项目申请书

项目名称：OSS Compass-开源社区画像

项目主导师：王晔晖

申请人：张益翔

日期：2024.06.01

邮箱：[zyx19937225859@163.com](mailto:zyx19937225859@163.com)

目录

[项目申请书 1](#_Toc168229890)

[1 项目背景 3](#_Toc168229891)

[1.1用户满意度评估 4](#_Toc168229892)

[1.2 ReadMe文档的快速评估和打分 4](#_Toc168229893)

[1.3代码内生质量评估 4](#_Toc168229894)

[1.4 项目的相关仓库 4](#_Toc168229895)

[2技术方法及可行性 6](#_Toc168229896)

[2.1爬虫与多维度特征融合 6](#_Toc168229897)

[2.2 文档分类和评估技术 6](#_Toc168229898)

[3项目的实现细节 7](#_Toc168229899)

[3.1用户满意度评估模型 7](#_Toc168229900)

[3.2 ReadMe文档的快速评估和打分 8](#_Toc168229901)

[3.3代码内生质量评估 10](#_Toc168229902)

[4规划 11](#_Toc168229903)

[4.1项目研发第一阶段（07 月 01 日 - 08 月 15 日）： 11](#_Toc168229904)

[4.2项目研发第二阶段（08 月 16日 - 09 月 30日）： 11](#_Toc168229905)

[4.3 期望： 11](#_Toc168229906)

[5参考文献 12](#_Toc168229907)

# 1 项目背景

目前的OSS Compass的开源评估体系共有十几个评估模型[1]。通过软件生产力，稳健性和创新力以及贡献者的生产力在三维空间内综合评估开源生态。

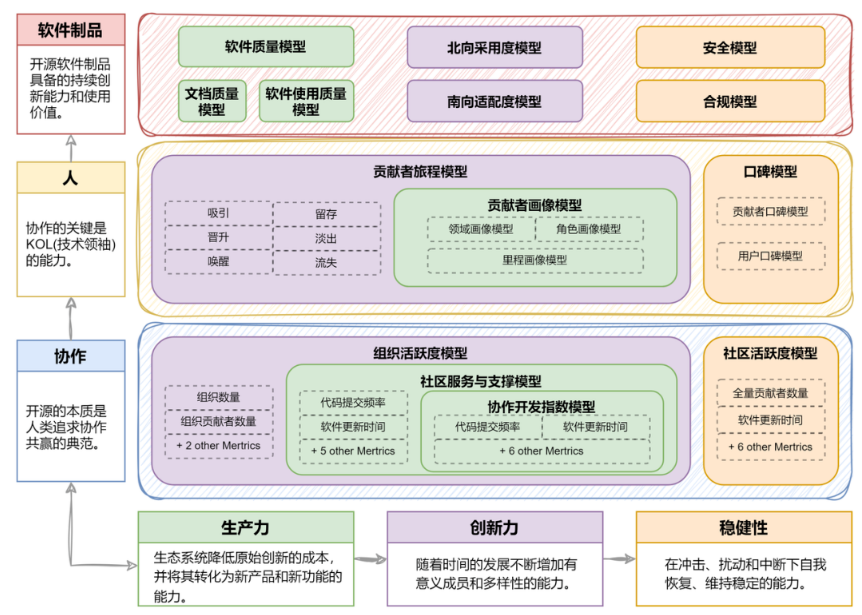


图1 OSS Compass评估体系[1]

软件与开源生态的交织产生了七种重要的评价模型：具有生产力属性的软件质量模型、软件可用性模型和文档质量模型;具有创造属性的北向采用模式和南向适应模式;以及具有稳健性属性的安全模型和合规模型。这些模型通过计算相关的指标，例如协作发展指数，开源项目活跃度等多维度评估开源项目的生态环境。

