# AN INVESTIGATION OF ISSUE LABELING IN OPEN SOURCE SOFTWARE PROJECTS USING LARGE LANGUAGE MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

İREM SELİN DENİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2024

**AN INVESTIGATION OF ISSUE LABELING IN OPEN SOURCE SOFTWARE PROJECTS USING LARGE LANGUAGE MODELS**

submitted by **İREM SELİN DENİZ** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

**Date: 06.09.2024**

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

**Name, Surname:** **İrem Selin Deniz**

**Signature :**

# ABSTRACT

**AN INVESTIGATION OF ISSUE LABELING IN OPEN SOURCE SOFTWARE PROJECTS USING LARGE LANGUAGE MODELS**

Deniz, İrem Selin

M.S., Department of Information Systems

Supervisor: Assist. Prof. Dr. Özden Özcan Top

Co-Supervisor: Prof. Dr. Altan Koçyiğit

September 2024, 109 pages

In the evolving landscape of open source software projects, effective issue management remains a pivotal aspect of sustaining project success. Issue reports provide valuable information, as they are created for reporting bugs, requesting new features, or asking questions about a software product. The high number of issue reports, which vary widely in quality, requires accurate issue classification mechanisms to prioritize work and manage resources effectively. Properly assigned issue labels are crucial for effective project management and for the reliability of research conducted to improve issue management, as such research often assumes the assigned issue labels as the ground truth. This study aims to assess the reliability of the assigned issue labels in open source software development projects to improve issue management processes. The research involves collecting two datasets of issue reports from open source software development projects hosted on GitHub. Experiments were conducted with state-of-the-art large language models for issue label classification. Furthermore, a qualitative analysis was performed to evaluate the relevance of the assigned issue labels with respect to the content of the issue reports. The empirical study performed on issue reports revealed a significant mismatch between the assigned issue labels and the actual content of the issue reports. The study also demonstrated the effectiveness of state-of-the-art large language models in classifying issue labels, while highlighting concerns about the reliability of issue labels in open source software development projects.

Keywords: issue management, issue classification, issue label, LLM, open source software

# ÖZ

## AÇIK KAYNAK KODLU YAZILIM PROJELERİNDE SORUN ETİKETLEMENİN BÜYÜK DİL MODELLERİ KULLANILARAK İNCELENMESİ

Deniz, İrem Selin

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Dr. Öğr. Üyesi. Özden Özcan Top

Ortak Tez Yöneticisi: Prof. Dr. Altan Koçyiğit

Eylül 2024, 109 sayfa

Açık kaynak kodlu yazılım projelerinin gelişen dünyasında, etkili sorun yönetimi, proje başarısını sürdürmenin temel bir unsuru olmaya devam etmektedir. Sorun raporları, yazılım ürünleri ile ilgili hataları bildirmek, yeni özellikler talep etmek veya sorular sormak amacıyla oluşturuldukları için değerli bilgiler sağlar. Kalite açısından büyük ölçüde farklılık gösteren çok sayıda sorun raporu, çalışmaları etkili bir şekilde önceliklendirmek ve kaynakları etkili bir biçimde yönetmek için doğru sorun sınıflandırma mekanizmalarını gerektirir. Doğru şekilde atanan sorun etiketleri, etkili proje yönetimi ve sorun yönetimini geliştirmek amacıyla gerçekleştirilen araştırmaların güvenilirliği açısından kritik öneme sahiptir, çünkü bu araştırmalar genellikle atanmış olan sorun etiketlerini gerçek referans değer olarak varsayar. Bu çalışma, sorun yönetim süreçlerini iyileştirmek için açık kaynak kodlu yazılım geliştirme projelerindeki atanan sorun etiketlerinin güvenilirliğini değerlendirmeyi amaçlamaktadır. Araştırma, GitHub'da bulunan açık kaynak kodlu yazılım geliştirme projelerinden iki sorun raporu veri kümesi toplanmasını içermektedir. Sorun etiketi sınıflandırması kapsamında en gelişkin büyük dil modelleri ile deneyler gerçekleştirilmiştir. Ayrıca, atanan sorun etiketlerinin, sorun raporlarının içeriği açısından ilgisini değerlendirmek için nitel bir analiz yapılmıştır. Sorun raporları üzerinde gerçekleştirilen deneysel çalışma, atanan sorun etiketleri ile sorun raporlarının asıl içeriği arasında önemli bir uyumsuzluk olduğunu ortaya koymuştur. Çalışma ayrıca, en gelişkin büyük dil modellerinin sorun etiketlerini sınıflandırmadaki etkinliğini gösterirken açık kaynak kodlu yazılım geliştirme projelerinde sorun etiketlerinin güvenilirliğine ilişkin endişeleri vurgulamıştır.

Anahtar Kelimeler: sorun yönetimi, sorun sınıflandırma, sorun etiketi, büyük dil modeli, açık kaynak kodlu yazılım

To my beloved mother and father,
Hülya Çağlar and Alper Çağlar

# ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my thesis supervisors, Assist. Prof. Dr. Özden Özcan Top and Prof. Dr. Altan Koçyiğit, for their unwavering support, valuable guidance, and encouragement throughout my master's studies.

I would like to thank my thesis committee members, Prof. Dr. Tuğba Taşkaya Temizel and Assoc. Prof. Dr. Ayça Kolukısa, for their valuable recommendations on this study.

I would like to express my sincere gratitude to my mother and father, Hülya Çağlar and Alper Çağlar. They have gifted me the most cherished and precious childhood memories a child could ever ask for. I am profoundly grateful for their unconditional love, support, and belief in me, which have been the foundation of my strength and perseverance. I am extremely proud and lucky to be their little daughter. This study is dedicated to their loving memory from the very beginning.

I would like to thank my little cat-daughter, Roma, for simply being herself. With her soft, gray-white fur and bright, curious, green eyes, she spreads an irresistible charm that never fails to melt my heart. Her playful attacks and gentle purring provide a sense of joy and calmness, instantly soothing away any traces of stress or tension. Despite having the option to sleep in our warm bed, she chose to stay by my side during long, late study hours, offering unwavering companionship during my master's studies. She always kept an eye on me to check how my studies were going and if I needed any help.

I would like to express my heartfelt gratitude to the love of my life, Burak Deniz. He was always there to cheer me up and make me a cup of coffee whenever I needed it while studying. Thank you for being my best friend through the ups and downs of this adventure we call life.

I am grateful to Mehmet Savaş Çengel, whose invaluable guidance, support, and mentorship have played a pivotal role in shaping both my personal and professional growth over the past five years.

# TABLE OF CONTENTS

ABSTRACT . . . . . iv

ÖZ . . . . . v

DEDICATION . . . . . vi

ACKNOWLEDGMENTS . . . . . vii

TABLE OF CONTENTS . . . . . viii

LIST OF TABLES . . . . . xii

LIST OF FIGURES . . . . . xv

LIST OF ABBREVIATIONS . . . . . xvi

CHAPTERS

1 INTRODUCTION . . . . . 1

1.1 Objectives and Scope . . . . . 3

1.2 Research Questions . . . . . 3

1.3 Research Methodology . . . . . 3

1.4 Organization of the Thesis . . . . . 4

2 BACKGROUND AND RELATED WORK . . . . . 5

2.1 Open Source Software Development . . . . . 5

2.2 GitHub . . . . . 6

2.3 Deep Learning for Natural Language Processing . . . . . 11
2.3.1 Traditional Deep Learning Models . . . . . 11
2.3.2 Transformer Model . . . . . 13
2.3.3 Large Language Models . . . . . 15
2.3.3.1 BERT . . . . . 15
2.3.3.2 RoBERTa . . . . . 16
2.3.3.3 Llama 3 . . . . . 17
2.4 Related Work . . . . . 17
3 METHODOLOGY . . . . . 23
4 EXPERIMENTS ON ISSUE LABEL CLASSIFICATION . . . . . 27
4.1 Datasets . . . . . 27
4.1.1 NLBSE'22 Dataset . . . . . 27
4.1.2 GitHub'23 Dataset . . . . . 28
4.2 Label Classification . . . . . 32
4.2.1 Label Classification with RoBERTa . . . . . 32
4.2.2 Label Classification with Llama 3 . . . . . 34
4.3 Reliability Analysis of Labels . . . . . 36
5 EXPERIMENTAL RESULTS . . . . . 39
5.1 Label Classification . . . . . 39
5.1.1 Label Classification with RoBERTa . . . . . 39
5.1.2 Label Classification with Llama 3 . . . . . 40
5.1.2.1 Label Classification . . . . . 42
5.1.2.2 Classification Confidence . . . . . 45

5.1.2.3 Issue Understandability . . . . . 48
5.2 Reliability Analysis of Labels . . . . . 51
6 DISCUSSION . . . . . 65
7 CONCLUSION AND FUTURE WORK . . . . . 69
7.1 Contributions . . . . . 69
7.2 Implications for Theory . . . . . 70
7.3 Implications for Practice . . . . . 70
7.4 Threats to Validity . . . . . 71
7.5 Future Research . . . . . 72
REFERENCES . . . . . 75
APPENDICES
A LISTS OF REPOSITORIES . . . . . 81
A.1 Initial List of Repositories . . . . . 81
A.2 List of Repositories Included in the GitHub'23 Dataset . . . . . 83
B LABEL MAPPING RULE SETS . . . . . 85
B.1 Rule Set for Round 1 . . . . . 85
B.2 Rule Set for Round 2 . . . . . 88
B.3 Rule Set for Round 3 . . . . . 91
C POST-PROCESSING APPLIED TO THE RESPONSES OF THE LLAMA 3 MODELS . . . . 93
C.1 Corrected Responses of Llama 3 8B on the Random Sample of the NLBSE'22 Test Set 93
C.2 Corrected Responses of Llama 3 70B on the Random Sample of the NLBSE'22 Test Set 95
C.3 Summarized Issue Reports in the Random Sample of the NLBSE'22 Test Set . . . . . 98

C.4 Corrected Responses of Llama 3 8B on the Random Sample of the GitHub'23 Test Set 99

C.5 Corrected Responses of Llama 3 70B on the Random Sample of the GitHub'23 Test Set 102

C.6 Summarized Issue Reports in the Random Sample of the GitHub'23 Test Set . . . . . 105

D DISTRIBUTION OF THE ASSIGNED AND PREDICTED LABELS . . . . . 107

# LIST OF TABLES


Table 1 Data fields of a repository on GitHub. . . . . 6

Table 2 Data fields of an issue on GitHub. . . . . 9

Table 3 Issue author association types on GitHub. . . . . 10

Table 4 Default issue labels on GitHub. . . . . 10

Table 5 Performance of the models experimented on the NLBSE'22 dataset. . . . . 20

Table 6 The number of distinct labels in each set in the NLBSE'22 dataset. . . . . 27

Table 7 Characteristics of the repositories in the initial list. . . . . 29

Table 8 Characteristics of the repositories satisfying the thresholds. . . . . 29

Table 9 Distribution of distinct issue labels after Round 1. . . . . 30

Table 10 Distribution of distinct issue labels after Round 2. . . . . 31

Table 11 Distribution of distinct issue labels after Round 3. . . . . 31

Table 12 The number of distinct labels in each set in the GitHub'23 dataset. . . . . 31

Table 13 The number of distinct labels in each set in the NLBSE'22 dataset after removing issue reports from the training set (to prevent data leakage). . . . . 34

Table 14 The number of distinct labels in each set in the NLBSE'22 dataset after removing duplicate issue reports from the training set. . . . . 34

Table 15 The number of distinct labels in the random sample of the NLBSE'22 test set. . . . . 35

Table 16 The number of distinct labels in the random sample of the GitHub'23 test set. . . . . 35

Table 17 Performance of RoBERTa on datasets. . . . . 40

Table 18 Confusion matrix of RoBERTa on the NLBSE'22 dataset. . . . . 40

Table 19 Confusion matrix of RoBERTa on the GitHub'23 dataset. . . . . 40

Table 20 Performance of the models on the random sample of the NLBSE'22 test set. . . . . 43

Table 21 Confusion matrix of RoBERTa on the random sample of the NLBSE'22 test set. . . . . 43

Table 22 Confusion matrix of Llama 3 8B on the random sample of the NLBSE'22 test set. . . . 43

Table 23 Confusion matrix of Llama 3 70B on the random sample of the NLBSE'22 test set. . . 43

Table 24 Performance of the models on the random sample of the GitHub'23 test set. . . . . 44

Table 25 Confusion matrix of RoBERTa on the random sample of the GitHub'23 test set. . . . . 44

Table 26 Confusion matrix of Llama 3 8B on the random sample of the GitHub'23 test set. . . . 45

Table 27 Confusion matrix of Llama 3 70B on the random sample of the GitHub'23 test set. . . 45

Table 28 Mean confidence levels by the Llama 3 models on random samples. . . . . 46

Table 29 Mean understandability levels by the Llama 3 models on random samples. . . . . 49

Table 30 The distribution of the issue author association types in unclear issue reports ($U$). . . . 53

Table 31 Confusion matrix of the questionable issue reports ($Q$). . . . . 54

Table 32 The number of distinct labels in plausible issue reports ($P$). . . . . 59

Table 33 Performance of the models on plausible issue reports ($P$). . . . . 60

Table 34 Confusion matrix of RoBERTa on plausible issue reports ($P$). . . . . 60

Table 35 Confusion matrix of Llama 3 8B on plausible issue reports ($P$). . . . . 60

Table 36 Confusion matrix of Llama 3 70B on plausible issue reports ($P$). . . . . 60

Table 37 The distribution of the issue author association types in plausible ($P$) and questionable ($Q$) issue reports. . . . . 63

Table 38 The repositories that are in the initial list. . . . . 81

Table 39 The repositories included in the GitHub'23 dataset. . . . . 83

Table 40 Rule Set for Round 1. . . . . 85

Table 41 Rule Set for Round 2. . . . . 88

Table 42 Rule Set for Round 3. . . . . 91

Table 43 Classification of labels by Llama 3 8B on the random sample of the NLBSE'22 test set. 94

Table 44 Confidence levels by Llama 3 8B on the random sample of the NLBSE'22 test set. . . 94

Table 45 Understandability levels by Llama 3 8B on the random sample of the NLBSE'22 test set. . . . . 95

Table 46 Classification of labels by Llama 3 70B on the random sample of the NLBSE'22 test set. . . . . 97

Table 47 Confidence levels by Llama 3 70B on the random sample of the NLBSE'22 test set. . 97

Table 48 Understandability levels by Llama 3 70B on the random sample of the NLBSE'22 test set. . . . . 98

Table 49 The indexes of the summarized issue reports in the random sample of the NLBSE'22 test set. . . . . 98

Table 50 Classification of labels by Llama 3 8B on the random sample of the GitHub'23 test set. 100

Table 51 Confidence levels by Llama 3 8B on the random sample of the GitHub'23 test set. . . . 101

Table 52 Understandability levels by Llama 3 8B on the random sample of the GitHub'23 test set. . . . . 101

Table 53 Classification of labels by Llama 3 70B on the random sample of the GitHub'23 test set. . . . . 104

Table 54 Confidence levels by Llama 3 70B on the random sample of the GitHub'23 test set. . . 104

Table 55 Understandability levels by Llama 3 70B on the random sample of the GitHub'23 test set. . . . . 104

Table 56 The indexes of the summarized issue reports in the random sample of the GitHub'23 test set. . . . . 105

Table 57 Combinations of the assigned and predicted labels. . . . . 107

# LIST OF FIGURES


Figure 1 Encoder and decoder of the transformer. . . . . 14

Figure 2 Overview of the research approach. . . . . 25

Figure 3 The prompt provided to Llama 3. . . . . 36

Figure 4 Confidence levels by the Llama 3 models on random samples. . . . . 45

Figure 5 Understandability levels by the Llama 3 models on random samples. . . . . 48

Figure 6 An issue report labeled as enhancement, manually classified as unclear. . . . . 53

Figure 7 An issue report labeled as bug, manually classified as unclear. . . . . 54

Figure 8 An issue report labeled as bug, manually classified as enhancement. . . . . 57

Figure 9 An issue report labeled as enhancement, manually classified as bug. . . . . 57

Figure 10 An issue report labeled as question, manually classified as bug. . . . . 58

Figure 11 The answer of the contributor to the issue depicted in Figure 10. . . . . 58

Figure 12 An issue report labeled as question, manually classified as enhancement. . . . . 59

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| BERT | Bidirectional Encoder Representations from Transformers |
| BiLSTM | Bidirectional Long Short-Term Memory |
| CNN | Convolutional Neural Network |
| GPT | Generative Pre-trained Transformer |
| GPU | Graphics Processing Unit |
| GRU | Gated Recurrent Unit |
| Llama | Large Language Model Meta AI |
| LLM | Large Language Model |
| LSTM | Long Short-Term Memory |
| MLM | Masked Language Modeling |
| MLP | Multilayer Perceptron |
| NLBSE | Natural Language-Based Software Engineering |
| NLP | Natural Language Processing |
| NSP | Next Sentence Prediction |
| OSS | Open Source Software |
| RCNN | Recurrent Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| RoBERTa | Robustly Optimized BERT Pretraining Approach |
| RoPE | Rotary Positional Embedding |
| SVM | Support Vector Machine |
| T5 | Text-to-Text Transfer Transformer |
| TPU | Tensor Processing Unit |

# CHAPTER 1

# INTRODUCTION

Open source software (OSS) refers to the software released under a license that allows for the inspection, use, modification, and redistribution of its source code [1]. It is developed through the collaborative sharing of knowledge among teams of geographically and organizationally distributed OSS community members [2]. OSS development has demonstrated its effectiveness with the success of many projects, including the Linux operating system, the Apache HTTP Server, and Mozilla Firefox [1]. Over time, OSS development has evolved into a worldwide software ecosystem, fostering collaboration between individuals and industry partners [3]. To this extent, various platforms have emerged to support the distributed and collaborative environment of OSS development for developers and users, such as GitHub, GitLab, and Bitbucket. For example, GitHub is one of the most widely used OSS development platforms, hosting over 100 million users and over 420 million repositories[1] as of 2024.

In the distributed and collaborative environment of OSS development, the complexity of modern software systems has been growing fast over the last decade. During software development and maintenance activities, a large number of OSS community members work on the software product in a collaborative manner [2], mostly voluntarily, with different motivations such as contributing to society by sharing knowledge and improving their skills [3]. In this setting, the utilization of issue tracking systems has been an indispensable part of software development and maintenance activities that contribute to the success of projects [4].

Issue tracking systems enable users to contribute to the progress and evolution of OSS development projects by means of issue reports. Users can create issue reports for a variety of reasons, including reporting encountered bugs, requesting new features to be implemented, or asking questions about the software product. An issue report typically includes a title for the user to briefly explain the reason for creating the issue report and an optional body section for providing more detailed information. For example, if an issue report is created for a bug, the issue title and the issue body are expected to include details such as the software version in which the bug is encountered, the observed (unexpected) behavior of the software, the steps to reproduce the misbehavior, and the normal (expected) behavior. If the issue report is related to a requested feature enhancement, the user is expected to describe the desired behavior in the issue report. An issue report is typically assigned to a developer responsible for addressing the issue report. Additionally, users can make contributions by adding comments and attachments to the issue reports created by others, sharing their knowledge with developers and users. Hence, issue reports provide valuable information to the OSS community [5].

[1] https://github.com/about

In OSS development platforms, the process of creating an issue report is often designed to be straightforward. For example, on GitHub, a user can create an issue report simply by describing the subject in the issue title and optionally explaining details in the issue body. Large-scale repositories generally share contribution guidelines with the users, explaining the desired structure they expect to have in issue reports and the rules that should be followed to make issue management more effective [6].

While this permissive approach aims to lower the barrier for creating issue reports, it can also lead to certain drawbacks. Users may tend to create issue reports having minimal information, resulting in low-quality issue reports that are difficult for developers to interpret or act upon. Since not every user has the same level of experience and knowledge about the project, the information they provide in issue reports may be of different levels. Although repositories share guidelines, users may not notice them or may skip them intentionally rather than spend time reading, leading to issue reports being created with incorrect or insufficient information or even in the wrong repository. In fact, user-submitted issue reports tend to vary widely in their quality [7, 8], and combined with the high number of issue reports created, it becomes increasingly difficult for software developers to handle these issue reports manually. Issue management becomes more challenging for project maintainers as a time-consuming task requiring manual effort. OSS community members, who work mostly voluntarily, often face imprecise issue reports and must spend additional effort to understand and clarify them [6].

To facilitate issue management, many OSS development platforms incorporate labeling mechanisms, allowing issue reports to be categorized with various labels indicating their types (such as bug, enhancement, and so on). This mechanism enables developers and users to easily mark and manage issue reports, providing immediate insights into the nature of the issues based on their labels [9].

