 2017.

[45] BREALEY R A, MYERS S C, ALLEN F, et al. Corporate finance : Vol 8[M]. [S.l.] : McGraw-Hill/Irwin Boston et al., 2006.

[46] DAMODARAN A. Investment valuation: Tools and techniques for determining the value of any asset : Vol 666[M]. [S.l.] : John Wiley & Sons, 2012.

[47] WHEELER D A. More than a gigabuck: Estimating GNU/Linux ' s size[H]. 2001.

[48] HAHN R. The Economic Value of Open Source Software[J], 2002.

[49] FITZGERALD B. The Transformation of Open Source Software[J]. MIS Quarterly, 2006, 30(3) : 587 – 598.

[50] MOCKUS A, FIELDING R, HERBSLEB J. Two Case Studies of Open Source Software Development: Apache and Mozilla[J/OL]. ACM Transactions on Software Engineering and Methodology, 2000, 11(3) : 309 – 346.
https://www.example.com/apache-mozilla-casestudy.

[51] CROWSTON K, WEI K, HOWISON J, et al. Free/Libre open-source software development: What we know and what we do not know[J]. ACM Computing Surveys (CSUR), 2008, 44(2) : 1 – 35.

[52] STEWART K J, AMMETER A P, MARUPING L M. Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects[J]. Information Systems Research, 2006, 17(2) : 126 – 144.

[53] MOCKUS A, FIELDING R T, HERBSLEB J D. Two case studies of open source software development: Apache and Mozilla[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2002, 11(3) : 309 – 346.

[54] CROWSTON K, KWASNIK B H. A framework for creating a facetted classification for genres: Addressing issues of multidimensionality[C] // 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the. 2004 : 9 – pp.

[55] DABBISH L, STUART C, TSAY J, et al. Social coding in GitHub: transparency and collaboration in an open software repository[C] // Proceedings of the ACM 2012 conference on computer supported cooperative work. 2012 : 1277 – 1286.

[56] THUNG F, BISSYANDE T F, LO D, et al. Network structure of social coding in github[C] // 2013 17th European conference on software maintenance and reengineering. 2013 : 323 – 326.

[57] ZHAO S, XIA X, FITZGERALD B, et al. OpenRank Leaderboard: Motivating Open Source Collaborations Through Social Network Evaluation in Alibaba[C] // . 2023.

[58] GUYON I, ELISSEEFF A. An introduction to variable and feature selection[J]. Journal of machine learning research, 2003, 3(Mar) : 1157 – 1182.

[59] JAMES G, WITTEN D, HASTIE T, et al. An introduction to statistical learning : Vol 112[M]. [S.l.] : Springer, 2013.

[60] KOHAVI R, OTHERS. A study of cross-validation and bootstrap for accuracy estimation and model selection[C] // Ijcai : Vol 14. 1995 : 1137 – 1145.

[61] POWERS D M. EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION[J], 2011.

[62] BREIMAN L. Random forests[J]. Machine learning, 2001, 45 : 5 – 32.

[63] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. nature, 2015, 521(7553) : 436 – 444.

[64] SEBER G A, LEE A J. Linear regression analysis[M]. [S.l.] : John Wiley & Sons, 2012.

[65] LIAW A, WIENER M, OTHERS. Classification and regression by randomForest[J]. R news, 2002, 2(3) : 18 – 22.

[66] CHEN T, GUESTRIN C. Xgboost: A scalable tree boosting system[C] // Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016 : 785 – 794.

[67] RUMELHART D E, HINTON G E, WILLIAMS R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088) : 533 – 536.

[68] JURAFSKY D, MARTIN J H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition[H]. .

[69] MANNING C D, SURDEANU M, BAUER J, et al. The Stanford CoreNLP natural language processing toolkit[C] // Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. 2014 : 55 – 60.

[70] LOUREIRO D, REZAEE K, PILEHVAR M T, et al. Language models and word sense disambiguation: An overview and analysis[J]. arXiv preprint arXiv:2008.11608, 2020.