当前 OSS Compass 已对开源社区有了初步的洞察数据和评估能力（贡献者画像、贡献分布画像），能够初步满足用户和开源社区的管理人员进行相关项目的评估需求。

## 1.1用户满意度评估

OSS compass的七种模型主要从开发者和管理者的视角来评估开源项目的完善程度，因此在用户视角方面有一定的欠缺。同时模型中还有一些评估指标未被纳入到评估模型中，可以考虑，采用一些用户视角的指标有助于更深度的完善OSS Compass的社区洞察能力。

从评估的合理性方面，同时参考最新的issue[2]，考虑加入融合以下几种指标：漏洞修复是否及时，代码使用情况，代码收欢迎程度，开发和测试时长，版本更新时长等。

## 1.2 ReadMe文档的快速评估和打分

在ReadMe和用户许可证种也包含了软件的很多信息，因此希望能够通过LLM对ReadMe文档和用户许可证的快速评估。通过大语言模型对词语和句子的理解程度以及其优秀的分类能力，实现对项目ReadMe文档和许可证的评估得分，进一步完善开源项目的评估模型。

## 1.3代码内生质量评估

评估开源项目的价值、稳定性和可持续性，对开发者至关重要。除了社区对代码评价之外，项目代码本身的质量也是衡量项目的重要标准。对于代码本身的质量评估，本项目我打算根据代码的可读性，可维护性，以及是否进行异常处理等情况,综合评估代码的内生质量。

## 1.4 项目的相关仓库

* 项目web服务：https://github.com/oss-compass/compass-web-service
* 项目web服务：<https://github.com/oss-compass/compass-web>
* 项目评估模型：<https://github.com/oss-compass/compass-metrics-model>
* 项目模型说明：https://github.com/chaoss/wg-metrics-models

在评估模型仓库中包括了多个指标的实现方法，同时针对这些指标的使用场景做出了相关解释。比如在协作发展指数，项目在仓库的ReadMe文档做出了详细的说明。

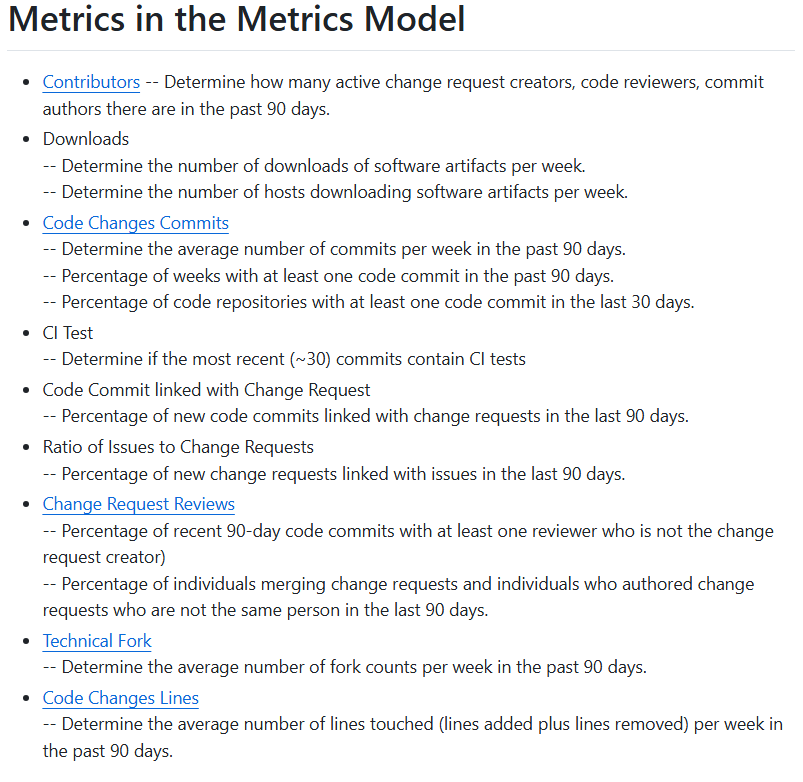


图2 协作开发指数相关指标[3]