In this context, the accurate assignment of issue labels is crucial for several reasons. First, developers utilize issue labels to identify and prioritize different types of issue reports. For example, they may prioritize issue reports related to bugs. Since these bugs reflect problems encountered by users, developers may prioritize immediate action to resolve them. In addition to bugs, developers may prefer to focus on issue reports related to new feature requests. Overall, issue labels help indicate the type of potential work that needs to be addressed by developers.

Second, issue labels provide valuable information for project management purposes. Members of the OSS community and industry partners can use these labels as historical data to estimate effort, resources, budget, and time. The current distribution of issue labels in a project can guide developers in planning future releases. Similarly, this information can serve as a foundation for planning new projects with similar characteristics [10, 11].

Moreover, research conducted to improve issue management processes often assumes that issue labels are accurate. As highlighted by Zeller [12], relying on the data stored in software repositories without assessing its quality may lead to misleading results and misinterpretations. Consequently, incorrect assignment of issue labels can undermine the reliability of these studies, since they consider incorrect issue labels as the ground truth. Although labeling positively impacts the effectiveness of issue processing, assigning labels to issue reports manually is a labor-intensive and time-consuming task for developers and users [11]. Hence, previous studies presented several approaches to automatically classify issue labels based on historical data [5, 13, 14, 15]. However, if there are mislabeled issue reports occurring frequently and systematically in datasets, they would introduce *epistemic uncertainty* in developed machine learning models, thereby threatening the external validity of any study that builds on such data.

### 1.1 Objectives and Scope

Although the accurate assignment of issue labels is crucial for effective issue management, only a few studies have investigated the problem of mislabeling. Existing research has primarily focused on issue reports from more traditional issue tracking systems, namely Bugzilla and Jira [16, 17, 18], with no studies specifically addressing the reliability of issue labels on GitHub. Hence, this study aims to evaluate the reliability of issue labels in recent OSS development projects. The scope of the study includes OSS development projects hosted on GitHub due to its widespread use in the OSS community, with a specific focus on the issue reports created within these projects. We experimented with state-of-the-art open source large language models (LLMs) to analyze the content of the issue reports. There are two main reasons for this approach: (1) the issue content consists of rich contextual information provided by the members of the OSS community, and (2) the recent advancements in open source LLMs have enabled extracting contextual information from text data while capturing nuances, surpassing the capabilities of traditional non-contextual natural language processing (NLP) algorithms.

### 1.2 Research Questions

This thesis aims to answer the following research questions:

> **RQ1.** To what extent are the labels of the issue reports created in OSS development projects reliable?
>
> In this research question, the aim is to explore to what extent the assigned labels of the issue reports are indeed relevant to the content of the issue reports.
>
> **RQ2.** To what extent can the label of a newly created issue report in an OSS development project be predicted using state-of-the-art LLMs?
>
> This research question investigates the effectiveness of state-of-the-art LLMs in issue label classification.

### 1.3 Research Methodology

A mixed-methods research approach was employed in this study, integrating both quantitative and qualitative research techniques to provide a comprehensive analysis designed to answer the research questions. The following steps were undertaken:

> **1. Data Collection.** Since GitHub is one of the famous OSS development platforms, the OSS development projects hosted on GitHub were included in the study. We specifically focused on the issue reports that were associated with an encountered bug, an enhancement request, or just a question based on their issue labels. We chose these labels since they are included by default in each repository once it is created on GitHub, and they are the three most commonly used issue labels in OSS development projects hosted on GitHub [9]. Two different datasets were included in the study: (1) a dataset used in the Natural Language-Based Software Engineering

(NLBSE) tool competition organized in 2022 [15], and (2) a dataset formed in the scope of this thesis study using the REST API provided by GitHub. These datasets contain the following data fields of issue reports: issue title, issue body, issue label, timestamp of creation, issue number, repository name, and the association of the issue author.

**2. Label Classification.** We performed experiments with three state-of-the-art open source LLMs: RoBERTa [19], and Llama 3, including its 8B and 70B variants [20], for issue label classification based on the content of the issue reports. These models were selected to investigate two types of experiments. In one setting, the pre-trained RoBERTa model was fine-tuned for the downstream task of issue label classification with labeled data in a supervised manner. This means that the RoBERTa model could potentially learn from incorrectly assigned issue labels, possibly adopting and reinforcing these errors in its predictions. However, no such risk was associated with the classifications performed by the Llama 3 model, as it was used only for inference, without training or fine-tuning. The performances of the models were evaluated and compared based on determined performance metrics. In addition, the Llama 3 model was requested not only to perform issue label classification but also to provide its confidence levels in making such classifications and to assess the understandability levels of the provided issues.

**3. Reliability Analysis.** For the reliability analysis of issue labels, the issue title and the issue body were considered, as they describe the content of an issue report written by its author. We investigated to what extent the assigned labels of the issue reports are relevant to their contents. To this end, we focused on the random samples taken from the test sets, as it was infeasible to manually classify issue reports based on their contents due to the orders of magnitude of the number of issue reports. First, we checked whether each random sample was representative of its larger population. After ensuring that, we manually classified the issue reports for which at least one of the three models classified the issue with a label different from the assigned one. We performed this classification relying only on the issue content, involving the issue title and the issue body. The results of the manual classification were compared with the assigned issue labels. Furthermore, we analyzed the confidence and understandability levels associated with the issues that were evaluated by the Llama 3 model.

### 1.4 Organization of the Thesis

The rest of the thesis is organized as follows:

In Chapter 2, key concepts related to OSS development, GitHub, and LLMs are introduced, and related work is presented. In Chapter 3, the research methodology followed in this thesis study is described. In Chapter 4, details of the datasets used and the design of experiments are provided. In Chapter 5, the results of the experiments are reported. In Chapter 6, the findings of the study are discussed. In Chapter 7, the contributions of the study, its implications for theory and practice, and threats to validity are described, and potential directions for future research are suggested.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

This section provides a concise overview of OSS development and GitHub. Next, key concepts related to LLMs are introduced. Finally, related work is presented.

## 2.1 Open Source Software Development

OSS development is a term that emerged towards the end of the 20$^{th}$ century, referring to the software released under a license that allows for the inspection, use, modification, and redistribution of its source code [1]. As a distinctive characteristic that sets it apart from proprietary software development, it corresponds to the development of software through the collaborative sharing of knowledge among teams of geographically and organizationally distributed OSS community members, which has been described as community-based development [2]. OSS community members usually contribute to projects as volunteers with different motivations. Their motivations include committing to the OSS ideology of making software accessible and free, contributing to society by sharing knowledge with others, improving their skills, and achieving professional growth. The success of OSS development is demonstrated by numerous successful projects, including the Linux operating system, the Apache HTTP Server, and Mozilla Firefox [1]. Following these prominent examples, OSS development has evolved into a worldwide software ecosystem, involving industry partners and collaboration between companies and individuals [3].

In this distributed and collaborative environment, some characteristics of OSS development differ from traditional software development processes. In OSS development projects, users of the software product usually request new features by describing their needs while the project is evolving. There is no formal requirements elicitation process or software requirements specification. New feature requests are decided to be implemented or not by the OSS community, including both developers and users. Conversely, in traditional software development, users are generally not involved in this decision-making process. In OSS development, both developers and users may contribute to and participate in the discussion of design decisions. In terms of implementation, developers choose the implementation they want to work on based on their preferences and skills. Both developers and users may contribute to the project's source code, documentation, and related artifacts [21].

In the last decade, various platforms have emerged to support the distributed and collaborative environment of OSS development for developers and users, such as GitHub, GitLab, and Bitbucket. The following section presents an overview of GitHub, the chosen platform for this study due to its widespread use in OSS development.

### 2.2 GitHub

GitHub[2] is one of the most widely used OSS development platforms, hosting over 100 million users and over 420 million repositories[3] as of 2024. It provides a version control mechanism and an issue tracking system to its users for software development and maintenance activities. It enables its users to produce new code, data, and files, modify existing ones, report bugs, request new features, and collaborate with each other.

A repository can be considered the most basic element of GitHub. It contains all of the code artifacts, data, and related files, as well as each file's revision history, including who made what changes and when. A repository on GitHub may be either public or private. Public repositories can be accessed by any GitHub user, but a user needs to be authorized to access a private repository. Any GitHub user with appropriate permissions can fork a repository to their local environment to make further changes independently, without affecting the original project. To communicate and collaborate with each other, users can create issue reports in repositories to report bugs, request new features, or ask questions about the software product. A repository may contain data fields explained in Table 1[4].

Table 1: Data fields of a repository on GitHub.

| **Data Field** | **Description** |
|---|---|
| Title | The title of the repository is a mandatory field that serves as the primary identifier for the repository, reflecting its purpose or content. |
| Description | The description of the repository briefly explains its objective. It is an optional field provided under the "About" heading. |
| Topics | A topic of the repository indicates a particular subject area of the repository. It is an optional field. A repository may have none, one, or multiple topics. |
| README File | The README file of the repository is a text file that contains summary information about the purpose and content of the repository. It is typically the first file a visitor to the repository would take a look at. |
| Files and Folders | The files and folders in the repository are related to its content and structure. A repository may contain related files located under corresponding folders in an organized manner. A repository on GitHub may contain source code, binary code, associated files, or any combination of these. |
| Date of Creation | The creation date of the repository. It is maintained by GitHub. |
| Date of Last Modified | The last modification date of the repository. It is maintained by GitHub. |

[2] https://github.com
[3] https://github.com/about
[4] https://docs.github.com/en/get-started/quickstart/github-glossary#repository

Table 1 (cont.)

| Data Field | Description |
|---|---|
| Languages | The languages section provides a distribution of programming languages used in the repository, along with the percentage of each language. GitHub employs the `Linguist` library, an OSS, to detect the languages used in a repository. The languages are identified based on the files and directories within the repository and updated after any change is pushed to the repository's default branch. GitHub uses these languages for syntax highlighting[5]. |
| Issues | An issue represents a reported bug, requested new feature, or other kinds of important statements related to the repository. In public repositories, any user can create an issue. However, only authorized users can create issues in private repositories. Issues are handled by OSS community members. Each issue has its own discussion thread where users can add comments and attachments. GitHub employs two issue states: open and closed. An issue can be categorized with labels and can be assigned to someone. |
| Pull Requests | A pull request is a request to merge one or more commits into one of the branches of a repository. It is created by a user when they do not have write permission but are proposing changes to be made in the repository. In this case, one of the collaborators of the repository reviews the proposed changes and either accepts or rejects the pull request. Each pull request has its own discussion thread, as in the case of issues. A pull request may be created to address a specific issue, and merging the pull request may result in that issue being marked as closed. GitHub's REST API considers every pull request as an issue, but not vice versa. |
| Releases | A release corresponds to a deployable software iteration. It is created to package the software along with its release notes, release tags, and links to binary files for others to use. Users with read access to a repository can view and compare different releases, but only users with write access can manage them. |
| Commits | A commit corresponds to some changes made in a repository. When a commit is made to save these changes, a unique ID, also known as the "SHA" or "hash", is generated by GitHub. This unique ID allows users to track the specific changes, along with who made them and when. A commit typically includes a commit message that briefly explains what changes were made and why. |

[5] https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-repository-languages

Table 1 (cont.)

| Data Field | Description |
|---|---|
| Stars | The stars information corresponds to the number of stars a repository has received from GitHub users. It is considered a display of appreciation for the repository and a way to rank the popularity of different repositories. When a user stars a repository, the repository is placed in the user's favorites group. In this way, finding the repository later becomes easier for the user. In addition, GitHub recommends other repositories with similar content to those starred by the user, helping the user discover related content more quickly. |
| Watchers | The watchers information corresponds to the number of users who are watching the repository. When a user watches a repository, a notification is sent to the user when there is any new activity in that repository. |
| Forks | The forks information corresponds to the number of repositories that have forked the original repository. Forking allows users to create a local copy of the repository in their GitHub accounts as a new repository, enabling them to make changes without affecting the original. Additionally, they may create a pull request to propose applying these changes to the original repository. The original repository is updated once the pull request is accepted by one of the collaborators. |
| Collaborators | The collaborators of the repository are the users who have been invited by the repository's owner to collaborate on the repository. A repository may have none, one, or multiple collaborators. |
| Contributors | The contributors of the repository are the users who have contributed to the repository by creating a pull request that was accepted and merged into the repository. A repository may have none, one, or multiple contributors. |
| Tags | A tag marks a specific commit at a point in the repository's history. When a particular commit is tagged, the tag corresponds to all the changes made before that commit in the repository. Tags can be used to compare the differences between different points in the repository's history and are commonly employed to mark release versions, with the release names serving as tag names. |
| Branches | A branch represents an independent line of software development in a GitHub repository. Users can work on a branch that is separate from the main branch. In this way, concurrent development is allowed without affecting the main branch of the repository. Once the changes on a branch are complete, the branch can be merged into the main branch to integrate and publish those changes. |

In GitHub, issues contain valuable information as they are created to report bugs, request new features, or share other kinds of important statements about the software product. They provide the primary means of information through which GitHub developers collect feedback on the repository. An issue may contain data fields explained in Table 2[6].

Table 2: Data fields of an issue on GitHub.

| **Data Field** | **Description** |
|---|---|
| Issue Number | In any repository on GitHub, each issue has a unique number, maintained by GitHub. |
| Title | The title of the issue is a mandatory field for the user to briefly explain the reason for creating the issue. |
| Body | The body of the issue is an optional field for the user to provide detailed information about why the issue is created. If the issue is related to an encountered bug, the issue body is expected to include details such as the software version in which the bug is encountered, the observed (unexpected) behavior of the software, the steps to reproduce the misbehavior, and the normal (expected) behavior. If the issue is related to a requested new feature, the user is expected to describe the desired behavior in the issue body. Although it is an optional field, it contains important information for OSS community members. |
| Assignee | The assignee of an issue is a user who is responsible for addressing the issue. In public repositories and in private repositories with a paid account, an issue may have up to ten users assigned. In private repositories on the free plan, only one user may be assigned to an issue. |
| Comments | Any user with write access to the repository in which the issue is created may add a comment to the discussion thread of the issue to provide feedback. |
| State | GitHub employs two issue states: open and closed. |
| Date of Creation | The creation date of the issue. It is maintained by GitHub. |
| Date of Last Modified | The last modification date of the issue. It is maintained by GitHub. |
| Date of Closure | The closure date of the issue. It is maintained by GitHub. |
| Labels | Issues and pull requests can be classified with labels on GitHub. An issue may have none, one, or multiple labels. |
| Projects | The project with which the issue is associated. An issue may be associated with none, one, or multiple projects. |
| Milestone | The milestone with which the issue is associated. An issue may be associated with none or a single milestone. GitHub does not support associating an issue with multiple milestones, as milestones are intended to represent specific points in time during development and maintenance activities. |
| Development | The development section displays branches and pull requests linked to the issue. |

[6] https://docs.github.com/en/issues/tracking-your-work-with-issues

The REST API of GitHub[7] offers a structured set of endpoints to interact with various GitHub features. For example, it provides endpoints for retrieving issue reports in a repository, creating and updating issue reports, and handling pull requests. In addition to the data fields provided in Table 2, the GitHub REST API stores the association of the issue author, indicating the relationship between the user who created the issue and the repository in which the issue was created. There are eight association types explained in Table 3[8].

Table 3: Issue author association types on GitHub.

| **Association** | **Description** |
|---|---|
| COLLABORATOR | The author has been invited to collaborate on the repository. |
| CONTRIBUTOR | The author has previously committed to the repository. |
| FIRST_TIMER | The author has not previously committed to GitHub. |
| FIRST_TIME_CONTRIBUTOR | The author has not previously committed to the repository. |
| MANNEQUIN | The author is a placeholder for an unclaimed user. |
| MEMBER | The author is a member of the organization that owns the repository. |
| NONE | The author has no association with the repository. |
| OWNER | The author is the owner of the repository. |

GitHub enables users to classify issue reports using different labels. The default issue labels provided by GitHub, explained in Table 4[9], are defined in a repository once it is created on GitHub. However, a user with write access may delete the default issue labels and/or define additional ones in the repository.

Table 4: Default issue labels on GitHub.

| **Label** | **Description** |
|---|---|
| BUG | Indicates an unexpected behavior of the software. |
| DOCUMENTATION | Indicates a need for improvements or additions to documentation. |
| DUPLICATE | Indicates that the issue is a duplicate of an existing one. |
| ENHANCEMENT | Indicates a new feature request. |
| GOOD FIRST ISSUE | Indicates that the issue is a good one for first-time contributors. |
| HELP WANTED | Indicates that the assignee needs help on the issue. |
| INVALID | Indicates that the issue is no longer relevant. |
| QUESTION | Indicates that more information on the issue is needed. |
| WONTFIX | Indicates that the work on the issue will not continue. |

[7] https://docs.github.com/en/rest?apiVersion=2022-11-28
[8] https://docs.github.com/en/graphql/reference/enums
[9] https://help.github.com/articles/about-labels/

## 2.3 Deep Learning for Natural Language Processing

Deep learning is a specialized field within machine learning that uses artificial neural networks with multiple layers to learn and extract hierarchical features from large datasets automatically. This section first summarizes traditional deep learning models that have been used in the NLP domain. Next, it introduces the transformer model and the state-of-the-art open source LLMs that are utilized in this thesis study. Finally, it concludes by explaining the motivation for selecting these models in this thesis.

### 2.3.1 Traditional Deep Learning Models

Before the evolution of the transformer model, most research in the NLP area focused on deep learning models such as Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Recurrent Convolutional Neural Networks (RCNNs), Long Short-Term Memory (LSTM) networks, Bidirectional LSTM (BiLSTM) networks, Gated Recurrent Units (GRUs), and sequence-to-sequence and attention-based sequence-to-sequence models [22]. These models are summarized below:

MLPs are neural networks consisting of an input layer, one or more hidden layers, and an output layer, where each layer is composed of a set of processing units known as neurons [23, 24, 25]. The MLP model is easy to understand and simple to implement. However, as it ignores the sequence information and struggles to capture the semantic relationships, it was subsequently replaced by advanced models like CNNs and RNNs for NLP tasks.

CNNs, originally developed to process images, are also explored for NLP tasks by treating text as a one-dimensional image [26, 27]. A CNN model can effectively learn local features using convolution layers, but may struggle to capture long-term dependencies. RNNs are specifically developed to process sequential data, such as text and time series [28]. An RNN model can handle inputs with varying lengths and process sequential data by maintaining a hidden state to capture the context from previous inputs. However, RNNs suffer from the vanishing gradients problem and again struggle to capture long-term dependencies.

To address the limitations of CNNs and RNNs, RCNNs are proposed, which combine the strengths of the two [29]. In RCNNs, CNNs are used to capture local features, such as n-grams in the text, while RNNs are employed to model the sequential dependencies within the data. This hybrid approach enables RCNNs to capture both local and global patterns effectively. However, they may still struggle with long-term dependencies, similar to traditional RNNs. The recurrent components can still be susceptible to problems like the vanishing gradients, making it hard to capture the relationships between distant parts of a text.

LSTM networks [30] and GRUs [31, 32] evolved as advanced RNN variants to address the problems with the RNN model. BiLSTM networks concatenate the outputs of two LSTM networks to predict each element in the sequence. The two LSTM networks have different data processing directions. One LSTM model processes the sequence from left to right and the other from right to left. So, any word in a sentence is predicted by preceding and following words. The gating mechanism in these models helps regulate the flow of information along the sequence and retain the most important features and patterns. However, RNN variants like LSTM networks and GRUs expect the input and output sequences to be

the same length, which is generally unusual in NLP tasks where the lengths of the input and output sequences often differ.

Consequently, researchers introduced the sequence-to-sequence model to handle tasks with different input and output sequence lengths [33]. The sequence-to-sequence model was initially developed for machine translation and later explored for other NLP tasks. It consists of an encoder and decoder based on RNNs or RNN variants, like LSTM networks or GRUs, to process the input sequence and generate the output sequence. In this setting, the encoder processes the input sequence to generate a fixed-size context vector based on which the decoder produces the output sequence. However, the fixed-size context vector fails to encode the entire information in the input sequence, especially when the input sequence is long [34]. The attention mechanism was introduced to address this problem, allowing the decoder to focus on the relevant input tokens at each decoding step [34, 35]. However, as the encoder and decoder of the sequence-to-sequence model are based on RNNs and RNN variants, they still suffer from the vanishing gradients problem and struggle to capture long-term dependencies.

The limitations of traditional deep learning models can be summarized as follows [36]:

**1. Lack of sequence and semantic understanding:** An MLP model takes a fixed-size input and treats each input feature independently, without considering the order in which it appears in the input. It does not consider the position of an input element relative to others. This makes MLPs effective for tasks where the input order does not matter, but not suitable for tasks where the sequence is important, like language processing. They also do not have any internal state or memory that allows them to maintain context over time. Hence, they cannot keep track of the relationships between different elements in a sequence, such as the words in a sentence. Due to these limitations, MLPs struggle with NLP tasks where the meaning of a sentence can change depending on the order of the words.