[71] LIU B. Sentiment analysis and opinion mining[M]. [S.l.] : Springer Nature, 2022.

[72] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[73] DEVLIN J, CHANG M-W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[74] RADFORD A, NARASIMHAN K, SALIMANS T, et al. Improving language understanding by generative pre-training[J], 2018.

[75] YANG Z, DAI Z, YANG Y, et al. Xlnet: Generalized autoregressive pretraining for language understanding[J]. Advances in neural information processing systems, 2019, 32.

[76] LIU Y, OTT M, GOYAL N, et al. Roberta: A robustly optimized bert pretraining approach[J]. arXiv preprint arXiv:1907.11692, 2019.

[77] CLARK K, LUONG M-T, LE Q V, et al. Electra: Pre-training text encoders as discriminators rather than generators[J]. arXiv preprint arXiv:2003.10555, 2020.

[78] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[79] TAN X, ZHOU M, FITZGERALD B. Scaling open source communities: An empirical study of the Linux kernel[C] // Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020 : 1222 – 1234.

[80] BIRD S. NLTK: the natural language toolkit[C] // Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions. 2006 : 69 – 72.

[81] GOUSIOS G. The GHTorent dataset and tool suite[C] // 2013 10th Working Conference on Mining Software Repositories (MSR). 2013 : 233 – 236.

[82] KALLIAMVAKOU E, GOUSIOS G, BLINCOE K, et al. The promises and perils of mining github[C] // Proceedings of the 11th working conference on mining software repositories. 2014 : 92 – 101.

[83] PRANA G A A, TREUDE C, THUNG F, et al. Categorizing the content of github readme files[J]. Empirical Software Engineering, 2019, 24 : 1296 – 1327.

[84] SNOEK J, LAROCHELLE H, ADAMS R P. Practical bayesian optimization of ma-

chine learning algorithms[J]. Advances in neural information processing systems, 2012, 25.

[85] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[86] HYNDMAN R J, KOEHLER A B. Another look at measures of forecast accuracy[J]. International journal of forecasting, 2006, 22(4): 679 – 688.

[87] THEIL H. Economic forecasts and policy[J]. (No Title), 1961.

致　　谢

在此，我想对所有支持我完成硕士学位的人表示最深切的感谢。首先，我必须特别感谢我的指导教师，王伟教授，他们不仅在学术上给予我巨大的帮助，更在生活中给予我关怀和支持。王伟教授对我的研究工作提出了宝贵的意见，并始终以耐心和专业的态度引导我。感谢钱卫宁院长对我一直以来的关怀、指导和帮助。感谢郭健美教授和黄波曾经给予我的悉心教导与包容。

我还要感谢我的家人，尤其是我的父母，他们无条件的爱和支持是我最坚强的后盾。感谢我的朋友们（尤其是梅子（Jessi May）、Clarisse Xi、Leyi Zhu、大毛（Ruijie Xu）、Zhangyue Yin（NLPer YZY）、一哥（周添一）、佳伟、超哥、吴博士、浩然、方雷、珂斌、徐姐、Jinn 和杨桑等），他们的理解和陪伴使我在学术旅程中从未感到孤独。

同样，我对我的同学（尤其是同一训练营的翔宇、志成、烨男、Andy 和衍童）和 X-lab 开放原子实验室团队成员表示衷心的感谢，感谢他们在研究过程中的合作和鼓励。我还要感谢同济大学的 Frank 和开源软件基金会，他们提供的技术支持和资源对我的研究至关重要。

最后，我希望通过我的努力，能够对我的研究领域作出贡献，并以此论文作为我学术旅程的一个起点，继续探索和学习。感谢所有人的支持和鼓励，没有你们，就没有我的今天。

吴 双

二零二四年三月

攻读硕士学位期间参与的项目以及学术成果

■ 已完成学术论文

1. **A Fast Machine Learning Framework with Distributed Packet Loss Tolerance**, MLANN 2024（已接收未发表）