项目评估模型仓库展示了模型在单个指标的实现，以及针对不同指标的提取方式。同时，模型仓库也给出了使用方式和需要配置的conf.yaml。

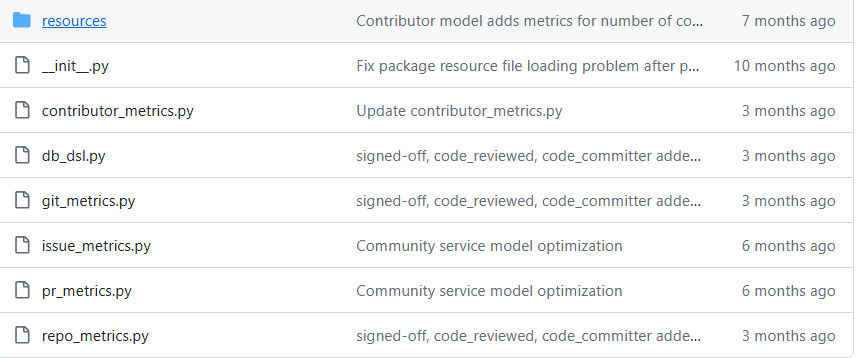
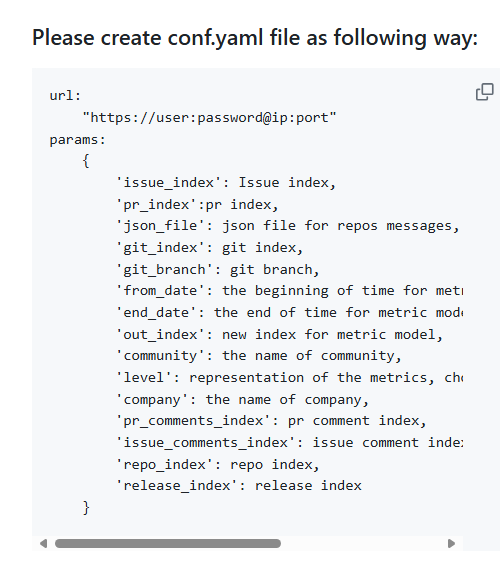
 

图3 相关实现指标以及开发的配置文档[3]

# 2技术方法及可行性

本项目在技术方案上主要包括爬虫技术以及多维度特征融合，以及文档分类与评估两项技术。

## 2.1爬虫与多维度特征融合

之前在做开源竞赛平台过程中，做过相关指标的提取。开源竞赛平台开发过程中，主要采用爬虫技术，爬取评论点赞的相关信息处理，最终得到评分。因此具有一定的爬虫经验。

另一方面，针对多指标融合任务，OSS Compass之前的做法是采用阈值筛选的方式，通过设定特定阈值，来判断当前指标是否达到合适的效果，之后根据不同指标的权重进行融合。之前在做C/C++二进制程序的第三方库检测工作中，采用过类似的方式，将多指标融合。

## 2.2 文档分类和评估技术

目前，大模型和深度学习模型对自然语言的理解已经有了一定的效果。之前在本科时候，做过文本分类的相关工作，当时采用动态LSTM针对垃圾邮件进行分类。因此具备一定的文档分类和评估的技术。

在大模型方面，目前大语言模型在自然语言处理、机器翻译、文本生成等领域取得了显著的成果。同时对我本身，之前为了针对C/C++的二进制第三方组件检测工作制作检测数据集，利用LLM构建过特定prompt，将archlinux中的项目makefile编译选项和pkgbuild进行批量修改，具有一定经验。

# 3项目的实现细节

## 3.1用户满意度评估模型

OSS Compass目前的评估模型主要从开发者和管理人员的视角进行评估，但在用户视角方面的评估还有待欠缺。用户在开源生态中也占据着重要席位。用户的使用量和对项目的优良评论数均反映了很多结果。

用户满意度评估模型的主要目的是从用户的视角来评估整个开源项目的使用情况。用户关注的最主要的可能是当前库或者开源项目的使用量，以及当前项目的受欢迎度和开发者对项目的关注。

用户的反馈还可以帮助开发团队更好地了解用户需求，从而改进和优化项目功能，提升用户体验。通过持续的用户反馈和数据分析，开发团队可以不断优化项目，使其更符合用户期望，进而提高用户满意度和项目的长期发展。从而进一步完善开源生态。