**2. High computational cost:** CNNs require a large number of parameters to perform effectively, making them resource-intensive. RCNNs are even more complex, as they integrate both convolution and recurrent layers. This added complexity makes the implementation more challenging and requires more computational resources, particularly in terms of the training time and memory usage. Similarly, LSTM networks and GRUs, which improve upon vanilla RNNs, also introduce additional complexity due to their gating mechanisms, further increasing computational costs regarding the training time and hardware resources.

**3. Vanishing gradients:** RNN models are particularly prone to the vanishing gradients problem, where gradients diminish as they propagate through the network during backpropagation. This problem reduces the ability of RNNs to learn long-term dependencies. Although LSTM networks and GRUs mitigate this problem to some degree, they still suffer from the vanishing gradients problem and struggle to capture long-term dependencies effectively.

**4. Sequential processing:** RNNs and their variants process input sequences one token at a time, inherently limiting their ability to leverage the parallel processing capabilities of modern hardware like GPUs and TPUs. This sequential nature not only slows down the training process but also makes these models less efficient, particularly when handling long sequences.

### 2.3.2 Transformer Model

RNNs and LSTM networks are widely used in sequential tasks such as next word prediction, machine translation, and text generation. However, one of the major challenges with the recurrent model is capturing long-term dependencies. To overcome this limitation, a new architecture called transformer was introduced in the paper titled *Attention Is All You Need* by Google in 2017 [37]. The transformer model is currently considered the state-of-the-art architecture in the NLP domain, enabling the development of new revolutionary models, including BERT, GPT series, and T5 [38].

The key ingredient behind the massive success of the transformer model is its self-attention mechanism. The self-attention mechanism allows the transformer model to process the input sequence without the need for recurrent or convolution layers. Compared with traditional deep learning models, the self-attention mechanism can better capture long-term dependencies in the input sequence, making the transformer model highly effective for NLP tasks.

The transformer consists of an encoder-decoder architecture depicted in Figure 1 with an example of translation from English to German.

In the encoder, there exists a stack of $N$ number of identical encoder layers that are on top of each other, where $N$ is set to 6 in the original setting [37]. Each encoder layer sends its output to the encoder layer above it. The input sentence is converted to an input embedding (embedding matrix). Then, the positional encoding, which indicates the order of the words that appear in the sentence, is added to the embedding matrix. The resulting matrix is given to the bottommost encoder layer.

Each encoder layer consists of two sub-layers: a multi-head attention (self-attention mechanism) and a feed-forward neural network. The self-attention mechanism adds contextual information to the model. While computing the representation of each word, it relates each word to all other words in the sentence to understand more about the word. This structure helps the model capture long-term dependencies better than traditional deep learning models. Instead of computing a single attention matrix, multiple attention heads are computed and concatenated to obtain more accurate results. The feed-forward neural network sub-layer introduces non-linearity to enable the model to learn and represent complex patterns in the data. It takes the attention matrix as input and returns the encoder representation as output. The add and norm component connects the input of the multi-head attention sub-layer to its output and the feed-forward neural network sub-layer's input to its output. It is basically a residual connection followed by layer normalization, where layer normalization promotes faster training by preventing the values in each layer from changing heavily. Next, the output obtained from the encoder layer is given as input to the encoder layer above it. The output of the final encoder is given as input to the decoder.

Similar to the encoder, the decoder consists of a stack of $N$ number of identical decoder layers, $N$ being 6 in the original setting [37]. As in the case of the encoder, instead of feeding the input directly to the decoder, the model converts it into an embedding matrix, adds the positional encoding to it, and then feeds it to the decoder. The output of one decoder layer is sent as input to the decoder layer above it. The encoder's representation of the input sentence is sent to all decoder layers.

Initially, the input to the decoder is the `<sos>` token, which indicates the start of the sentence. At each iteration, it tries to generate the next word in the output sentence using the current input and the

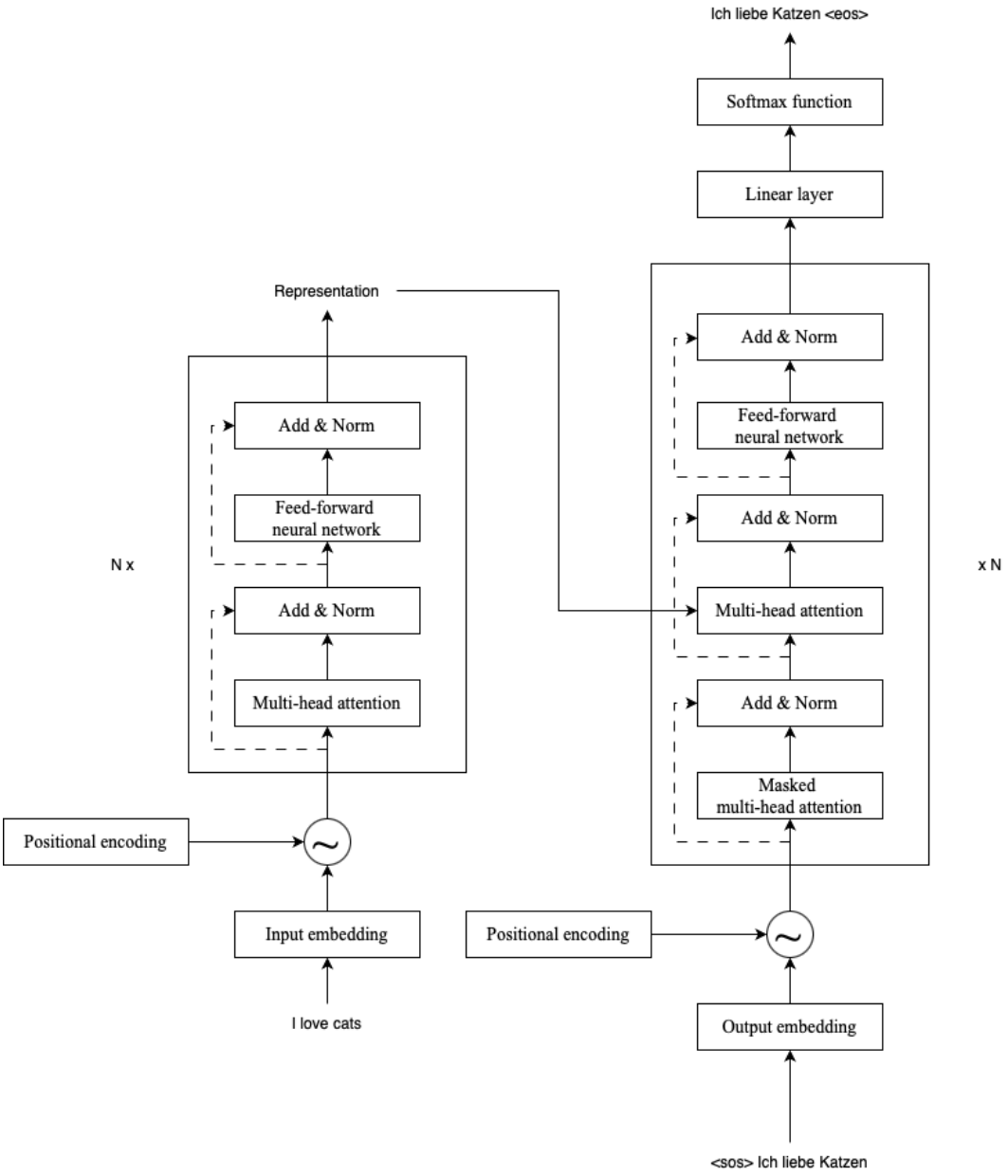


Figure 1: Encoder and decoder of the transformer.

newly generated word from the previous iteration. Once the <eos> token, which indicates the end of the sentence, is generated, it implies that the decoder has completed generating the output sentence.

Each decoder layer consists of three sub-layers: a masked multi-head attention, a multi-head attention, and a feed-forward neural network. Different from the encoder layer, there are two multi-head attention sub-layers in the decoder layer, where one of them is masked. Masked self-attention is employed in the decoder's self-attention mechanism to prevent the model from considering future words during training, ensuring that predictions for a word depend only on the already generated predictions before it. Since the entire input sentence is available in the encoder, there is no need for masking, and the

attention mechanism can consider all positions in the input sentence. The multi-head attention sub-layer in each decoder layer receives two inputs: one is from the previous decoder sub-layer (masked multi-head attention), and the other is the encoder's representation of the input sentence. The add and norm component connects the input and output of a sub-layer, as in the case of the encoder. Similarly, the feed-forward neural network sub-layer takes the attention matrix as input and returns the decoder representation as output. The output obtained from the decoder layer is given as input to the decoder layer above it.

Once the decoder learns the representation of the output sentence, the output obtained from the topmost decoder is provided as input to the linear and softmax layers. The linear layer generates the logits whose size equals the vocabulary size in the output sentence. Then, these logits are converted into a probability using the softmax function, indicating the likelihood of each word being the next word in the generated sequence. The decoder returns the word which has the highest probability value.

As the decoder predicts the probability distribution and selects the highest probability as output, the model needs to minimize the difference between the predicted and actual probability distributions. Hence, cross-entropy loss is used as the loss function, and the model is trained to minimize the loss function while using AdamW as the optimizer. To prevent over-fitting, the model applies dropout to the output of each sub-layer and also to the sum of the embeddings and the positional encodings.

### 2.3.3 Large Language Models

A large language model (LLM) is a deep learning model pre-trained on vast amounts of textual data, enabling it to comprehend and generate human-like text. It is typically based on the transformer architecture, which is an encoder-decoder model incorporating the self-attention mechanism. At its core, the transformer converts one sequence of tokens to another, making it ideal for handling natural language tasks. Being pre-trained on massive textual datasets, LLMs exhibit special abilities that allow them to achieve remarkable performance without any task-specific training in many NLP tasks [37].

#### 2.3.3.1 BERT

The Bidirectional Encoder Representations from Transformers (BERT) is a language representation model released by Google in 2018 [39]. It has created a significant breakthrough in the field of NLP by achieving superior results in many NLP tasks such as question answering, text generation, and sentence classification. One of the primary reasons for the success of BERT is that it is a context-based model, unlike other traditional embedding models, which are context-free. In addition, it is a pre-trained model that can be fine-tuned for specific downstream tasks, eliminating the need for training the model from scratch.

BERT is built on the transformer architecture [37] with some adjustments. As its name suggests, it can be considered the transformer architecture, but involving only the encoder. Similar to the transformer model, the input sentence is given to the encoder, and the encoder returns its representation as output. However, unlike the original transformer model, which is designed primarily for sequence-to-sequence tasks focusing on encoding and decoding contexts, the BERT model employs a bidirectional approach to understand the full context of a word within a sentence. The encoder of the transformer is bidirectional in nature since it can read a sentence in both directions. In this way, the BERT model captures

richer representations of words based on their surrounding words, leading to improved performance on various language understanding tasks.

Before being fed into the BERT model, the input is converted into embeddings using the three embedding layers, which are token, segment, and positional embeddings. In the token embedding layer, the `[CLS]` token is added at the beginning of the first sentence, and a `[SEP]` token is added at the end of each sentence in the input. Then, the tokens are converted into embeddings. The segment embedding layer is used to distinguish different sentences. Apart from the `[SEP]` token introduced in the token embedding layer, the segment embedding layer maps all tokens that appear in the same sentence to the same segment embedding. Finally, the positional embedding layer stores the position of tokens in the input sentence to learn the contextual meaning of the text.

The BERT model is pre-trained on the Toronto BookCorpus and Wikipedia datasets using two different tasks: masked language modeling (MLM) and next sentence prediction (NSP). In the MLM task, the model is trained to predict masked words within a sequence of words. This task can be categorized into two types: auto-regressive and auto-encoding. Auto-regressive language models involve either forward (left-to-right) or backward (right-to-left) prediction, following the sentence in only one direction. In contrast, auto-encoding language models, like BERT, leverage both forward and backward predictions, reading the sentence in both directions to make predictions. This bidirectional approach provides a more comprehensive understanding of the sentence, resulting in better performance. The NSP task is designed to be a binary classification problem in which two sentences are given to the BERT model, and it is trained to predict whether the second sentence is the next sentence of the first one or not.

The pre-trained BERT model can be fine-tuned for specific downstream tasks by adding an additional output layer without the need for training the model from scratch for each new task. Since training these models requires significant computational resources and time, this pre-training approach is particularly valuable, making BERT highly adaptable and effective for a wide range of NLP applications.

There are different variations of the BERT model, including RoBERTa [19], CodeBERT [40], ALBERT [41], and seBERT [42], to leverage BERT in specific contexts.

#### 2.3.3.2 RoBERTa

The Robustly Optimized BERT Pretraining Approach (RoBERTa) [19] is an extension of the BERT model [39] having some improvements in the pre-training steps. The RoBERTa model is pre-trained with only the MLM task, without using the NSP task employed in BERT. Dynamic masking is used during training instead of static masking in the MLM task. The dynamic masking pattern introduced in RoBERTa duplicates the training data and carries out a range of masking strategies. When data is passed into the RoBERTa model, different masking strategies are performed every single time. The BERT model, on the other hand, uses a static masking strategy and undertakes masking only during data preprocessing. The RoBERTa model is pre-trained with larger batch sizes than BERT and uses a different tokenizer. With these improvements, it achieves advanced capabilities in understanding contextual and semantic aspects of textual data.

It is important to note that RoBERTa is a pre-trained model generally fine-tuned for downstream tasks. This means that it is first initialized with the pre-trained parameters, and all parameters are fine-tuned using the labeled data from the downstream task, which is issue label classification in our case.

#### 2.3.3.3 Llama 3

The Llama series[10] has represented a significant advancement in LLMs since its launch by Meta in February 2023. It outperformed the much larger, closed-source GPT-3 model, which boasts 175 billion parameters, even with its version involving 13 billion parameters [43].

On April 18, 2024, Meta introduced the Llama 3 model, offering configurations with 8 billion (8B) and 70 billion (70B) parameters. Thanks to the extensive pre-training on more than 15 trillion data tokens, the Llama 3 model has achieved the state-of-the-art performance across a broad range of tasks, establishing the Llama 3 family as among the finest open-source LLMs available for a wide variety of applications and deployment scenarios [20].

Being built on the transformer architecture [37], it incorporates various improvements that optimize performance and scalability. To improve the training stability, it normalizes the input of each transformer sub-layer instead of normalizing the output. It employs a different non-linearity function in the feed-forward neural network sub-layer to improve performance. It removes the absolute positional encodings, and instead, uses rotary positional embeddings (RoPE), introduced by Su et al. [44], at each layer of the architecture.

In this thesis, RoBERTa [19] and Llama 3, including its 8B and 70B variants [20], were selected to conduct experiments for issue label classification. In this way, two types of experiments were performed. In one setting, the pre-trained model, RoBERTa, was fine-tuned with the dataset of the downstream task, that is issue label classification, in a supervised manner. The model was initialized with the pre-trained parameters, and all parameters were fine-tuned using the labeled data. This means that the RoBERTa model could potentially learn from incorrectly assigned issue labels, possibly adopting and reinforcing these errors in its predictions. In contrast, no such risk was associated with the predictions performed by the Llama 3 model, as it involved only inference without any training or fine-tuning.

### 2.4 Related Work

Researchers have proposed automated approaches to predict the label that should be assigned to a new issue report. However, only a few studies have investigated the quality of data used in these approaches regarding misclassified issue labels.

In 2008, Antoniol et al. [16] highlighted the problem of misclassified issue reports. They randomly selected 1,800 closed bug reports from three OSS development projects: Mozilla, Eclipse, and JBoss. These projects used Bugzilla and Jira as their issue tracking systems. Through manual classification, they observed that only 45% of the bug reports were actually related to bugs, while 38% referred to enhancements, which were not necessarily about fixing a problem but improving existing features or adding new ones. In addition, they classified 17% of the bug reports as other, indicating that these bug reports did not fit into the categories of bug or enhancement. After removing the bug reports manually classified as other, they experimented with a Naive Bayes classifier, decision trees, and logistic regression to automatically classify issue labels as bug and non-bug based on the corpus they manually created. The best performance in accuracy was achieved with logistic regression in all three

[10] https://llama.meta.com

projects, where the accuracy of the model was 77% for Mozilla bug reports and 82% for bug reports of Eclipse and JBoss.

Later, Herzig et al. [17] manually investigated 7,401 issue reports from five OSS development projects: HTTPClient, Jackrabbit, Lucene-Java, Rhino, and Tomcat5, with Rhino maintained by Mozilla and the others by Apache. Bugzilla and Jira were the issue tracking systems used in these projects, which were commonly used issue tracking systems, having the majority of the issue reports labeled as bugs. They collected all completed issue reports marked as RESOLVED, CLOSED, or VERIFIED with a resolution status of FIXED, ensuring that only finalized issue reports were included in their analysis. They found that more than 40% of the issue reports were misclassified, with 33.8% of all bug reports not referring to corrective code maintenance but something else based on their manual classification.

To address this problem, Zhou et al. [18] combined unstructured free-text data with structured data to train a classifier able to predict if a bug report is actually a bug report or another kind of issue. In the first stage of their algorithm, the free-text part of each bug report is classified into the three discrete levels of possibilities: {high, middle, low} by a Multinomial Naive Bayes classifier. In the second stage, the output of the first stage is combined with the structured features of the bug reports: {severity, priority, component, assignee, reporter, operating system, reproducibility} and fed into a Bayesian Network classifier. The performance of the proposed approach was evaluated on a dataset including more than 7,000 manually labeled random issue reports collected from ten OSS development projects: Mozilla, Eclipse, JBoss, Ant, Apache Http-2, Firefox, Lenya, TomCat5, Fedora, and OpenFOAM, which used Bugzilla, Redhat Bugzilla, and Mantis as issue tracking systems. Comparative experiments with previous work [16] (using only unstructured free-text data) on the same projects, Mozilla, Eclipse, and JBoss, achieved reasonable improvements from 77.4% to 81.7%, 76.1% to 81.6%, and 87.4% to 93.7%, respectively, over their best-performing model (logistic regression) in terms of the weighted average F1-score. Across ten projects, the performance gain obtained in each project ranged between 0.1% and 6.3%, with an average improvement gain of 4.4%.

Pandey et al. [45] experimented with different classification algorithms, namely Naive Bayes, linear discriminant analysis, k-nearest neighbors, support vector machines (SVMs), decision trees, and random forests, to classify the issue reports as bug and non-bug. They employed a subset of issue reports classified manually in previous work [17]. Specifically, they chose the issue reports extracted from Jira, leading to 5,587 issue reports from three OSS development projects: HTTPClient, Lucene, and Jackrabbit, maintained by Apache. It was observed that random forest was the best-performing model in all three projects, with the highest accuracy values of 74.8%, 83.4%, and 80.9% for HTTPClient, Lucene, and Jackrabbit, respectively.

After studies that relied solely on term frequency-based methods, such as term-document frequency, neural network-based approaches to distributional semantics, commonly known as word embeddings [46, 47], have increasingly gained attention for their ability to capture richer and more nuanced semantic relationships between words. It is important to note that these studies assumed historical issue labels as the ground truth, without performing any action, such as manual classification, to verify or refine these labels for developing automated issue label classification approaches.

Kallis et al. [13] proposed a machine learning classifier that leveraged the textual content of the issue title and the issue body, whose vectorial representations were based on fastText. The fastText is a tool that is open sourced by Facebook AI Research in 2016, which uses linear models with a rank constraint and fast loss approximation [48]. In terms of preprocessing, the title and the body of the

issues were concatenated into a single textual paragraph. Then, the resulting text was tokenized and the bag of words representation was derived. From the extracted bag of words representation, each word was represented by a vector of character n-grams, corresponding to the desired input of the fastText. The model was trained and evaluated on 30,000 issue reports randomly collected from GitHub, where 10,000 issues were labeled as bug, 10,000 as enhancement, and the remaining 10,000 as question. Each issue had a single label. The model achieved F1-score values of 83.1%, 82.3%, and 82.5% for the bug, enhancement, and question classes, respectively. When the trained model was evaluated in an unbalanced dataset of 34,000 issue reports (where 48.0% of the issues were labeled as bug, 41.8% as enhancement, and the remaining 10.2% as question), the F1-score values obtained for the bug, enhancement, and question classes were 75%, 74%, and 48%, respectively [14].