用户满意度模型，目前初步想法是利用多种评估指标融合的方法进行整体评估。技术层面，可以通过利用爬虫或者API接口技术提取相关特征，之后通过多维度特征融合进行模型训练。

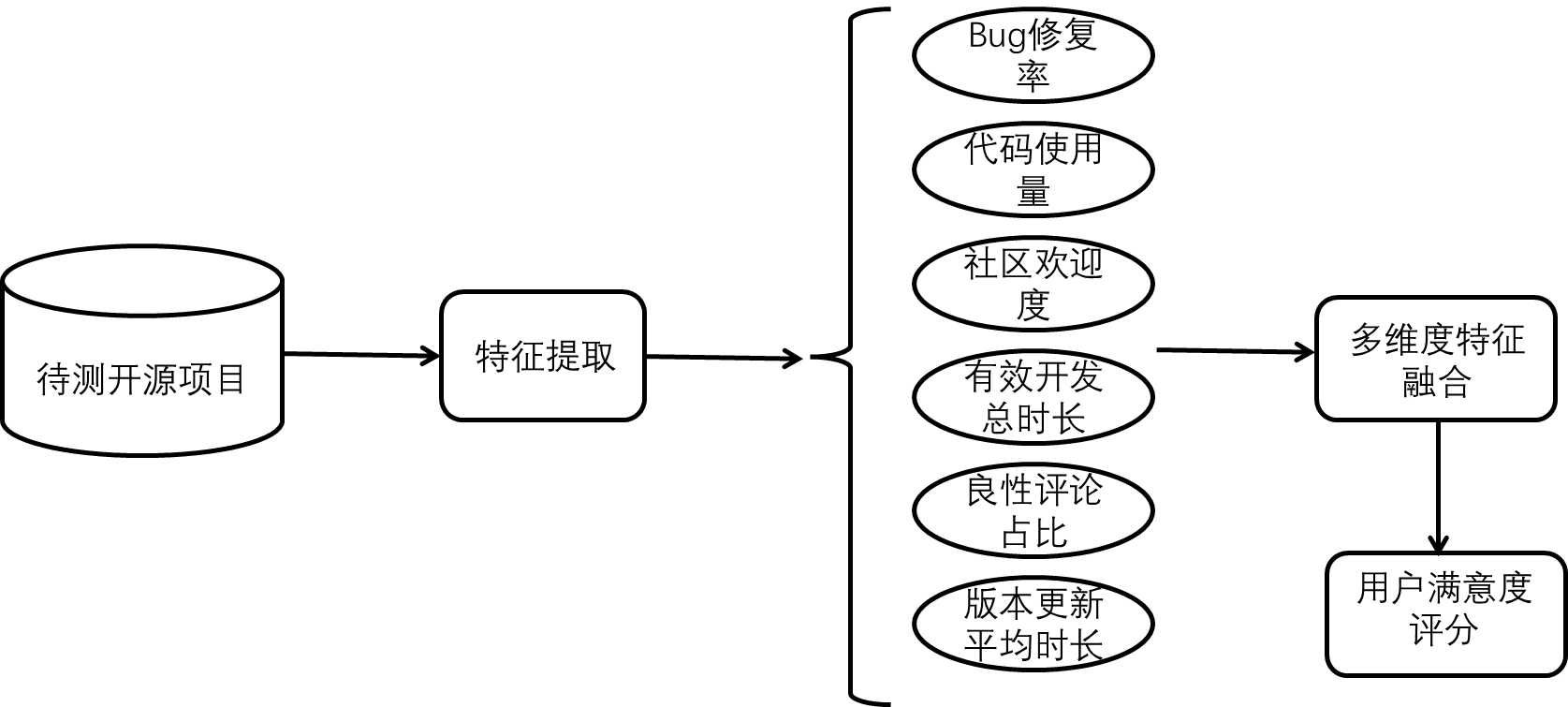


图4 用户满意度评估模型

1. 漏洞修复是否及时。作者对于项目issue中所提到的bug的修复情况，可以通过bug issue的提出时间到解决时间的情况判别。
2. 代码使用量。用户采用代码和调用此代码的次数，可以通过项目下载量判别。
3. **Popularity**。用户对代码的star与收藏情况，利用爬虫或者github，gitee接口。
4. 有效开发总时长。团队人员协作开发过程中的总时长，可通过开源项目从创建到发布上线的总时长来统计。
5. 用户口碑较好的评论占比。针对项目的评论数进行爬取，根据语义和语境态度，采用自然语言处理的方式进行分类，计算良性评论占比。
6. 版本更新平均时长。版本更新时长的越短，证明作者对用户提出的意见修改更及时。可以采用爬虫或者调用API接口的方式，记录版本之间的更新时间。
7. 相同软件不同行业所使用的数量。（软件普适性）
8. 论坛，论坛评论和点赞。数据类型越多。论坛能否适应当前模型。

## 3.2 ReadMe文档的快速评估和打分

在开源项目的世界里，一个出色的 README.md 文件就像是一座灯塔，指引着初次来访的开发者和用户了解项目的核心价值和使用方法。README.md 不仅是项目的入口，更是项目的名片。它向访问者展示了项目的目标、功能和使用方法。一个详尽、清晰的 README 能够帮助开发者快速理解项目的价值和用途，从而决定是否要使用或参与该项目。因此在开源生态评估中，一个出色的 README.md 文件是很重要的。