Among recent advancements, pre-trained LLMs have started to be compared with traditional deep learning models and employed for issue label classification. The performance of a pre-trained contextual model, BERT, was compared with three traditional deep learning models, namely CNNs, RCNNs, and BiLSTM networks, in multi-label issue classification by Wang et al. [49]. They specifically focused on the issue reports collected from 73 most-starred OSS development projects hosted on GitHub and trained a separate model for each project. It was observed that F1-score values achieved the highest values when $k$ equals two or three for most projects. As only 10% of the issues had three or more labels, $k$ was set to two. This means that the models recommended two possible labels for each issue. The BERT model outperformed traditional deep learning models $@k{=}2$ in terms of accuracy, precision, recall, and F1-score values, achieving an overall accuracy of 55%. Conversely, CNNs performed better than BERT in the presence of small-size training data, consisting of less than 5,000 issue reports.

Izadi et al. [50] adapted BERT to train a multi-class classifier. They identified separate lists of labels indicating if an issue report is related to a bug, an enhancement, or is a support/documentation issue. Based on these labels, 817,743 issue reports were collected from 60,958 projects hosted on GitHub. Each issue in the dataset had a single label. The fine-tuned RoBERTa classifier achieved 82% accuracy, outperforming the baseline models based on Multinomial Naive Bayes, BiLSTM networks, and the approach previously proposed by Kallis et al. [13].

Kallis et al. [15] introduced a dataset used in the NLBSE tool competition organized in 2022, referred to as the NLBSE'22 dataset throughout this thesis study. This dataset includes 803,417 issue reports collected from GitHub having one of the three labels: bug, enhancement, or question. On this dataset, they obtained an accuracy of 81.8% with the fastText approach they previously developed [13] and extended later [14].

Several studies experimented on this dataset, where most of the proposed models were developed based on BERT [39], including its different variants: RoBERTa [19], CodeBERT [40], ALBERT [41], and seBERT [42]. Table 5 presents the classification results of the models ranked by overall accuracy. First, the approaches are summarized, and then the results are compared with each other.

Izadi [51] applied the preprocessing steps explained in Section 4.2.1 and fine-tuned RoBERTa [19] for issue label classification. In addition, a logistic regression model was reported as an additional baseline model by the author.

Bharadwaj and Kadam [52] proposed several classifiers based on BERT (including vanilla BERT [39], CodeBERT [40], and RoBERTa [19]) and XLNet [56] to encode the issue content (title and body) as embeddings. They introduced three additional binary features: (1) whether the issue was submitted

Table 5: Performance of the models experimented on the NLBSE'22 dataset.

| Model | Bug | | | Enhancement | | | Question | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | |
| RoBERTa by *Izadi* [51] | .894 | .897 | .896 | .874 | .885 | .879 | .720 | .664 | .691 | **.872** |
| RoBERTa by *Bharadwaj & Kadam* [52] | .872 | .911 | .891 | .879 | .877 | .878 | .714 | .539 | .614 | **.865** |
| CodeBERT by *Bharadwaj & Kadam* [52] | .883 | .894 | .888 | .866 | .891 | .878 | .693 | .551 | .614 | **.862** |
| RoBERTa by *Colavito et al.* [53] | .875 | .898 | .886 | .871 | .874 | .872 | .767 | .559 | .612 | **.859** |
| BERT by *Siddiq & Santos* [54] | .883 | .888 | .885 | .859 | .888 | .873 | .678 | .546 | .605 | **.858** |
| seBERT by *Trautsch & Herbold* [55] | .866 | .906 | .886 | .864 | .877 | .871 | .731 | .487 | .584 | **.857** |
| XLNet by *Bharadwaj & Kadam* [52] | .879 | .885 | .882 | .853 | .890 | .871 | .706 | .534 | .608 | **.856** |
| BERT by *Bharadwaj & Kadam* [52] | .875 | .892 | .883 | .866 | .871 | .868 | .660 | .570 | .611 | **.855** |
| MLP by *Colavito et al.* [53] | .893 | .834 | .863 | .879 | .839 | .859 | .472 | .753 | .581 | **.829** |
| Logistic Regression by *Izadi* [51] | .841 | .867 | .854 | .822 | .850 | .835 | .655 | .432 | .521 | **.822** |
| fastText (Baseline) by *Kallis et al.* [15] | .811 | .904 | .855 | .844 | .815 | .830 | .669 | .336 | .447 | **.818** |

early in the project history, (2) whether the issue was created by the project owner, and (3) whether the sentences in the issue title were questions. The main preprocessing steps applied included regex-based substitutions of code snippets, URLs, usernames, and numbers with predefined tags.

Colavito et al. [53] experimented with fine-tuning BERT [39] and its variants ALBERT [41] and RoBERTa [19]. They observed that RoBERTa achieved the best F1-score on the minority class, the question class, on a validation set. Then, they compared the performance of using only the issue title

and the issue body with incorporating the association of the issue author. To combine text and author information, they trained an MLP classifier that leveraged the RoBERTa-based embeddings of the issue report with a one-hot encoding representation of the issue author association. The best overall accuracy performance was reached when only the issue title and the issue body were given to the model. The authors preprocessed the issues by replacing textual elements, including images, links, code snippets, URLs, email addresses, percent and currency symbols, phone numbers, user mentions, time, date, and numbers with predefined tags.

Siddiq and Santos [54] proposed a BERT-based [39] classifier, fine-tuned using the issue title and the issue body. Preprocessing steps included removing repeating whitespace characters (i.e., spaces, tabs, and line breaks) and replacing tabs and line breaks with spaces.

Trautsch and Herbold [55] fine-tuned seBERT [42], a variant of BERT [39], using the issue title and the issue body. Preprocessing steps involved replacing line breaks with whitespaces and removing multiple subsequent whitespaces.

All of the proposed models outperformed the initial study which was based on fastText [15] on overall accuracy. While the traditional models, MLP and logistic regression, achieved higher performance, the improvements were small (0.011 and 0.004, respectively), possibly resulting from the higher performance achieved for the question class. The remaining classifiers, based on BERT, achieved a comparable overall accuracy performance (0.855 - 0.872). Compared to fastText [15] and the other models [52, 53, 54, 55], the model proposed by Izadi [51] achieved the highest overall accuracy with a value of .872. Given that all of these models are based on BERT, the reason of its superior performance may be the additional data preprocessing applied, specifically the removal of duplicate issue reports in the training set.

Based on the error analysis of the proposed model [53], Colavito et al. [57] experimented with different data quality criteria applied to the NLBSE'22 dataset to evaluate the impact of such data quality filters on the model performance. To this end, they defined several data quality criteria and applied this set of filters to the training set of the NLBSE'22 dataset in a progressive manner to filter out uncertain data. They stated that none of their attempts had a significant effect on the model performance. In addition to the NLBSE'22 dataset, they also experimented with a Jira dataset, introduced in previous work [58]. This dataset consisted of 2.2M issue reports with a label of bug, enhancement, or question, already satisfying the determined data quality criteria. The same preprocessing steps in previous work [53] were applied to the Jira dataset. RoBERTa achieved an overall accuracy of 91.36% on this dataset. However, the smaller number of issue reports in the question class had a significant impact on the model performance, resulting in lower precision, recall, and F1-score values for all classes except for the bug class.

# CHAPTER 3

# METHODOLOGY

This thesis study primarily aims to investigate the relevance of the issue labels assigned to the issue reports created in OSS development projects. In addition, it investigates the utilization of state-of-the-art LLMs for issue label classification. To this end, we employed a mixed-methods research approach, integrating both qualitative and quantitative research techniques, to provide a comprehensive analysis of the research questions raised in Chapter 1, which are as follows:

> **RQ1.** To what extent are the labels of the issue reports created in OSS development projects reliable?
>
> **RQ2.** To what extent can the label of a newly created issue report in an OSS development project be predicted using state-of-the-art LLMs?

We focused on the OSS development projects hosted on GitHub, as it is one of the most prominent OSS development platforms. Specifically, we performed analyses on two datasets consisting of issue reports submitted to multiple OSS development projects hosted on GitHub: (1) a dataset used in the NLBSE tool competition organized in 2022 [15], and (2) a dataset created as part of this thesis study using the REST API provided by GitHub. The first dataset contains issue reports from a large number of repositories without covering all issues within those repositories. Additionally, it does not consider the star ratings and the accumulated activity of the repositories. Hence, we constructed an additional dataset with the aim of obtaining a more representative dataset reflecting the nature of large-scale OSS development projects. This dataset includes all issue reports created in the selected repositories, which were ensured to be popular based on their star ratings and active based on their accumulated activity. Further details regarding these datasets are provided in Section 4.1.

We primarily considered the issue reports associated with an encountered bug, an enhancement request, or a question about the software product. Indeed, bug, enhancement, and question are among the issue labels included by default in each repository once it is created on GitHub, as described in Section 2.2, and they are the three most commonly used issue labels in OSS development projects hosted on GitHub [9]. We identified the relevant issue reports by considering the corresponding labels assigned to each issue. For each issue report, we utilized the following data fields: issue title, issue body, issue label, timestamp of creation, issue number, repository name, and the association of the issue author.

The experiments were conducted using RoBERTa [19] and Llama 3, including its 8B and 70B variants [20]. It is important to note that RoBERTa is a pre-trained model generally fine-tuned for downstream tasks. In this thesis study, it was first initialized with the pre-trained parameters, and all parameters

were fine-tuned using the labeled data from the training sets in a supervised manner. This means that the RoBERTa model could potentially learn from incorrectly assigned issue labels and be potentially misled by these errors in its predictions. In contrast, there was no such risk for the predictions made by the Llama 3 model, as it involved only inference without any training or fine-tuning. It classified an issue report based solely on the issue title and the issue body, without using the assigned issue label as input. As the Llama 3 model requires high computational power and processing time, particularly for the Llama 3 70B variant, the experiments with Llama 3 variants were conducted using the random samples taken from two different test sets.

In the RoBERTa model, the following data fields were provided to the model as input: issue title, issue body, timestamp of creation, repository name, and the association of the issue author. All this information is available at the time the issue is created. The model was fine-tuned with the assigned issue labels in the training sets. In the case of Llama 3, only the issue title and the issue body were given to the model, as the other fields were considered to provide no valuable information due to the lack of corresponding fine-tuning. Since the study aims to investigate the reliability of the assigned issue labels, we requested the Llama 3 model to classify each issue into the most suitable category among the provided ones: bug, enhancement, question, or other. In addition, we requested its confidence level in making such a classification and its assessment of the understandability level of the provided issue, both ranging between 0 and 100.

The classification performance of a model with respect to the assigned issue labels was measured by overall accuracy and class-based performance metrics: precision, recall, and F1-score, calculated using the equations 3.1, 3.2, 3.3, and 3.4, respectively.

$$A = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i + TN_i + FN_i)} \tag{3.1}$$

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{3.2}$$

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{3.3}$$

$$F1_c = \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \tag{3.4}$$

where $n$ is the number of classes and $c$ denotes a class in the dataset.

In these equations, the $TP$, $FP$, $TN$, and $FN$ denote the number of True Positives, False Positives, True Negatives, and False Negatives, respectively, as explained below.

$TP_c$ : The number of instances predicted as class $c$ that truly belong to class $c$.

$FP_c$ : The number of instances predicted as class $c$ but do not actually belong to class $c$.

$TN_c$ : The number of instances that are not predicted as class $c$ and do not belong to class $c$.

$FN_c$ : The number of instances that are not predicted as class $c$ but actually belong to class $c$.

For the reliability analysis of issue labels, the title and the body of the issue reports were considered, as they describe the content of an issue report written by its author. We investigated to what extent the assigned labels of the issue reports are relevant to their contents. To do that, we focused on the random samples taken from the test sets, as it was infeasible to manually classify issue reports based on their contents due to the orders of magnitude of the number of issue reports. First, we checked whether each random sample was representative of its larger population by comparing the performance of the RoBERTa model on the random sample to its performance on the entire dataset. After ensuring that, we manually classified the issue reports for which at least one of the three models classified the issue report with a label different from the assigned one. We performed this classification relying only on the issue content, involving the issue title and the issue body. The results of the manual classification were compared with the assigned issue labels and the classifications performed by each LLM.

Figure 2 depicts the research approach followed to answer the research questions.

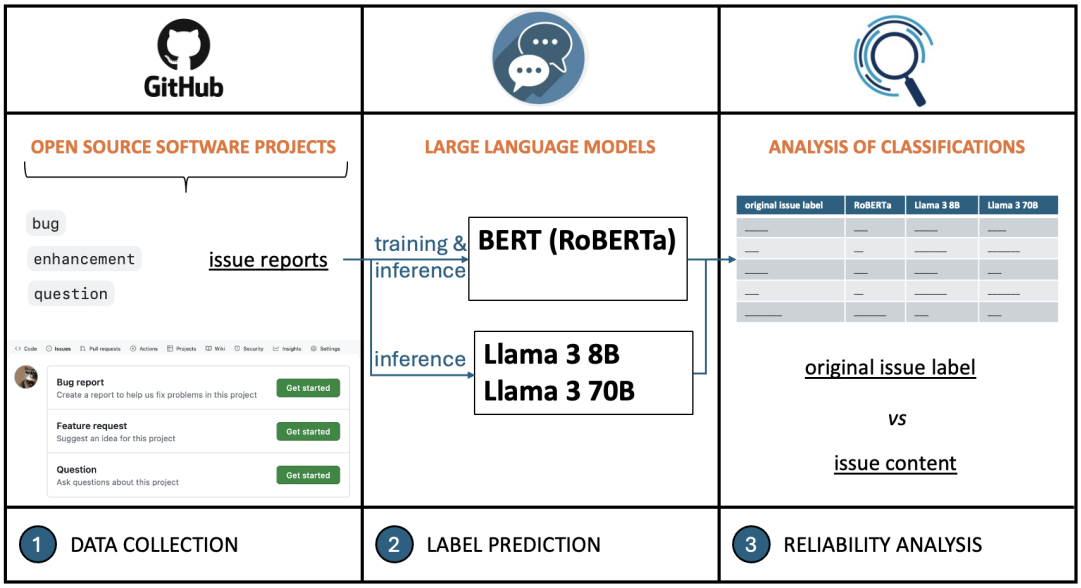


Figure 2: Overview of the research approach.

# CHAPTER 4

# EXPERIMENTS ON ISSUE LABEL CLASSIFICATION

This chapter presents the design details of the experiments conducted for issue label classification and the reliability analysis. The datasets are introduced in Section 4.1. The details of the experiments conducted with LLMs are provided in Section 4.2. The analysis performed to investigate the reliability of the assigned issue labels is detailed in Section 4.3. The results are presented in the following chapter.

## 4.1 Datasets

The experiments were conducted on two different datasets. This section introduces the data collection process and provides a brief description of each dataset.

### 4.1.1 NLBSE'22 Dataset

One of the datasets was provided by Kallis et al. [15] in the NLBSE tool competition organized in 2022, referred to as the NLBSE'22 dataset throughout this thesis study. This dataset contains 803,417 issues collected from 127,595 OSS projects hosted on GitHub. The authors of the dataset extracted closed issues during the first semester of 2021 (from January 1$^{st}$ to May 31$^{st}$) that contained any of the following labels at the time of issue closing: bug, enhancement, or question.

Approximately 90% of the dataset is designated as the training set by the creators of the dataset, which includes 722,899 issues. The remaining 80,518 issues belong to the test set. Issues are categorized into one of the three labels: bug, enhancement, or question, with each issue assigned only a single label. In both the training and test sets, approximately 50% of the issue reports are labeled as bug, 41% are labeled as enhancement, and 9% are labeled as question. The number of issue reports with distinct issue labels in each set is provided in Table 6.

Table 6: The number of distinct labels in each set in the NLBSE'22 dataset.

| **Label** | **Train** | **Test** | **Total** |
|---|---|---|---|
| bug | 361,103 | 40,288 | 401,391 |
| enhancement | 299,374 | 33,203 | 332,577 |
| question | 62,422 | 7,027 | 69,449 |
| Total | 722,899 | 80,518 | 803,417 |

The dataset contains the following data fields for each issue report: issue title, issue body, issue label, timestamp of creation, issue URL, repository URL, and the association of the issue author. All issue reports have a non-empty issue title, as this is a mandatory data field for issue reports on GitHub. However, about 72K issue reports in the training set and 8K issue reports in the test set lack the issue body; that is, they do not have any description in addition to the issue title. There are six types of issue author association in the training set: collaborator, contributor, mannequin, member, none, and owner. The test set includes the same association types, except for the mannequin. The oldest issue report in the training set was created in 2005, while the newest one was opened in 2021. In the test set, the oldest issue was created in 2007, and the newest one was opened in 2021.

There are 26,220 and 380 duplicate issue reports in the training and test sets, respectively. Additionally, 7,086 issue reports (corresponding to 5,426 unique issues) in the training set also appear in the test set. To prevent data leakage, we removed these issue reports from the training set. No modifications were performed on the test set to preserve its integrity.

### 4.1.2 GitHub'23 Dataset

The second dataset was constructed as part of this thesis study using the REST API provided by GitHub with Python scripts, and it is referred to as the GitHub'23 dataset throughout this thesis. As noted by Kalliamvakou et al. [59], there are important aspects that might hinder the quality of the data scraped from the website or gathered from the API of GitHub. For example, the authors observed that most of the publicly available projects hosted on GitHub are actually personal and inactive. Additionally, many repositories are not used for software development purposes but rather as free storage services or web hosting platforms. Since there were no concerns about the popularity or accumulated activity of the projects included in the NLBSE'22 dataset, we aimed to construct an additional dataset that is more representative of large-scale OSS development projects. As the first step, we determined the repositories to be included in the study based on the following criteria:

**1. Programming Language:** JavaScript, Python, and Java have been the top three most commonly used programming languages on GitHub by developers since 2015[11]. Among these, we arbitrarily selected the Java programming language and focused on the repositories that utilized Java in their software development and maintenance activities.

**2. Public:** We focused on OSS development projects hosted on GitHub. As they were public, we were able to extract data on issues, pull requests, commits, and releases without requiring any authorization.

**3. Creation Time:** We restricted our analysis to the repositories that were at least five years old to ensure that the projects had sufficient time to accumulate issues, pull requests, commits, and releases.

**4. Unarchived:** On GitHub, a repository may be archived by its owner if it is no longer actively maintained. In this case, the repository becomes read-only for all users. We included only unarchived repositories in our study to focus on those that had sufficient time for software development and maintenance activities.

[11] https://octoverse.github.com/2022/top-programming-languages

**5. Stars:** There are three common popularity metrics on GitHub: the number of stars, watchers, and forks. Among these, the number of stars is considered the most useful to assess the popularity of GitHub projects by software development practitioners. In addition, it is observed that many issues co-occur with a faster repository growth in terms of the number of stars [60]. Therefore, in line with the recent empirical studies in software engineering [61, 62, 63, 64], the number of stars was relied on while selecting the repositories. In particular, a repository was included in the study only if it had more than 10,000 stars.

Based on these criteria, the following query was searched on GitHub[12] in October 2023.

```
language:Java
AND is:public
AND archived:FALSE
AND created:<2018-10-01
AND stars:>10000
```

This query returned 171 repositories, which can be seen in Appendix A.1. For each repository in this initial list, the number of issues, pull requests, commits, and releases were obtained through the REST API provided by GitHub. The characteristics of these repositories are presented in Table 7.

Table 7: Characteristics of the repositories in the initial list.

| State | # of repositories | # of issues | # of pull requests | # of commits | # of releases |
|---|---|---|---|---|---|
| Open | - | 66,878 | 12,176 | - | - |
| Closed | - | 413,256 | 448,780 | - | - |
| Total | 171 | 480,134 | 460,956 | 2,227,380 | 8,249 |

Next, we filtered the repositories in the initial list using the following thresholds. The reason for this filtering was to include a repository in further analysis only if it had sufficient accumulated activity.

```
(# of issues >= 1000)
AND (# of pull requests >= 1000)
AND (# of commits >= 1000)
AND (# of releases >= 50)
```

Table 8 provides the characteristics of the repositories that satisfied these thresholds and were included in further analysis.

Table 8: Characteristics of the repositories satisfying the thresholds.

| State | # of repositories | # of issues | # of pull requests | # of commits | # of releases |
|---|---|---|---|---|---|
| Open | - | 28,660 | 3,230 | - | - |
| Closed | - | 240,528 | 255,481 | - | - |
| Total | 32 | 269,188 | 258,711 | 608,033 | 4,582 |

[12] https://github.com

To investigate the assigned labels of the issue reports, open issues were excluded from the study, as work on them might still need to be completed, or their current labels could change as work progresses. Among 240,528 closed issue reports, 42,447 issue reports did not have any assigned label. Hence, they were also excluded from the study. After these steps, 198,081 issue reports remained in the dataset. Finally, the issue reports created in two repositories, `halo-dev/halo` and `alibaba/nacos`, were also excluded, as the language of the issue reports in these repositories was not English. In the end, 190,436 issues created in 30 repositories, listed in Appendix A.2, were included in further analysis.

There were 1,914 unique issue labels used in these repositories. The average number of distinct labels per repository was 74.4, while the median was 43. We filtered this dataset to include issue reports associated with a bug, an enhancement, or a question, as these labels are included by default in each repository on GitHub when it is created, as described in Section 2.2, and they are the three most commonly used issue labels in OSS development projects hosted on GitHub [9]. Additionally, these labels were aligned with the NLBSE'22 dataset for comparison purposes. We aimed to enable a fair comparison of the performances achieved by the models. The identification of these issues was performed in three rounds, explained below.

**Round 1**

As labels can be defined in free text form in repositories, we observed that different projects employed various labels with a common semantic meaning. For example, issues created to report bugs were assigned labels such as Bug, type:bug, and type/bug. Similarly, issue reports related to enhancements had labels like Type:Enhancement and >enhancement. Hence, we grouped these labels into corresponding categories. For each selected repository, the author of the thesis manually reviewed the labels defined on GitHub for that particular repository. First, the labels indicating that an issue was related to a bug were included in the rule set and grouped under a single label as bug. Next, the labels indicating that an issue was related to an enhancement were included in the rule set and grouped under a single label as enhancement. Based on the repository and label pairs, each issue report in the dataset was assigned a new label as follows: bug, enhancement, or to-be-defined. The rule set created for Round 1 can be seen in Appendix B.1, and the distribution of labels obtained after Round 1 is shown in Table 9.

Table 9: Distribution of distinct issue labels after Round 1.

| **Label** | **# of issues** | **% of issues** |
|---|---|---|
| to-be-defined | 100,258 | 53 |
| bug | 54,669 | 29 |
| enhancement | 35,509 | 18 |
| Total | 190,436 | 100 |

**Round 2**

In Round 2, the remaining issue reports (categorized as to-be-defined in Round 1) were analyzed, and up to the top 10 issue labels, representing the highest proportion of issue reports in each repository, were manually reviewed. During this review, it was observed that specific labels were used to indicate that an issue was not related to a bug, an enhancement, or a question. For example, some issues were created to manage testing processes or outline documentation changes. Therefore, being different from Round 1, specific labels were consolidated under the category of other, indicating that the issue labeled as such was neither a bug, nor an enhancement, nor a question, but belonged to a different category.

The rule set developed for Round 2 is provided in Appendix B.2, and the distribution of labels obtained after Round 2 is presented in Table 10.

Table 10: Distribution of distinct issue labels after Round 2.

| Label | # of issues | % of issues |
|---|---|---|
| other | 59,678 | 32 |
| bug | 55,639 | 29 |
| enhancement | 36,365 | 19 |
| to-be-defined | 26,749 | 14 |
| question | 12,005 | 6 |
| Total | 190,436 | 100 |

**Round 3**

Finally, the steps performed in Round 2 were applied once again in Round 3. The rule set created for Round 3 can be seen in Appendix B.3 and the distribution of labels obtained after Round 3 is provided in Table 11.

Table 11: Distribution of distinct issue labels after Round 3.

| Label | # of issues | % of issues |
|---|---|---|
| other | 66,467 | 35 |
| bug | 55,945 | 29 |
| enhancement | 36,369 | 19 |
| to-be-defined | 19,318 | 10 |
| question | 12,337 | 7 |
| Total | 190,436 | 100 |

In the end, 104,651 issue reports were identified as being related to any of the following labels: bug, enhancement, or question in the GitHub'23 dataset. In line with the NLBSE'22 dataset, 90% of the issue reports were randomly selected as the training set. The train-test split was performed in a stratified manner based on the issue labels, with the random seed parameter set to 42. The training set includes 94,185 issue reports, while the remaining 10,466 issue reports constitute the test set. In both the training and test sets, 53% of the issue reports are labeled as bug, 35% are labeled as enhancement, and 12% are labeled as question. The number of issue reports with distinct issue labels in each set is provided in Table 12.

Table 12: The number of distinct labels in each set in the GitHub'23 dataset.

| Label | Train | Test | Total |
|---|---|---|---|
| bug | 50,350 | 5,595 | 55,945 |
| enhancement | 32,732 | 3,637 | 36,369 |
| question | 11,103 | 1,234 | 12,337 |
| Total | 94,185 | 10,466 | 104,651 |

The dataset contains the same data fields as the NLBSE'22 dataset: issue title, issue body, issue label, timestamp of creation, issue URL, repository URL, and the association of the issue author. All issue reports have a non-empty issue title, as this is a mandatory data field for issue reports on GitHub. However, 275 issue reports in the training set and 22 issue reports in the test set contain no text in the issue body. There are five types of issue author association in the training set: collaborator, contributor, member, none, and owner. The test set includes the same association types, except for the owner. The oldest issue report in the training set was created in 2003, while the newest one was opened in 2023. In the test set, the oldest issue was created in 2004, and the newest one was opened in 2023.

There are no duplicate issue reports in the training and test sets, and no issue reports from the training set are included in the test set.

### 4.2 Label Classification

This section provides the details of the steps followed to conduct the experiments for issue label classification. Specifically, RoBERTa [19] and Llama 3, including its 8B and 70B variants [20], were used for the issue label classification problem. The experiments were conducted on Google Colab Pro[13], utilizing an NVIDIA A100 GPU with 40 GB of RAM.

#### 4.2.1 Label Classification with RoBERTa

The pre-trained RoBERTa [19] model, whose details are explained in Section 2.3.3.2, was fine-tuned for issue label classification. To prevent data leakage, we removed the issue reports in the training sets that were also included in the test sets, as explained below.

> **0. Preventing data leakage:** The issue reports in the training sets that were also included in the test sets were identified based on the issue URL information and removed from the training sets. No modifications were performed on the test sets to preserve their integrity. This process led to the removal of 7,086 issue reports from the NLBSE'22 training set. After this step, the number of issue reports with distinct labels in each set of the NLBSE'22 dataset can be seen in Table 13. No issues from the GitHub'23 training set were included in the GitHub'23 test set.

In addition, the following preprocessing steps were applied to the datasets, in line with previous work by Izadi [51].

> **1. Removing duplicate issue reports:** Duplicate issues were identified based on the issue URL information and removed only from the training sets to preserve the integrity of the test sets. This process led to the removal of 24,560 issue reports from the NLBSE'22 training set. After this step, the number of issue reports with distinct issue labels in each set of the NLBSE'22 dataset is provided in Table 14. No duplicate issues were found in the GitHub'23 training set.

[13] https://colab.research.google.com

**2. Replacing function calls:** Function calls that appear in the issue body were identified using regular expressions and replaced with a fixed token of `" function "` in both the training and test sets.

**3. Replacing issue numbers:** Issue numbers that appear in the issue title and the issue body were identified using regular expressions and replaced with a fixed token of `" issue "` in both the training and test sets.

**4. Converting to lower case:** The issue title and the issue body were converted to lower case in both the training and test sets.

**5. Removing punctuations:** Punctuations, except for periods (.) and question marks (?), were removed from the issue title and the issue body in both the training and test sets. The rationale for retaining periods and question marks was to preserve the contextual meaning of the text, as RoBERTa and Llama 3 are contextual models.

**6. Removing non-ASCII characters:** Non-ASCII characters were removed from the issue title and the issue body in both the training and test sets using regular expressions.

**7. Replacing the fixed part in the repository URL:** The fixed part in the repository URL field, `https://api.github.com/repos/`, was removed in both the training and test sets to extract the repository name.

**8. Removing multiple spaces between words:** Consecutive whitespace characters were replaced with a single space in the issue title and the issue body in both the training and test sets.

**9. Truncating columns:** The average number of tokens was approximately 7 in the issue title and 118 in the issue body in both the NLBSE'22 training and test sets. The corresponding values for the issue title and the issue body were approximately 8 and 200, respectively, in both the GitHub'23 training and test sets. In this dataset, the median number of tokens in the issue body was 121 in the training set and 119 in the test set. To avoid exceeding the memory capacity of the GPU, the maximum lengths for the issue title and the issue body were set to 30 and 170 tokens, respectively, making a total of 200 tokens.

**10. Concatenating issue fields:** Issue fields were concatenated as follows:

```
'time ' + {timestamp of creation} +
' author ' + {association of the issue author} +
' repo ' + {repository name} +
' title ' + {issue title} +
' body ' + {issue body}
```

**11. Encoding categorical labels:** Each distinct issue label was mapped with a unique integer value as shown below:

- **bug** was mapped to **0**
- **enhancement** was mapped to **1**
- **question** was mapped to **2**

Table 13: The number of distinct labels in each set in the NLBSE'22 dataset after removing issue reports from the training set (to prevent data leakage).

| Label | Train | Test | Total |
|---|---|---|---|
| bug | 357,293 | 40,288 | 397,581 |
| enhancement | 297,000 | 33,203 | 330,203 |
| question | 61,520 | 7,027 | 68,547 |
| Total | 715,813 | 80,518 | 796,331 |

Table 14: The number of distinct labels in each set in the NLBSE'22 dataset after removing duplicate issue reports from the training set.

| Label | Train | Test | Total |
|---|---|---|---|
| bug | 344,994 | 40,288 | 385,282 |
| enhancement | 288,015 | 33,203 | 321,218 |
| question | 58,244 | 7,027 | 65,271 |
| Total | 691,253 | 80,518 | 771,771 |

To fine-tune the pre-trained RoBERTa model, the following libraries were used in provided versions: `transformers`[14] in version `4.40.2` and `simpletransformers`[15] in version `0.64.3`.

The AdamW optimizer and the cross-entropy loss function were used for optimization. The following parameters were set in line with previous work [51].

- learning rate = $3e$ - 5
- epochs = 4
- dropout = 0
- maximum input length = 200
- training batch size = 100
- evaluation batch size = 100

#### 4.2.2 Label Classification with Llama 3

The Llama 3 model, including its 8B and 70B variants [20], was utilized for issue label classification. As these models require high computational power and processing time, particularly for the Llama 3 70B variant, and the subsequent manual classification performed for the reliability analysis is labor-intensive, the experiments were conducted using random samples taken from the test sets.

A random sample of 5,000 issue reports was taken from the NLBSE'22 test set, corresponding to 6% of the test set. The sampling was performed in a stratified manner based on the assigned issue labels,

[14] https://github.com/huggingface/transformers
[15] https://github.com/ThilinaRajapakse/simpletransformers

with the random seed parameter set to 42. The distribution of the assigned issue labels in this random sample can be seen in Table 15.

Table 15: The number of distinct labels in the random sample of the NLBSE'22 test set.

| **Label** | **# of issues** |
|---|---|
| bug | 2,502 |
| enhancement | 2,062 |
| question | 436 |
| Total | 5,000 |

Similarly, 5,000 issue reports were randomly selected from the GitHub'23 test set, representing 48% of the total test set. The sampling was performed in a stratified manner based on the assigned issue labels, with the random seed parameter set to 42. The distribution of the assigned issue labels in this random sample can be seen in Table 16.

Table 16: The number of distinct labels in the random sample of the GitHub'23 test set.

| **Label** | **# of issues** |
|---|---|
| bug | 2,673 |
| enhancement | 1,738 |
| question | 589 |
| Total | 5,000 |

The prompt presented in Figure 3 was provided to both the 8B and 70B variants of the Llama 3 model. By utilizing this prompt, the Llama 3 model attempted to perform three tasks in total. First, it classified the issue into the most suitable category among the provided ones: bug, enhancement, question, or other. Second, it provided its confidence level in making such a classification. Third, it evaluated the understandability level of the issue, determining whether its content was easily comprehensible or not. It is important to note that the issue title and the issue body were provided to the model exactly as they appeared in the issue report, with no preprocessing performed.

To access the API of the Llama 3 model, the `Ollama` library[16] was utilized, with the following parameters set as explained below:

- temperature = 0
  (to reduce randomness in generated responses)
- context length = 8192
  (to set the maximum number of tokens the model can process before generating responses)

[16] https://ollama.com/library

```
Classify the following issue as "bug", "enhancement", "question" or
"other" and rate your classification confidence and issue
understandability in range 0 - 100. Your answer should only include:
Classification: your classification answer
Prediction Confidence: your level of confidence
Issue Understandability: level of understandability
Do not provide any explanation.
Issue Title: {issue title}
Issue Body: {issue body}
```

Figure 3: The prompt provided to Llama 3.

### 4.3 Reliability Analysis of Labels

To investigate the reliability of the assigned issue labels, the classifications performed by three models—RoBERTa, Llama 3 8B, and Llama 3 70B—were compared. Specifically, the issues for which at least one of the models classified the issue with a label different from the assigned one were analyzed. We manually classified these issues based strictly on the text provided in the issue title and the issue body. The decision to use only the issue title and the issue body for manual classification was based on the fact that these two describe the motivation behind the creation of the issue stated by its author. In addition, these two were the only inputs provided to the Llama 3 model, including its 8B and 70B variants, for issue label classification. The manual classification was performed based on the following guideline determined by the author and supervisors of the thesis, where a majority vote was taken to resolve possible conflicts.

An issue report was categorized as a bug if it described an unexpected behavior of the software product that prevented it from functioning correctly for users. At this point, we aimed to differentiate between the problems inherent in the software product and the questions simply asked to resolve user errors. The reason is that developers should take necessary actions as soon as possible to eliminate the problems with the software system, as these problems may affect a large number of users. They need to investigate the root cause of the problem, make corrections in the code and necessary files, test the applied changes, and deploy the product to users again. On the other hand, for user errors, developers generally guide the user on the actions they should take to resolve the problem. Hence, an issue report regarding the problems caused by user errors was labeled as a question, whereas an issue report created to report system errors was categorized as a bug.

An enhancement request corresponds to a formal suggestion or proposal to improve, extend, or modify existing features or functionality of a software application. In contrast to bug reports, which address problems that need to be fixed because the software is not functioning as intended, enhancement requests focus on improving the software product by adding new capabilities, advancing its performance, or refining user experience. An issue report containing such suggestions was labeled as an enhancement.

An issue report created to ask questions about the software product was categorized as a question. These issue reports include inquiries about how to solve a problem, modify the software, integrate it with other software or hardware, whether a specific functionality is supported, release planning details, and similar topics.

An issue report was categorized as other if it did not fit any of these categories: bug, enhancement, or question, but contained information related to a different category. Examples of this category include the issue reports created to share general announcements or guidelines.

An issue report was excluded from the manual classification if its issue title and issue body were in a language other than English.

Finally, if an issue report did not contain enough information to be classified as a bug, enhancement, question, or other, it was categorized as unclear. The reason is that the text provided in the issue title and the issue body was not sufficient to make a proper classification.

# CHAPTER 5

# EXPERIMENTAL RESULTS

This section presents the results obtained from the experiments conducted for issue label classification and the subsequent analysis performed to investigate the reliability of the issue labels.

## 5.1 Label Classification

For issue label classification, experiments with RoBERTa [19] and Llama 3, including its 8B and 70B variants [20], were conducted.

### 5.1.1 Label Classification with RoBERTa

For each dataset, NLBSE'22 and GitHub'23, a separate RoBERTa model was fine-tuned on its respective training set, and the model's performance was evaluated on the corresponding test set. Fine-tuning the RoBERTa model on the NLBSE'22 training set took 1 hour and 15 minutes, while it took 10 minutes to fine-tune the model on the GitHub'23 training set.

The class-based precision, recall, and F1-score, and the overall accuracy performance of the RoBERTa model on the NLBSE'22 test set and the GitHub'23 test set are presented in Table 17. The confusion matrices are reported in Table 18 and Table 19 for the NLBSE'22 and the GitHub'23 datasets, respectively. The RoBERTa model achieved an overall accuracy of **.8729** on the NLBSE'22 dataset and **.8827** on the GitHub'23 dataset.

Notably, the more populated categories (bug and enhancement) obtained higher classification results for all metrics compared to the less-represented class, question. This difference can be explained by two factors: (1) the smaller number of samples in the question class, and (2) the greater diversity of information and vocabulary within this class. The NLBSE'22 dataset was collected using only a time-based criterion for inclusion, without any concerns about the popularity or activity of the repositories. In contrast, the GitHub'23 dataset was drawn from large-scale repositories selected based on their popularity and activity, with the aim of obtaining a more representative dataset reflecting the nature of large-scale OSS development projects. All class-based performance metrics (except for the recall and F1-score of the enhancement class, which were slightly decreased) improved on the GitHub'23 dataset. The improvements in the worst-performing question class were 13% in precision, 5% in recall, and 9% in F1-score. Although the overall accuracy increased from .8729 to .8827, the difference was not significant.

Table 17: Performance of RoBERTa on datasets.

| Model | Bug | | | Enhancement | | | Question | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | |
| NLBSE'22 | .8940 | .8987 | .8964 | .8789 | .8834 | .8812 | .7142 | .6758 | .6945 | **.8729** |
| GitHub'23 | .8957 | .9269 | .9110 | .8844 | .8727 | .8785 | .8077 | .7115 | .7566 | **.8827** |

Table 18: Confusion matrix of RoBERTa on the NLBSE'22 dataset.

| Actual | Prediction | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 36,207 (90%) | 3,022 (7%) | 1,059 (3%) |
| enhancement | 3,031 (9%) | 29,331 (88%) | 841 (3%) |
| question | 1,260 (18%) | 1,018 (14%) | 4,749 (68%) |

Table 19: Confusion matrix of RoBERTa on the GitHub'23 dataset.

| Actual | Prediction | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 5,186 (93%) | 286 (5%) | 123 (2%) |
| enhancement | 377 (11%) | 3,174 (87%) | 86 (2%) |
| question | 227 (18%) | 129 (11%) | 878 (71%) |

#### 5.1.2 Label Classification with Llama 3

This section presents the results obtained with the Llama 3 model, including its 8B and 70B variants. As these models require high computational power and processing time, particularly for the Llama 3 70B variant, and the subsequent manual classification performed for the reliability analysis is labor-intensive, the experiments were conducted using random samples selected from the NLBSE'22 and the GitHub'23 test sets. To ensure the representativeness of the random samples, we used stratified sampling based on the assigned issue labels. A random sample of 5,000 issue reports was taken separately from each test set, and the same random samples were used across all subsequent experiments.

The prompt described in Section 4.2.2 was provided to both the 8B and 70B variants of the Llama 3 model. By utilizing this prompt, the Llama 3 model attempted to perform three tasks in total. First, it classified the issue into the most suitable category among the provided ones: bug, enhancement, question, or other. Second, it provided its confidence level in making such a classification. Third, it evaluated the understandability level of the issue, determining whether its content was easily comprehensible or not. The issue title and the issue body were provided to the model as they appeared in the issue report, without applying any preprocessing. For the random sample taken from the NLBSE'22 test set, the processing of the issues took 34 minutes with the 8B variant and 5 hours and 20 minutes with the 70B variant. Processing all issues in the random sample taken from the GitHub'23 test set took 39 minutes with the 8B variant and 6 hours and 5 minutes with the 70B variant.

In the following, the post-processing steps applied to the responses generated by the models are explained, and the results obtained for each of the three tasks performed by the models are provided.

**Model: Llama 3 8B**
**Dataset: NLBSE'22**

The Llama 3 8B model provided 116 unique responses to the prompt for the 5,000 issue reports randomly selected from the NLBSE'22 test set. Responses were converted to lower case and parsed with the keywords in the prompt: "classification:", "prediction confidence:", and "issue understandability:". For some issue reports, the responses included special characters (* and %) that did not conform to the expected output format and, as a result, could not be parsed correctly. These responses were manually processed. The details of this process, including the issue reports associated with such responses, the original outputs, the corresponding corrections, and the distributions of responses obtained after corrections are provided in Appendix C.1.

After the corrections, it was observed that the responses returned for 19 issue reports were not in the requested format described in the prompt. For these issue reports, the model either provided only a classification, without generating any output regarding the confidence level of its classification or the understandability level of the issue, or it provided explanations about what was described in the issue content, such as recommendations. By taking a closer look, we realized that for these issue reports, the issue body contained different symbols and blocks with large numbers of tokens and characters. Hence, we summarized the issue body of these issue reports with the Llama 3 70B variant. The indexes of these issue reports in the NLBSE'22 test set are presented in Appendix C.3. Then, for each issue report, we provided the same prompt to the Llama 3 8B variant with the summarized issue body instead of the original one and parsed the response again. The responses generated with the summarized issue body included outputs for the three tasks in the requested format. Therefore, we continued with the summarization approach, although other text extraction methods could also be applied.

**Model: Llama 3 70B**
**Dataset: NLBSE'22**

The Llama 3 70B model provided 101 unique responses to the prompt for the 5,000 issue reports randomly selected from the NLBSE'22 test set. These responses were parsed and analyzed by following the same steps described earlier. The details of these steps are given in Appendix C.2.

For 19 issue reports, where the Llama 3 8B variant could not generate responses in the requested format, the Llama 3 70B variant also produced problematic responses. The indexes of these issue reports can be seen in Appendix C.3. For each of these issue reports, the prompt was provided to the Llama 3 70B model once more, using the summarized issue body obtained by the Llama 3 70B variant instead of the original one, and then the response was parsed again.