README.md 文件是项目的门面，其重要性不言而喻。一个完整的 README.md 通常囊括项目标题、详细描述、清晰的安装指南、生动的使用示例、友好的贡献指南以及明晰的许可证说明。这些部分的结合，旨在为读者提供全方位、深入的了解，使其能够轻松上手并深入参与到项目中。简洁性、明确性和具体性是编写 README 文档时不可或缺的三大准则，它们为文档赋予了清晰、简洁、具体的特质，使读者能够快速把握项目的核心信息。

优秀的ReadMe文档应该写作规范，标题清晰，同时能够为读者提供明确的操作说明。因此在这里提出一套评估指标用于评估ReadMe文档，主要包括，完整性，清晰度，结构示例和更新性。

1. 完整性：文档是否包含了项目或API的所有重要信息，如功能介绍、安装指南、使用示例、常见问题解答等。
2. 结构：文档是否有清晰的结构和布局，包括标题、段落、列表、代码块等，以便读者能够快速找到所需信息。清晰度：文档是否清晰易懂，使用简洁明了的语言描述功能和用法，避免术语混淆或表达不清的情况。
3. 示例：文档是否提供了具体的示例代码，帮助开发者更好地理解如何使用项目或API，并加快上手速度。
4. 更新性：文档是否及时更新，反映了项目或API的最新功能和变化，避免信息过时导致误解或困惑。

技术上主要采用大语言模型进行预训练，之后对ReadMe文档进行打分和评估。目前大型语言模型（LLM）在自然语言处理（NLP）领域的成功应用，这些模型在文本分类任务中也取得了显著的进展。大语言模型通过大规模的预训练和微调，能够在文本分类中展现出其优势。

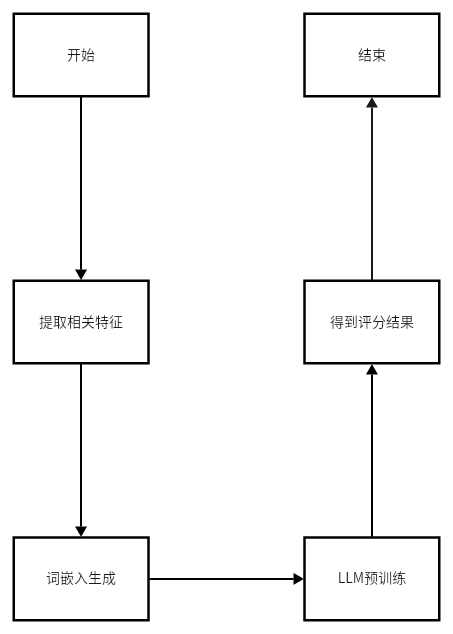


图5 LLM文本训练流程

具体实现上，首先需要进行优秀文档的标注（打标签），这部分需要手工标记，过度耗时。因此，目前希望能够使用OSS compass之前用于评估训练的数据集，分析这些数据集中，优秀的ReadMe文档。之后采用LLM模型进行训练，构建合适的prompt，之后通过针对优秀文档的预训练，最终得到一个自动评分模型。