**Model: Llama 3 8B**
**Dataset: GitHub'23**

The Llama 3 8B model provided 113 unique responses to the prompt for the 5,000 issue reports randomly selected from the GitHub'23 test set. These responses were parsed and analyzed in a similar manner. The details of these steps are presented in Appendix C.4.

It was noted that the responses returned for 27 issue reports were not in the requested format illustrated in the prompt. For these issue reports, the model either provided only a classification, without generating any output regarding the confidence level of its classification or the understandability level of the issue, or it provided explanations about what was described in the issue content. We summarized the issue body of these issue reports with the Llama 3 70B variant. The indexes of these issue reports in the GitHub'23 test set are listed in Appendix C.6. Then, for each issue report, we provided the same prompt to the Llama 3 8B variant with the summarized issue body instead of the original one and parsed the response again.

**Model: Llama 3 70B**
**Dataset: GitHub'23**

The Llama 3 70B model provided 90 unique responses to the prompt for the 5,000 issue reports randomly selected from the GitHub'23 test set. By following the same steps outlined earlier, the responses were parsed and analyzed. The details of these steps are provided in Appendix C.5.

It was observed that the responses generated for 28 issue reports by the Llama 3 70B variant (including the same 27 issue reports for which the Llama 3 8B variant could not generate responses in the requested format and an additional issue report) were problematic. The indexes of these issue reports are presented in Appendix C.6. For each of these issue reports, the prompt was provided to the Llama 3 70B model once more, using the summarized issue body obtained by the Llama 3 70B variant instead of the original one, and then the response was parsed again.

The analysis of the responses for each of the three tasks performed by the Llama 3 8B and Llama 3 70B models is presented below for both datasets.

#### 5.1.2.1 Label Classification

The performance of the three models—RoBERTa, Llama 3 8B, and Llama 3 70B—on the random sample selected from the NLBSE'22 test set can be seen in Table 20, and the confusion matrices are reported in Table 21, Table 22, and Table 23, respectively.

It was noted that the overall accuracy achieved by the RoBERTa model on the entire NLBSE'22 test set **(.8729** as given in Table 17) and on the random sample of 5,000 issue reports taken from this dataset **(.8764)** were very close to each other. The class-based precision, recall, and F1-score values obtained by the RoBERTa model were also consistent across these two datasets. This close alignment suggested that the random sample was representative of the larger population from which it was drawn.

The 8B and 70B variants of the Llama 3 model achieved an overall accuracy of **.7706** and **.8310**, respectively, as presented in Table 20. Compared with the RoBERTa model, both Llama 3 variants obtained lower values in terms of class-based performance metrics over all three classes and overall accuracy. As indicated by the confusion matrices, the models mostly struggled with identifying the question class, generally classifying question issues as either bugs or enhancements. Among the Llama 3 models, the 70B model outperformed the 8B model in overall accuracy and all class-based performance metrics. The performance improvement was mostly due to the higher precision achieved in the question class, which appeared to be the most difficult to predict.

Table 20: Performance of the models on the random sample of the NLBSE'22 test set.

| **Metric** | **RoBERTa** | | | **Llama 3 8B** | | | **Llama 3 70B** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| bug | .8970 | .8981 | .8975 | .8389 | .8157 | .8272 | .8762 | .8713 | .8737 |
| enhancement | .8836 | .8870 | .8853 | .7236 | .8569 | .7846 | .8391 | .8700 | .8543 |
| question | .7200 | .7018 | .7108 | .3629 | .1032 | .1607 | .6882 | .4151 | .5179 |
| other* | - | - | - | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 |
| **Accuracy** | | **.8764** | | | **.7706** | | | **.8310** | |

* Added since some predictions made by the Llama 3 8B and 70B models were in this category.

Table 21: Confusion matrix of RoBERTa on the random sample of the NLBSE'22 test set.

| **Actual** | **Prediction** | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 2,247 (90%) | 190 (7%) | 65 (3%) |
| enhancement | 179 (9%) | 1,829 (89%) | 54 (2%) |
| question | 79 (18%) | 51 (12%) | 306 (70%) |

Table 22: Confusion matrix of Llama 3 8B on the random sample of the NLBSE'22 test set.

| **Actual** | **Prediction** | | | |
|---|---|---|---|---|
| | bug | enhancement | other | question |
| bug | 2,041 (82%) | 429 (17%) | 0 (0%) | 32 (1%) |
| enhancement | 247 (12%) | 1,767 (86%) | 1 (0%) | 47 (2%) |
| other | - | - | - | - |
| question | 145 (33%) | 246 (57%) | 0 (0%) | 45 (10%) |

Table 23: Confusion matrix of Llama 3 70B on the random sample of the NLBSE'22 test set.

| **Actual** | **Prediction** | | | |
|---|---|---|---|---|
| | bug | enhancement | other | question |
| bug | 2,180 (87%) | 263 (11%) | 23 (1%) | 36 (1%) |
| enhancement | 137 (7%) | 1,794 (87%) | 85 (4%) | 46 (2%) |
| other | - | - | - | - |
| question | 171 (39%) | 81 (19%) | 3 (1%) | 181 (41%) |

The performance of the three models—RoBERTa, Llama 3 8B, and Llama 3 70B—on the random sample taken from the GitHub'23 test set can be seen in Table 24, and the confusion matrices are provided in Table 25, Table 26, and Table 27, respectively.

First, the overall accuracy achieved by the RoBERTa model on the entire GitHub'23 test set (**.8827** as given in Table 17) and on the random sample of 5,000 issue reports selected from this dataset **(.8724)** were compared. The overall accuracy, along with the class-based precision, recall, and F1-score values obtained by the RoBERTa model on these two datasets were closely aligned. This result indicated that the random sample was representative of the larger population from which it was taken.

Similar to the results obtained with the random sample of the NLBSE'22 test set, the performance of the RoBERTa model was superior to that of the Llama 3 models. The class-based performance metrics achieved by both Llama 3 models were lower for all three classes compared with RoBERTa, except for the recall value of the bug class in the Llama 3 70B model. The Llama 3 8B and the 70B models achieved an overall accuracy of **.7454** and **.8276**, respectively.

Table 24: Performance of the models on the random sample of the GitHub'23 test set.

| **Metric** | **RoBERTa** | | | **Llama 3 8B** | | | **Llama 3 70B** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| bug | .8880 | .9166 | .9021 | .7902 | .8668 | .8268 | .8210 | .9368 | .8751 |
| enhancement | .8751 | .8631 | .8691 | .6825 | .7791 | .7276 | .8554 | .8239 | .8394 |
| question | .7818 | .6995 | .7384 | .6747 | .0951 | .1667 | .7594 | .3430 | .4725 |
| other* | - | - | - | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 |
| **Accuracy** | | **.8724** | | | **.7454** | | | **.8276** | |

* Added since some predictions made by the Llama 3 8B and 70B models were in this category.

Interestingly, the behaviors of the Llama 3 models were similar to each other across the two datasets. From the confusion matrices, we observed that the models did not classify too many bug and enhancement issue reports as questions, but they did classify some bug issues as enhancements, and vice versa. In terms of the question class, they mostly classified the issue reports as either bugs or enhancements. For example, at best, only 41% of the question issue reports were correctly classified as questions by the Llama 3 70B model on the random sample of the NLBSE'22 test set.

Table 25: Confusion matrix of RoBERTa on the random sample of the GitHub'23 test set.

| **Actual** | **Prediction** | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 2,450 (92%) | 153 (6%) | 70 (2%) |
| enhancement | 193 (11%) | 1,500 (86%) | 45 (3%) |
| question | 116 (20%) | 61 (10%) | 412 (70%) |

Table 26: Confusion matrix of Llama 3 8B on the random sample of the GitHub'23 test set.

| **Actual** | **Prediction** | | | |
|---|---|---|---|---|
| | bug | enhancement | other | question |
| bug | 2,317 (87%) | 338 (12%) | 0 (0%) | 18 (1%) |
| enhancement | 374 (22%) | 1,354 (78%) | 1 (0%) | 9 (0%) |
| other | - | - | - | - |
| question | 241 (41%) | 292 (50%) | 0 (0%) | 56 (9%) |

Table 27: Confusion matrix of Llama 3 70B on the random sample of the GitHub'23 test set.

| **Actual** | **Prediction** | | | |
|---|---|---|---|---|
| | bug | enhancement | other | question |
| bug | 2,504 (94%) | 134 (5%) | 5 (0%) | 30 (1%) |
| enhancement | 270 (16%) | 1,432 (82%) | 2 (0%) | 34 (2%) |
| other | - | - | - | - |
| question | 276 (47%) | 108 (18%) | 3 (1%) | 202 (34%) |

#### 5.1.2.2 Classification Confidence

Figure 4 presents the confidence levels in the classifications performed by the Llama 3 8B and Llama 3 70B models on the random samples taken from the NLBSE'22 and GitHub'23 test sets.

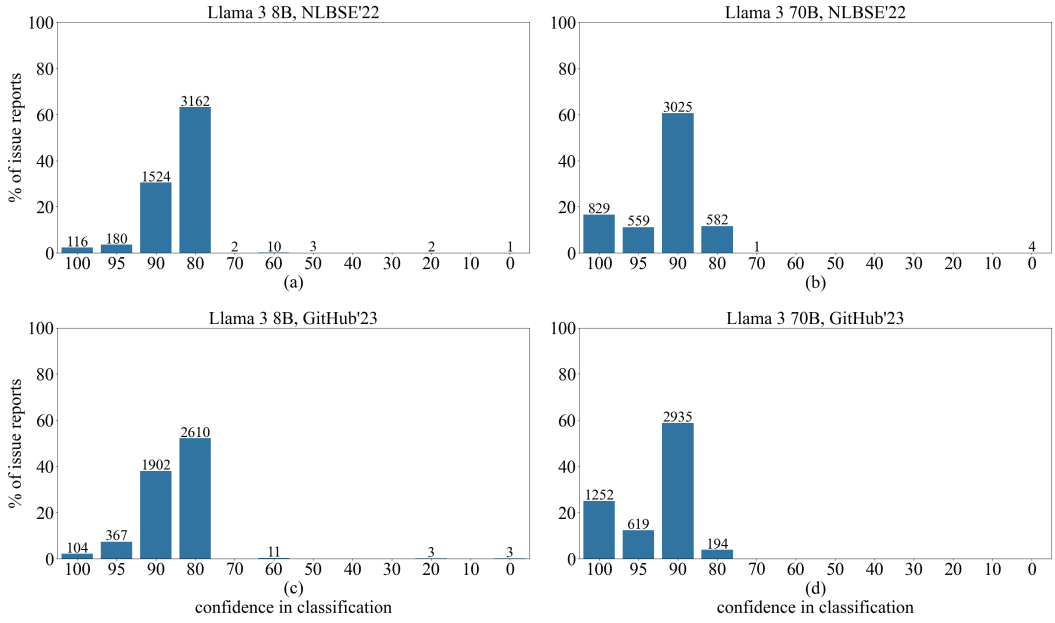


Figure 4: Confidence levels by the Llama 3 models on random samples.

In the random sample of the NLBSE'22 test set, only 18 issue reports were classified with a confidence level less than 80% by the Llama 3 8B model, and the mean confidence level was 84% (Figure 4.a). In the same dataset, the Llama 3 70B model provided a confidence level of less than 80% for only 5 issues, and the mean confidence level was 91% (Figure 4.b). The mean confidence level of the Llama 3 8B model on the random sample taken from the GitHub'23 test set was 85%, with 17 issue reports classified with a confidence level of less than 80% (Figure 4.c). The Llama 3 70B model classified all issue reports in this dataset with a confidence level of 80% or higher, reaching an average confidence level of 93% (Figure 4.d).

We compared the mean confidence levels of classifications when the model's classification matched the assigned issue label and when it did not, as shown in Table 28. Since the mean confidence levels were higher when the model's classification matched the assigned issue label, we aimed to statistically test whether this difference was significant.

Table 28: Mean confidence levels by the Llama 3 models on random samples.

| **Assigned vs. Predicted Label** | **NLBSE'22** | | **GitHub'23** | |
|---|---|---|---|---|
| | Llama 3 8B | Llama 3 70B | Llama 3 8B | Llama 3 70B |
| different | 81.11% | 87.69% | 82.03% | 90.90% |
| same | 84.80% | 91.65% | 86.27% | 93.12% |

**Hypothesis Testing**

Our aim was to statistically test whether the confidence levels were higher in classifications that matched the assigned issue labels compared to those that did not. We could have used a one-sided t-test [65] if the following assumptions were satisfied:

**1. Normality:** The data is approximately normally distributed.

**2. Equal variances:** The two groups have equal variances.

To test the normality assumption, we formulated the following hypotheses and applied the Shapiro-Wilk test [66]:

$H_0$ : Confidence levels are normally distributed.

$H_a$ : Confidence levels are not normally distributed.

The data of the confidence levels had the following statistics:

- For the classification levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 1147$, with a mean of $\bar{x} = 81.11$, a median of $80.0$, a standard deviation of $s = 5.78$, and a variance of $33.37$. The Shapiro-Wilk test yielded a test statistic of $W = 0.402$, with a $p - value$ of $0.000$.

- For the classification levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 3853$, with a mean of $\bar{x} = 84.80$, a median of $80.0$, a standard deviation of $s = 5.84$, and a variance of $34.14$. The Shapiro-Wilk test yielded a test statistic of $W = 0.723$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 845$, with a mean of $\bar{x} = 87.69$, a median of $90.0$, a standard deviation of $s = 7.52$, and a variance of $56.58$. The Shapiro-Wilk test yielded a test statistic of $W = 0.547$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 4155$, with a mean of $\bar{x} = 91.65$, a median of $90.0$, a standard deviation of $s = 5.47$, and a variance of $29.94$. The Shapiro-Wilk test yielded a test statistic of $W = 0.761$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **8B** model on the **GitHub'23** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 1273$, with a mean of $\bar{x} = 82.03$, a median of $80.0$, a standard deviation of $s = 7.25$, and a variance of $52.61$. The Shapiro-Wilk test yielded a test statistic of $W = 0.471$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **8B** model on the **GitHub'23** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 3727$, with a mean of $\bar{x} = 86.27$, a median of $90.0$, a standard deviation of $s = 5.91$, and a variance of $34.87$. The Shapiro-Wilk test yielded a test statistic of $W = 0.772$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **GitHub'23** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 862$, with a mean of $\bar{x} = 90.90$, a median of $90.0$, a standard deviation of $s = 5.04$, and a variance of $25.43$. The Shapiro-Wilk test yielded a test statistic of $W = 0.760$, with a $p-value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **GitHub'23** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 4138$, with a mean of $\bar{x} = 93.12$, a median of $90.0$, a standard deviation of $s = 4.85$, and a variance of $23.52$. The Shapiro-Wilk test yielded a test statistic of $W = 0.754$, with a $p-value$ of $0.000$.

Based on these results, we rejected the null hypothesis at $\alpha = 0.05$ and concluded that the confidence levels in the different groups were not normally distributed, with a $p-value$ of $0.000$. This $p-value$ indicated that the data was highly unlikely to be normally distributed, with extremely strong evidence against the null hypothesis of normality. Actually, this could also be observed from the histograms presented in Figure 4.

Since the data was not proven to be normally distributed, we decided to use the Mann-Whitney U test [67] instead of the t-test. The Mann-Whitney U test is a non-parametric statistical test used to

compare the differences between two independent groups. It is particularly useful when the normality assumption required for the t-test is not met, making it an appropriate alternative for analyzing ordinal or non-normally distributed continuous data. The test assesses whether one group tends to have higher or lower values than the other, without assuming a specific distribution of the data. We formulated the following hypothesis to apply the Mann-Whitney U test.

$H_0$ : The confidence levels in classifications that match the assigned issue labels are less than or equal to those in classifications that do not match.

$H_a$ : The confidence levels in classifications that match the assigned issue labels are greater than those in classifications that do not match.

The Mann-Whitney U tests yielded a test statistic of $U = 2882552.000$ and $U = 2333425.500$ in the classifications performed by the Llama 3 8B and Llama 3 70B models, respectively, on the NLBSE'22 dataset. On the GitHub'23 dataset, the Llama 3 8B and Llama 3 70B models yielded a test statistic of $U = 3180573.500$ and $U = 2160084.000$, respectively. For each case, a $p - value$ of $0.000$ was observed. Hence, at $\alpha = 0.05$, we rejected the null hypothesis and concluded that the confidence levels in classifications that match the assigned issue labels are greater than those in classifications that do not match.

#### 5.1.2.3 Issue Understandability

Figure 5 shows the understandability levels of issues as evaluated by the Llama 3 8B and Llama 3 70B models on the random samples taken from the NLBSE'22 and GitHub'23 test sets.

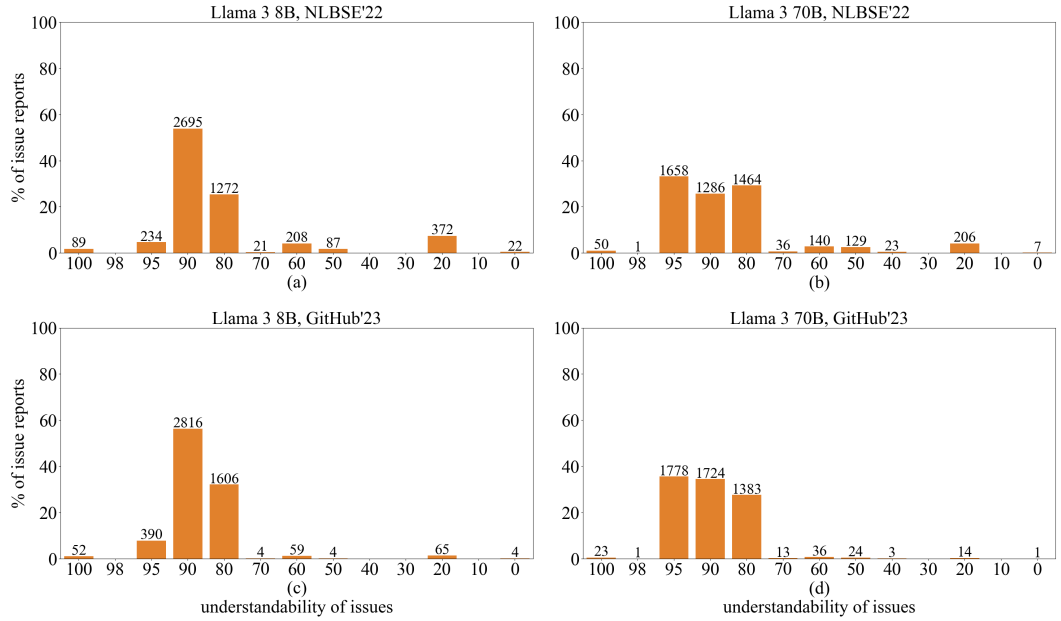


Figure 5: Understandability levels by the Llama 3 models on random samples.

The Llama 3 8B model assessed 86% of the issues in the random sample of the NLBSE'22 dataset as having an understandability level of 80% or higher, and the mean understandability level was 80% (Figure 5.a). The mean understandability level was 84% according to the Llama 3 70B model on the same dataset, with 541 issue reports assigned an understandability level lower than 80% (Figure 5.b). The average understandability level of issues in the random sample taken from the GitHub'23 test set was 86% and 88%, based on the results of the Llama 3 8B (Figure 5.c) and 70B (Figure 5.d) models, respectively.

The mean understandability levels of the issue reports, for both when the model's classification matched the assigned issue label and when it did not, are reported in Table 29. We noticed that the mean understandability levels provided by the Llama 3 70B model for issues classified the same as the assigned issue labels were higher than for those classified differently in both datasets, reflecting the pattern observed in the confidence levels. In contrast, for the Llama 3 8B model, the mean understandability levels slightly decreased for issues classified the same as the assigned issue labels compared to those classified differently in both datasets.

Table 29: Mean understandability levels by the Llama 3 models on random samples.

| **Assigned vs. Predicted Label** | **NLBSE'22** | | **GitHub'23** | |
|---|---|---|---|---|
| | Llama 3 8B | Llama 3 70B | Llama 3 8B | Llama 3 70B |
| different | 80.32% | 77.81% | 87.05% | 87.41% |
| same | 80.21% | 84.75% | 85.50% | 88.55% |

We tested the normality assumption to use the t-test with the following hypothesis:

$H_0$ : Understandability levels are normally distributed.

$H_a$ : Understandability levels are not normally distributed.

The data of the understandability levels had the following statistics:

- For the understandability levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 1147$, with a mean of $\bar{x} = 80.32$, a median of $90.0$, a standard deviation of $s = 21.36$, and a variance of $456.09$. The Shapiro-Wilk test yielded a test statistic of $W = 0.552$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 3853$, with a mean of $\bar{x} = 80.21$, a median of $90.0$, a standard deviation of $s = 19.59$, and a variance of $383.80$. The Shapiro-Wilk test yielded a test statistic of $W = 0.611$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 845$, with a mean of $\bar{x} = 77.81$, a median of $90.0$, a standard deviation of $s = 24.66$,

and a variance of $608.18$. The Shapiro-Wilk test yielded a test statistic of $W = 0.683$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 4155$, with a mean of $\bar{x} = 84.75$, a median of $90.0$, a standard deviation of $s = 14.80$, and a variance of $219.03$. The Shapiro-Wilk test yielded a test statistic of $W = 0.659$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **8B** model on the **GitHub'23** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 1273$, with a mean of $\bar{x} = 87.05$, a median of $90.0$, a standard deviation of $s = 9.44$, and a variance of $89.11$. The Shapiro-Wilk test yielded a test statistic of $W = 0.401$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **8B** model on the **GitHub'23** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 3727$, with a mean of $\bar{x} = 85.50$, a median of $90.0$, a standard deviation of $s = 10.20$, and a variance of $104.07$. The Shapiro-Wilk test yielded a test statistic of $W = 0.566$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **70B** model on the **GitHub'23** dataset where the model classified the issue with a label **different** from the assigned one, the sample size was $n = 862$, with a mean of $\bar{x} = 87.41$, a median of $90.0$, a standard deviation of $s = 9.40$, and a variance of $88.31$. The Shapiro-Wilk test yielded a test statistic of $W = 0.688$, with a $p - value$ of $0.000$.

- For the understandability levels assessed by the Llama 3 **70B** model on the **GitHub'23** dataset where the model classified the issue with a label **same** as the assigned one, the sample size was $n = 4138$, with a mean of $\bar{x} = 88.55$, a median of $90.0$, a standard deviation of $s = 7.85$, and a variance of $61.60$. The Shapiro-Wilk test yielded a test statistic of $W = 0.691$, with a $p - value$ of $0.000$.

Once again, we rejected the null hypothesis at $\alpha = 0.05$ and concluded that the understandability levels in the different groups were not normally distributed with a $p - value$ of $0.000$ in both datasets. Similarly, this could also be observed from the histograms presented in Figure 5.

Next, we formulated the following hypothesis to apply the Mann-Whitney U test.

$H_0$ : The understandability levels of the issue reports classified the same as the assigned issue labels are less than or equal to those classified differently.

$H_a$ : The understandability levels of the issue reports classified the same as the assigned issue labels are greater than those classified differently.

With a $p - value$ greater than $\alpha = 0.05$, we fail to reject the null hypothesis and assume that the understandability levels of the issues classified the same as the assigned issue labels are less than or equal to those classified differently. Otherwise, we can reject the null hypothesis and conclude that the

understandability levels of the issues classified the same as the assigned issue labels are greater than those classified differently. Based on the results, we could reject the null hypothesis except for the responses generated by the Llama 3 8B model on both the NLBSE'22 and GitHub'23 datasets. In the Mann-Whitney U test, the following test statistics and $p - values$ were obtained:

- $U = 2016820.500$ and $p - value = 1.000$ with model: Llama 3 8B, dataset: NLBSE'22
- $U = 1904035.000$ and $p - value = 0.000$ with model: Llama 3 70B, dataset: NLBSE'22
- $U = 2077475.500$ and $p - value = 1.000$ with model: Llama 3 8B, dataset: GitHub'23
- $U = 1871431.500$ and $p - value = 0.008$ with model: Llama 3 70B, dataset: GitHub'23

### 5.2 Reliability Analysis of Labels

In the reliability analysis of issue labels, we investigated to what extent the assigned labels of the issue reports are relevant to their contents. To this end, we focused on the issue reports for which at least one of the three models—RoBERTa, Llama 3 8B, or Llama 3 70B—classified the issue with a label different from the assigned one. We manually classified issue reports based solely on the issue content, involving the issue title and the issue body, as these two describe the issue content written by its author. To analyze the issue reports, we combined the assigned issue label and the classifications performed by the models in a newly introduced "case" label for each issue report, which encoded four labels assigned according to:

- **1st letter** denoted the **assigned issue label**
- **2nd letter** denoted the prediction made by **RoBERTa**
- **3rd letter** denoted the classification performed by **Llama 3 8B**
- **4th letter** denoted the classification performed by **Llama 3 70B**

Each issue label was encoded a single letter as follows:

- **bug** was denoted by **b**
- **enhancement** was denoted by **e**
- **question** was denoted by **q**
- **other** was denoted by **o**

The assigned issue labels involved the following three issue labels: bug, enhancement, and question. Since RoBERTa was trained on these originally assigned issue labels, its predictions also included these three labels. However, we requested the Llama 3 8B and 70B models to classify issues into the most suitable category by providing four options: bug, enhancement, question, or other. As a result, the total number of possible label combinations that corresponded to distinct case labels was:

$$Total\ \#\ of\ combinations = 3 \cdot 3 \cdot 4 \cdot 4 = 144 \tag{5.1}$$

Out of the 144 possible distinct case labels, three of them ("bbbb", "eeee", "qqqq") corresponded to the cases where the assigned issue label and the classifications made by the three models were all the same. For example, "bbbb" corresponded to the issue reports that were assigned the label "bug" and also classified as "bug" by all three models. After removing these three cases, 141 possible distinct combinations remained that required closer investigation. Among them, we observed 74 distinct case labels in 1,541 issue reports, for which at least one of the models classified the issue with a label different from the assigned one. The occurrences of these combinations, along with their count and percentage values, are listed in Appendix D. These 1,541 issues were manually classified based on the issue title and the issue body, as explained in Section 4.3. The results of the manual classification can be summarized as follows:

**1.** The initial number of issue reports included in the dataset = 5,000.
(Set $I$, stands for "Issues")

**2.** The number of issue reports for which the assigned issue label and the classifications made by the models were all the same = 3,459.
(Subset $S$, stands for "Same", note that $S \subseteq I$)

**3.** The number of issue reports for which at least one of the models classified the issue with a label different from the assigned issue label = 1,541.
(Subset $D$, stands for "Different", note that $D \subseteq I$ and $S \cup D = I$)

**4.** The number of issue reports in Set *D* for which the manually classified issue label and the assigned issue label were the same = 747.
(Subset $A$, stands for "as Assigned", note that $A \subseteq D$)

**5.** The number of issue reports for which the assigned issue label was considered to be plausible = 4,206.
(Subset $P$, stands for "Plausible", note that $S \cup A = P$)

**6.** The number of issue reports in Set *D* for which manual classification could not be performed due to insufficient information in the issue title and the issue body = 49.
(Subset $U$, stands for "Unclear", note that $U \subseteq D$)

**7.** The number of issue reports in Set *D* for which manual classification could not be performed due to the issue title and the issue body being in a language other than English = 189.
(Subset $L$, stands for "other Language", note that $L \subseteq D$)

**8.** The number of issue reports in Set *D* for which the manually classified issue label and the assigned issue label were not the same = 556.
(Subset $Q$, stands for "Questionable", note that $Q \subseteq D$ and $A \cup U \cup L \cup Q = D$)

The assigned issue labels were considered plausible in 84% of the issue reports ($S \cup A = P$). For these issue reports, either all three models (RoBERTa, Llama 3 8B, and Llama 3 70B) classified the issue with the same label as the assigned one, or we manually classified the issue with a label identical to the assigned issue label.

However, for the remaining 16% of the issue reports ($U \cup L \cup Q$), either the assigned issue label was questionable, or it was not possible for the models to perform a meaningful classification (either due to the insufficient context of the issue report or the issue report being in a language other than English). In the latter case, approximately 1% of the issue reports, corresponding to 49 issue reports, lacked sufficient information to perform a meaningful manual classification. Examples of such issue reports are illustrated in Figure 6 and Figure 7.

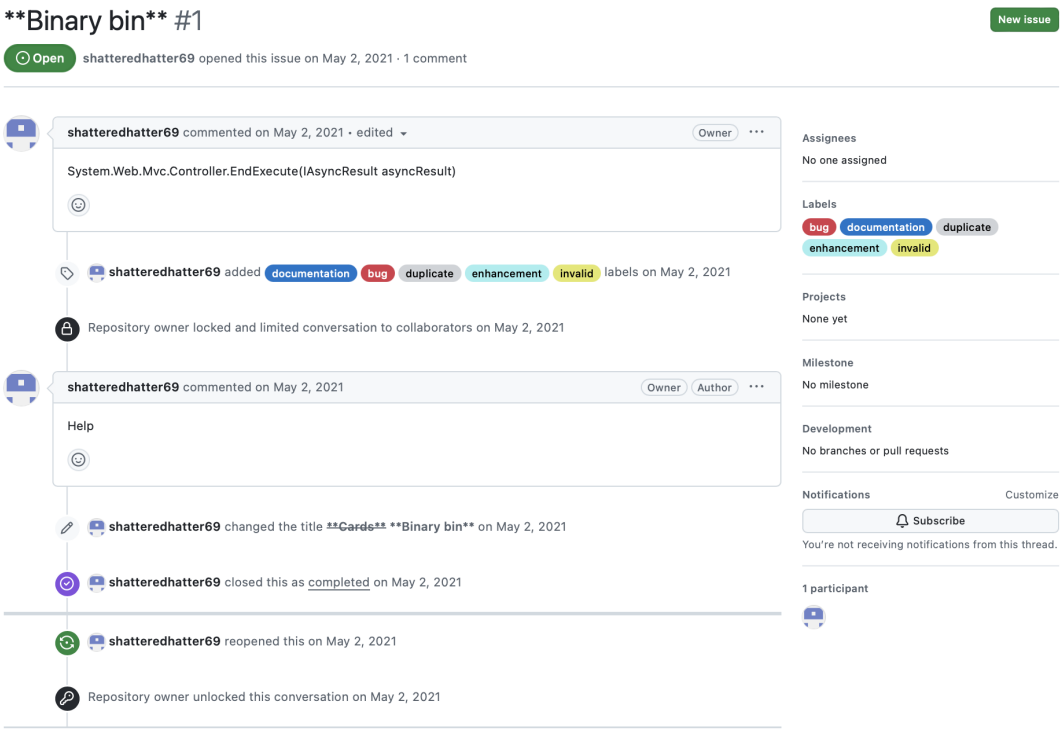


Figure 6: An issue report labeled as enhancement, manually classified as unclear.

Both of these issue reports were created by the owner of the projects, which may be why they included minimal information. The distribution of the issue author association types in the issue reports manually classified as unclear ($U$) is reported in Table 30.

Table 30: The distribution of the issue author association types in unclear issue reports ($U$).

| **Association** | **# of issues** | **% of issues** |
|---|---|---|
| owner | 25 | 51 |
| none | 16 | 33 |
| contributor | 5 | 10 |
| collaborator | 2 | 4 |
| member | 1 | 2 |
| Total | 49 | 100 |

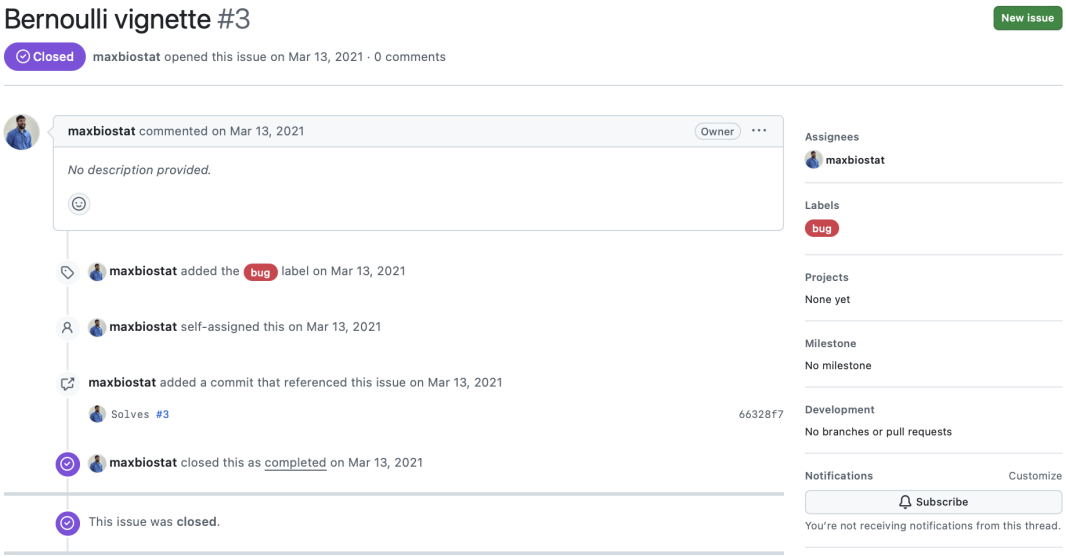


Figure 7: An issue report labeled as bug, manually classified as unclear.

Approximately 4% of the issue reports, corresponding to 189 issues, consisted of text written in a language other than English in the issue content, involving the issue title and the issue body. As the RoBERTa [19] and Llama 3 models are mostly trained on English text[17], the performance of these models is optimized for English. Consequently, using a dataset that includes mixed languages may lead to inaccurate predictions or misclassifications for these instances.

Based on our observations, the assigned issue labels of **11%** of the issue reports ($Q$) were indeed questionable. We believe that these issue reports should have a label that is different from the assigned one based on their contents. The confusion matrix of the assigned issue labels and the labels that were determined during manual classification for these questionable issue reports is presented in Table 31.

Table 31: Confusion matrix of the questionable issue reports ($Q$).

| **Actual** | **Manual Classification** | | | | **Total** |
|---|---|---|---|---|---|
| | bug | enhancement | other | question | |
| bug | - | 174 (77%) | 26 (11%) | 27 (12%) | 227 |
| enhancement | 90 (54%) | - | 44 (26%) | 34 (20%) | 168 |
| other | - | - | - | - | - |
| question | 98 (61%) | 55 (34%) | 8 (5%) | - | 161 |
| Total | 188 | 229 | 78 | 61 | 556 |

In manual classification, 556 issue reports (out of 5,000) were identified as questionable, of which 227 had an assigned issue label of bug, 168 were labeled as enhancement, and the remaining 161 belonged to the question class, originally. In the random sample of the NLBSE'22 test set, there were 2,502, 2,062, and 436 issue reports assigned the labels bug, enhancement, and question, respectively,

[17] https://ai.meta.com/blog/meta-llama-3/

as shown in Table 15. This means that 9% (227), 8% (168), and 37% (161) of the issue reports with the assigned issue labels bug, enhancement, and question, respectively, were classified with a different label in manual classification.

From the confusion matrix provided in Table 31, we observe the distribution of these differences for the questionable issue reports.

- Among 227 issue reports in the bug class, 77% were classified as enhancement during manual classification. This may be due to the fact that not all users and developers have the same level of experience and knowledge about the project. Without having a comprehensive requirements specification document, it is difficult to distinguish if what is described in the issue report is related to something that should already be implemented or a new request.

- 54% of the questionable issue reports with an assigned issue label of enhancement were labeled as bug based on the issue content. Again, this may be due to the lack of a requirements specification document. 26% of these issue reports were classified as other; that is, they were not related to a label of bug, enhancement, or question but something else. We observed a variety of categories under the other class, including guidelines, notifications, tasks, and tests. Finally, the remaining 20% of the issue reports were labeled as question in manual classification. The enhancement class corresponds to new feature requests, improvements, and suggestions. As observed during manual classification, some issue reports are created just to ask whether specific functionality or integration with particular software or hardware is supported, or to learn about the current behavior of the software product, rather than explicitly requesting a new feature. We suspect that these issue reports tend to be quite similar, with only minor distinctions.

- Regarding the questionable issue reports, with an assigned issue label of question, 61% of them were labeled as bug in manual classification. In the reliability analysis, we observed that some issues labeled as question actually report inconsistent behavior of the software or missing code, thus resembling the structure and content of bug reports. Often, questions contain an error message, which is also common in bug reports. As a result, it is difficult to distinguish between cases where the user is merely asking a question (possibly due to user error, integration of the software with third party software or hardware, and even a non-existing functionality) and cases where there is indeed a bug in the software system that needs to fixed. It is challenging for an LLM to understand and identify these differences, especially considering there is no information given to the models about the project's functionalities.

- In addition to these scenarios, we also observed some misclassifications that cannot be explained by a systematic characteristic but rather by the fact that users and developers perceive some issues differently. As they have different backgrounds and experiences, the issue labeling rationale of the OSS community members may vary, leading to the use of different labels for the same type of situations.

We compared the confusion matrix of the manual classification with the ones obtained by three models, RoBERTa, Llama 3 8B, and Llama 3 70B on the random sample of the NLBSE'22 test set which are presented in Table 21, Table 22, and Table 23, respectively.

Our findings can be summarized as follows:

- **RoBERTa:** If there is a misclassification within the bug class, it predominantly results in an enhancement label rather than being classified as a question. The opposite is also true; when there is a misclassification within the enhancement class, the model tends to classify it as a bug rather than a question. Regarding the question class, when misclassifications occur, the model is slightly more inclined to classify them as bugs rather than enhancements, although the difference is not substantial (18% and 12%, respectively).

- **Llama 3 8B:** The misclassification patterns of the Llama 3 8B model closely mirror those of RoBERTa for the bug and enhancement classes. However, in contrast to RoBERTa, when there is a misclassification within the question class, the Llama 3 8B model is more likely to classify it as an enhancement (57%) rather than a bug (33%).

- **Llama 3 70B:** For the bug class, the misclassification pattern is the same as those of RoBERTa and Llama 3 8B. Regarding the enhancement class, when misclassifications occur, the model is slightly more inclined to classify them as the bug class rather than other, although the difference is not significant (7% and 4%, respectively). In terms of the question class, like RoBERTa, the model is slightly more inclined to classify them as bugs rather than enhancements when misclassifications occur.

- **Manual Classification:** Among the three models, the pattern we observed in manual classification most closely resembles that of Llama 3 70B. Based on our results, a bug report should actually be labeled as an enhancement most of the time, if its issue content is not relevant to its assigned issue label, bug. Regarding the enhancement class, 54%, 26%, and 20% of the issues are manually labeled as bug, other, and question, respectively, when there is a mismatch between the issue content and its label. Finally, in the question class, the issues are manually labeled as bugs rather than enhancements most of the time, if the issue label is identified as irrelevant.

In the following, we report and comment on some representative examples of these issue reports. Specifically, we demonstrate patterns observed in the confusion matrices; that is, issue reports labeled as bugs and enhancements were generally not classified as questions. However, some bug reports were classified as enhancements and vice versa. Regarding the question class, issues were mostly classified as bugs, except for the Llama 3 8B model, which classified most of these issues as enhancements.

The example in Figure 8 illustrates an issue labeled as a bug. A user of the project states that the Debian package currently has a strict dependency on Alacritty as the terminal emulator, even though users might prefer a different terminal emulator. The requested change is to make Alacritty an optional or flexible dependency rather than a strict one, allowing the Debian package to only require Neovim as a mandatory dependency. Clearly, the author of the issue does not report any unexpected behavior of the software. The current dependency setup is actually functioning as intended, so this is not a bug. The request is to make the package more adaptable to different user environments and preferences, which corresponds to the exact meaning of the enhancement label. All three models classified this issue as an enhancement, which is incorrect in terms of the issue classification problem, as the assigned label of the issue report is bug. However, their classifications are actually correct in terms of the issue content.




Figure 8: An issue report labeled as bug, manually classified as enhancement.

On the other hand, the issue report presented in Figure 9 is assigned with a label of enhancement. However, the content and structure of the issue are typical of a bug. Indeed, this is a well-written bug report that describes the actual and expected behavior of the software, steps to reproduce the misbehavior, and the specific version used. All three models classified this issue as a bug, which is consistent with the issue content. Although it may turn out that there is no bug in the software product and the user might be missing some details, considering only the contextual information—without knowing the details of the project—the issue label that best matches this issue content is bug, as classified by the LLMs and by us in the manual classification.

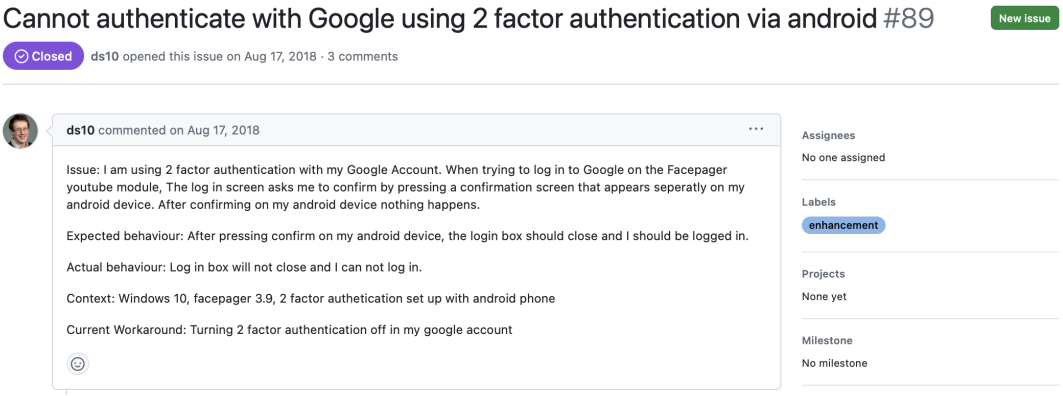


Figure 9: An issue report labeled as enhancement, manually classified as bug.