## 3.3代码内生质量评估

软件质量是软件产品内部固有的、与生俱来的质量特性，这些特性决定了软件在特定条件下满足其功能和性能的能力。软件质量包括了功能性、可靠性、效率、可维护性、可移植性、 安全性等多个方面。代码内生质量也与之类似，代码内生质量更多关注代码的本身的情况，包括漏洞情况，代码注释信息，代码规范，语法情况。

另一方面，代码内生质量也可反映出开源项目本身的工作量和开发者对项目的实际代码贡献，可以假设项目功能点和代码质量呈正比关系。目前主要思路有三个模块，包括代码体积，代码规范性和代码可读性。

代码体积主要是反应代码的长度和工作量，因此这部分主要包括代码行数，复杂度，以及语法复杂度。具体解释如下：

1. 有效代码行数：行数因开发语言而异，可以为代码行数分配较低权重。
2. 代码复杂度指标：圈复杂度、其他复杂度指标。
3. 语法复杂度、多样性：做语法树分析，分析语法单元的数目、类别。

代码规范性是反映开发者开发代码时的质量，可以通过代码的命名规则，代码内部不同模块的耦合程度来评估。

1. 标识符命名规范：整个项目命名风格是否都一致，都遵循驼峰命名、下划线命名或其他命名格式。
2. 模块耦合程度：遵循低耦合、高内聚的原则。

代码可读性是在代码复用人员的视角下，代码是否可读。这部分可以通过代码的注释和标识符的含义来判断。可以通过计算模块、类和函数上的注释占比，以及标识符是否含有自然语言来评估。

目前在技术实现部分，代码体积的代码行数和复杂度指标均可以完美实现，通过Halstead’s Volume[4]目前可以较好的评估一个代码的复杂度。语法复杂度方面，SonarQube[5] 可以快速的定位代码中潜在的或者明显的错误。同时也开源，可以用SonarQube的API来自定义构建语法解析工具。代码规范性与代码可读性主要是分析代码相关标识符的和注释，这部分可以采用LLM辅助完成。

# 4规划

## 4.1项目研发第一阶段（07 月 01 日 - 08 月 15 日）：

* 完成用户满意度模型的相关指标的提取以及特征融合
  + 用户满意度相关指标的调用与爬取
  + 用户满意度多维度特征融合
* 采用深度学习或者大模型进行ReadMe文档和许可证文档评分
  + 大模型prompt的构建与数据集的收集测试
  + 完成文档评分和许可证评分方案

## 4.2项目研发第二阶段（08 月 16日 - 09 月 30日）：

* 解决在中期验收阶段发现的问题
* 对第一阶段的内容进行更详细的测试
* 代码内生质量模型的研发与测试
* 准备最终答辩

## 4.3 期望：

希望借助开源之夏这个机会，系统地学习开源生态评估相关内容，深入了解开源社区画像对开发者和管理人员的重要性。同时，通过此项目，希望总结某一特定领域优秀开源项目的特点，掌握开源项目评估指标的特征和融合方法。最后，期望能为OSS Compass项目做出自己的贡献。

# 5参考文献

[1] https://oss-compass.org/zh/docs/quick-start/

[2] <https://github.com/oss-compass/compass-metrics-model/issues/118>

[3] https://oss-compass.org/docs/docs/metrics-models/

[4] https://sourcery.ai/blog/maintainability-index/

[5] https://www.sonarsource.com/products/sonarqube/

https://ohpm.openharmony.cn/#/cn/home