Figures 10 and 11 depict an issue report labeled as question. The author of the issue describes the problem and provides all necessary information that should be involved in a typical bug report (the steps to reproduce the misbehavior, the actual and expected behavior, and the specific version used). The issue is initially labeled as a bug by its author. On the same day, one of the contributors of the project starts handling the issue, and adds a comment explaining the current behavior of the software to clarify the situation for the user and removes the bug label. If the problem were not the expected behavior of the software but rather the result of a bug, then the issue text would have been the same, as it currently involves all information related to the problem. In this case, the contributor would not have removed the bug label. This example demonstrates how the difference between questions and

bugs might be subtle and challenging to be identified by LLMs no matter how capable they are. All three models classified this issue as a bug, which is consistent with the issue content.

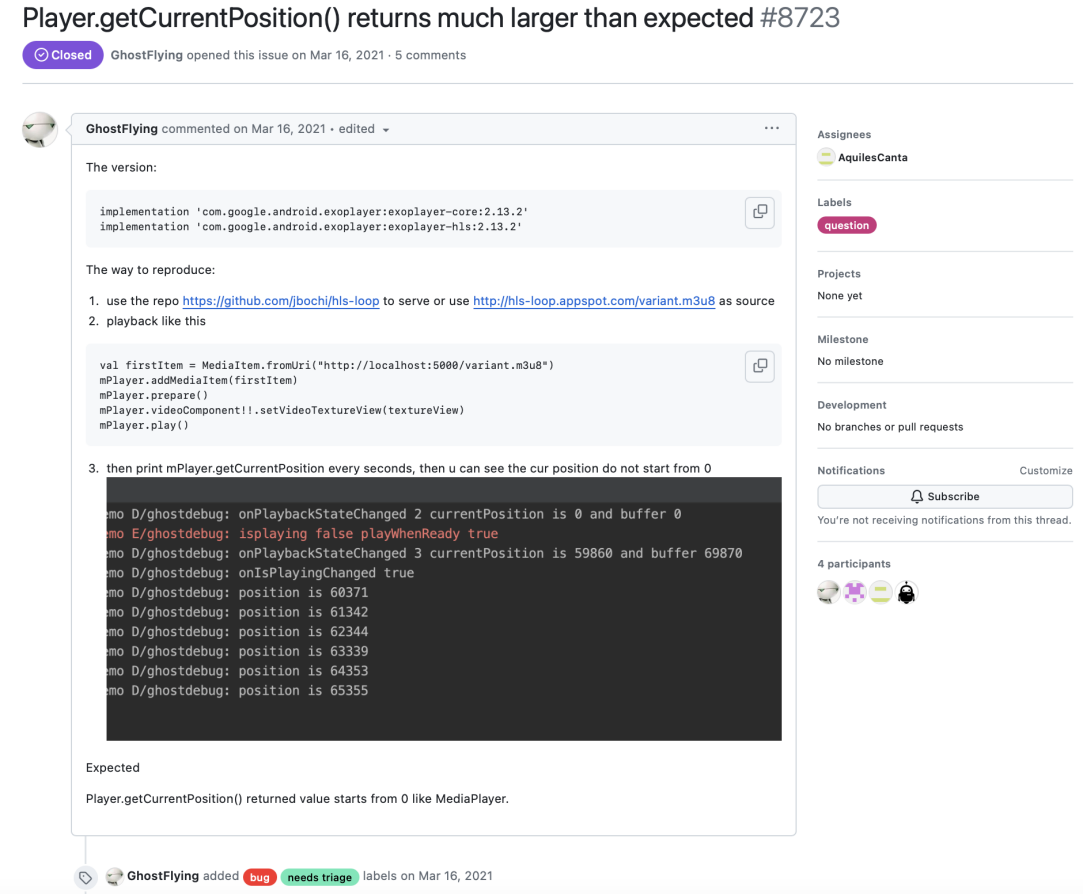


Figure 10: An issue report labeled as question, manually classified as bug.




Figure 11: The answer of the contributor to the issue depicted in Figure 10.

Another issue labeled as question and its answer can be seen in Figure 12. The author of the issue explicitly suggests adding a new feature and describes the desired behaviour. After reviewing it, one of the contributors of the project mentions that the requested feature is already supported by the software

and adds the question label before closing the issue. However, based on the issue content, this issue should be classified as an enhancement. When even the users of the software are not familiar with all the functionalities supported by the software application, it seems not fair to expect an LLM to identify these differences. For this issue, the classification performed by RoBERTa was question, while the Llama 3 models classified it as an enhancement, which is more related to the issue content.

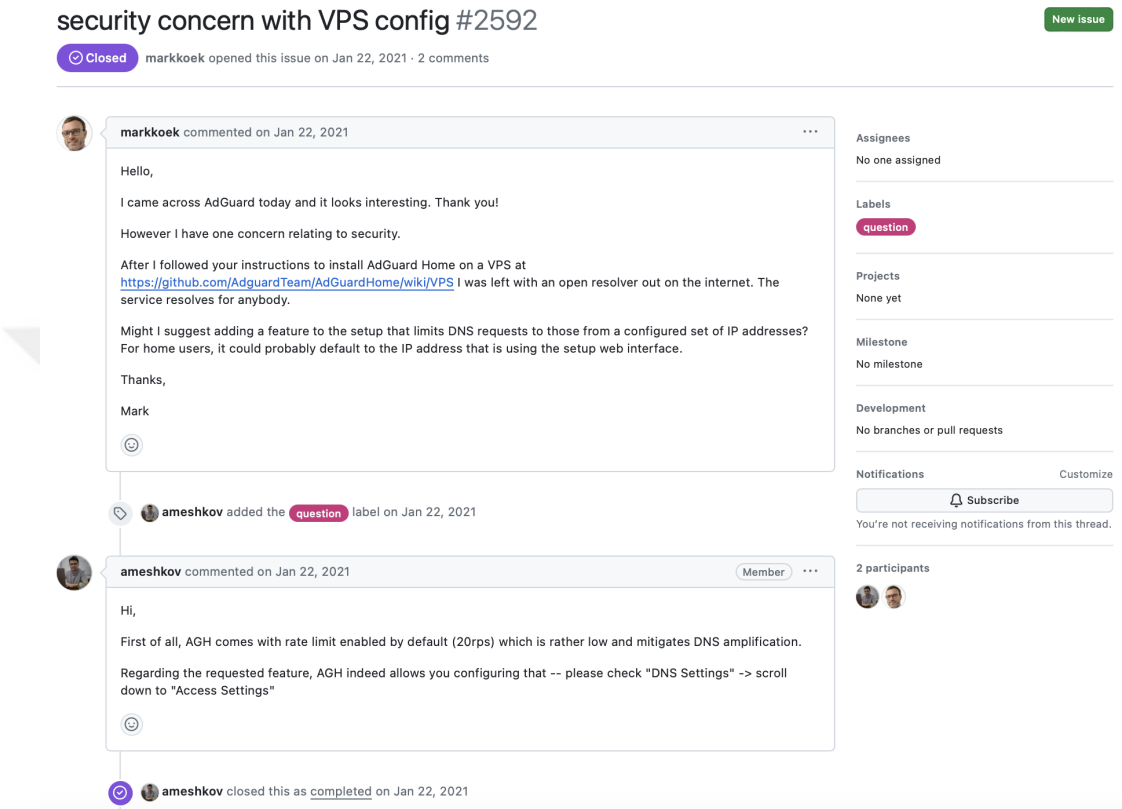


Figure 12: An issue report labeled as question, manually classified as enhancement.

After these motivating examples, we compared the performance of the three models with and without the inclusion of the problematic issue reports (i.e., $U \cup L \cup Q$ vs. $P$), constituting 16% of the whole sample. We used the same random sample, as it was previously observed to be representative of the larger population from which it was drawn, as explained in Section 5.1.2.1. The number of distinct labels in this dataset are provided in Table 32. The number of issue reports having bug, enhancement, and question labels decreased by 14%, 13%, and 43%, respectively, after excluding the problematic issue reports.

Table 32: The number of distinct labels in plausible issue reports ($P$).

| **Label** | **Count** |
|---|---|
| bug | 2,159 |
| enhancement | 1,800 |
| question | 247 |
| Total | 4,206 |

The RoBERTa model was trained on the training set without any changes. However, it was evaluated on the subset $P$ of the random sample of the NLBSE'22 test set to compare the performance of the model with or without including the problematic issue reports. Regarding the Llama 3 models, their responses on the plausible issue reports were evaluated. The performance of the three models, RoBERTa, Llama 3 8B and Llama 3 70B, on the random sample taken from the NLBSE'22 test set with excluding the problematic issues reports are shown in Table 33 and the confusion matrices are reported in Table 34, Table 35, and Table 36, respectively.

Table 33: Performance of the models on plausible issue reports ($P$).

| **Metric** | **RoBERTa** | | | **Llama 3 8B** | | | **Llama 3 70B** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| bug | .9547 | .9560 | .9553 | .9068 | .9009 | .9038 | .9544 | .9699 | .9621 |
| enhancement | .9642 | .9428 | .9534 | .8161 | .9094 | .8602 | .9571 | .9672 | .9621 |
| question | .7641 | .8785 | .8173 | .6364 | .1417 | .2318 | .8901 | .6559 | .7552 |
| other* | - | - | - | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 |
| **Accuracy** | | **.9458** | | | **.8600** | | | **.9503** | |

* Added since some predictions made by the Llama 3 70B model were in this category.

Table 34: Confusion matrix of RoBERTa on plausible issue reports ($P$).

| **Actual** | **Prediction** | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 2,064 (96%) | 49 (2%) | 46 (2%) |
| enhancement | 82 (5%) | 1,697 (94%) | 21 (1%) |
| question | 16 (6%) | 14 (6%) | 217 (88%) |

Table 35: Confusion matrix of Llama 3 8B on plausible issue reports ($P$).

| **Actual** | **Prediction** | | |
|---|---|---|---|
| | bug | enhancement | question |
| bug | 1,945 (90%) | 206 (10%) | 8 (0%) |
| enhancement | 151 (8%) | 1,637 (91%) | 12 (1%) |
| question | 49 (20%) | 163 (66%) | 35 (14%) |

Table 36: Confusion matrix of Llama 3 70B on plausible issue reports ($P$).

| **Actual** | **Prediction** | | | |
|---|---|---|---|---|
| | bug | enhancement | other | question |
| bug | 2,094 (97%) | 55 (3%) | 2 (0%) | 8 (0%) |
| enhancement | 38 (2%) | 1,741 (97%) | 9 (0%) | 12 (1%) |
| other | - | - | - | - |
| question | 62 (25%) | 23 (9%) | 0 (0%) | 162 (66%) |

With the elimination of the questionable issue reports, the performances of all models were improved. All models obtained better results in class-based precision, recall, and F1-score values over all three classes and overall accuracy. The accuracy of the RoBERTa, Llama 3 8B, and Llama 3 70B models improved by **8%**, **12%**, and **14%**, respectively, compared with the results obtained on the entire random sample (reported previously in Table 20). The best performing model in terms of overall accuracy was Llama 3 70B, followed by RoBERTa with a slight decrease in accuracy. In both models, it seemed that there was still room for improvement in the question class.

Inspired by these improvements, we statistically tested whether the confidence levels of the classifications and the understandability levels of the issue reports, as assessed by the Llama 3 models, were higher for the issue reports in the subset $P$ compared to those in the subset $Q$ from the random sample of the NLBSE'22 test set. Specifically, the subset $P$ represented the issue reports with plausible assigned issue labels. Out of 5,000 issue reports, we identified 4,206 issue reports having plausible issue labels, while the assigned issue labels of 556 issue reports were indeed considered questionable based on our observations.

**Classification Confidence**

The normality assumption was tested with the Shapiro-Wilk test [66] that was introduced previously, with the following hypothesis:

$H_0$ : Confidence levels are normally distributed.

$H_a$ : Confidence levels are not normally distributed.

The data of the confidence levels had the following statistics:

- For the classification levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the issues were considered **plausible**, the sample size was $n = 4206$, with a mean of $\bar{x} = 84.32$, a median of $80.0$, a standard deviation of $s = 6.12$, and a variance of $37.41$. The Shapiro-Wilk test yielded a test statistic of $W = 0.688$, with a $p - value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the issues were considered **questionable**, the sample size was $n = 556$, with a mean of $\bar{x} = 82.19$, a median of $80.0$, a standard deviation of $s = 4.94$, and a variance of $24.37$. The Shapiro-Wilk test yielded a test statistic of $W = 0.526$, with a $p - value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the issues were considered **plausible**, the sample size was $n = 4206$, with a mean of $\bar{x} = 91.60$, a median of $90.0$, a standard deviation of $s = 5.32$, and a variance of $28.30$. The Shapiro-Wilk test yielded a test statistic of $W = 0.793$, with a $p - value$ of $0.000$.
- For the classification levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the issues were considered **questionable**, the sample size was $n = 556$, with a mean of $\bar{x} = 88.36$, a median of $90.0$, a standard deviation of $s = 8.98$, and a variance of $80.56$. The Shapiro-Wilk test yielded a test statistic of $W = 0.418$, with a $p - value$ of $0.000$.

Having $p - values$ of $0.000$ indicated that the confidence levels obtained by both models, in two groups were not normally distributed at $\alpha = 0.05$. Next, we formulated the following hypothesis to

statistically test whether the confidence levels associated with the issue reports considered plausible were higher than those associated with questionable issue reports. As previously, we applied a Mann-Whitney U test.

$H_0$ : The confidence levels in classifications of issues considered plausible are less than or equal to those considered questionable.

$H_a$ : The confidence levels in classifications of issues considered plausible are greater than those considered questionable.

With a $p-value$ greater than $\alpha = 0.05$, we fail to reject the null hypothesis and assume that the confidence levels in classifications of issues considered plausible are less than or equal to those considered questionable. However, we rejected the null hypothesis and concluded that the confidence levels in classifications of issues considered plausible are greater than those considered questionable, based on the following results:

- $U = 1399642.000$ and $p - value = 0.000$ with model: Llama 3 8B, dataset: NLBSE'22
- $U = 1465554.500$ and $p - value = 0.000$ with model: Llama 3 70B, dataset: NLBSE'22

**Issue Understandability**

We tested the normality assumption to use the t-test with the following hypothesis:

$H_0$ : Understandability levels are normally distributed.

$H_a$ : Understandability levels are not normally distributed.

The data of the understandability levels had the following statistics:

- For the understandability levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the issues were considered **plausible**, the sample size was $n = 4206$, with a mean of $\bar{x} = 81.16$, a median of 90.0, a standard deviation of $s = 18.42$, and a variance of 339.42. The Shapiro-Wilk test yielded a test statistic of $W = 0.600$, with a $p - value$ of 0.000.
- For the understandability levels assessed by the Llama 3 **8B** model on the **NLBSE'22** dataset where the issues were considered **questionable**, the sample size was $n = 556$, with a mean of $\bar{x} = 79.99$, a median of 90.0, a standard deviation of $s = 22.63$, and a variance of 512.03. The Shapiro-Wilk test yielded a test statistic of $W = 0.538$, with a $p - value$ of 0.000.
- For the understandability levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where issues were considered **plausible**, the sample size was $n = 4206$, with a mean of $\bar{x} = 84.88$, a median of 90.0, a standard deviation of $s = 14.84$, and a variance of 220.22. The Shapiro-Wilk test yielded a test statistic of $W = 0.654$, with a $p - value$ of 0.000.
- For the understandability levels assessed by the Llama 3 **70B** model on the **NLBSE'22** dataset where the issues were considered **questionable**, the sample size was $n = 556$, with a mean of

$\bar{x} = 81.98$, a median of 90.0, a standard deviation of $s = 21.29$, and a variance of 453.20. The Shapiro-Wilk test yielded a test statistic of $W = 0.620$, with a $p - value$ of 0.000.

At $\alpha = 0.05$, we rejected the null hypothesis and concluded that the understandability levels in the different groups were not normally distributed with a $p - value$ of 0.000, regarding both Llama 3 models. Next, we formulated the following hypothesis to apply the Mann-Whitney U test.

$H_0$ : The understandability levels of the issues considered plausible are less than or equal to those considered questionable.

$H_a$ : The understandability levels of the issues considered plausible are greater than those considered questionable.

Based on the below results, we failed to reject the null hypothesis and assumed that the understandability levels of the issues considered plausible are less than or equal to those considered questionable, at $\alpha = 0.05$, regarding both Llama 3 models:

- $U = 1097105.000$ and $p - value = 0.996$ with model: Llama 3 8B, dataset: NLBSE'22
- $U = 1157667.000$ and $p - value = 0.655$ with model: Llama 3 70B, dataset: NLBSE'22

**Issue Author Association**

Finally, we compared the distribution of the issue author association types observed in the plausible and questionable issue reports. Based on the results reported in Table 37, there seemed to be no significant difference.

Table 37: The distribution of the issue author association types in plausible ($P$) and questionable ($Q$) issue reports.

| **Author** | **Plausible Issue Reports** | | **Questionable Issue Reports** | |
|---|---|---|---|---|
| | # of issues | % of issues | # of issues | % of issues |
| none | 1,539 | 37 | 263 | 47 |
| owner | 929 | 22 | 102 | 18 |
| contributor | 674 | 16 | 91 | 17 |
| member | 535 | 13 | 51 | 9 |
| collaborator | 529 | 12 | 49 | 9 |
| Total | 4,206 | 100 | 556 | 100 |

# CHAPTER 6

# DISCUSSION

This thesis aims to investigate the reliability of the issue labels used in OSS development projects. To achieve this, the first research question explored the extent to which the labels assigned to issue reports created in OSS development projects are reliable. The second research question investigated the effectiveness of state-of-the-art open source LLMs in issue label classification.

Regarding issue label classification, researchers have proposed automated approaches to predict the label that should be assigned to a new issue. However, a few studies have investigated concerns about the quality of data, particularly regarding misclassified issue labels.

In 2008, Antoniol et al. [16] highlighted the problem of misclassified issue reports. They collected a random sample of 1,800 bug reports collected from Bugzilla and Jira. Through manual classification, they observed that only 45% of the bug reports were actually related to the bug class, while 38% were referred to enhancements. The remaining 17% were classified as other, indicating that these bug reports did not fit into the categories of either bug or enhancement. After removing the bug reports which were manually classified as other, they experimented with a Naive Bayes classifier, decision trees, and logistic regression to automatically classify issue labels as bug and non-bug, based on the corpus they manually created. The best performance in overall accuracy was achieved with logistic regression in the experimented projects, with values ranging from 77% to 82%. In our analysis, we did not focus on only bug reports, but included the following labels in our analysis: bug, enhancement, and question. We utilized state-of-the-art open source LLMs in issue label classification on a random sample of 5,000 issue reports collected from the OSS projects hosted on GitHub and after manual classification, we identified that 11% of the issue reports actually had questionable issue labels. In this set of questionable issue reports, there were 227 issue reports which were originally labeled as bug. Based on our manual classification, 77% of them had an issue content associated with an enhancement, 12% of them were related to the class of question, and we categorized the remaining 11% as other.

In 2013, Herzig et al. [17] manually investigated 7,401 issue reports collected from Bugzilla and Jira, where issue reports were generally labeled under a single category, bug. They found that more than 40% of the total issue reports were misclassified, with 33.8% of all bug reports not referring to corrective activities based on manual classification. Compared to Bugzilla and Jira, GitHub offers a larger number of labels to its users. In addition, the tendency in early issue tracking systems was to label issue reports as bug generally. Hence, our findings suggest that there has been an improvement in the utilization of issue labels in OSS development projects hosted on GitHub compared to the ones hosted on traditional issue tracking systems.

Some studies [18, 45] leveraged traditional machine learning algorithms—Naive Bayes, linear discriminant analysis, k-nearest neighbors, SVMs, decision trees, and random forests—to classify the issue reports as bug and non-bug. They specifically experimented on a manually classified dataset provided in previous work [17], and achieved the highest accuracy values ranged between 75% and 94% among different projects.

Recently, BERT-based models have been started to be used for the issue label classification problem. However, these studies have assumed the assigned issue labels as the ground truth, without performing any action to verify or refine these labels for developing automated issue label classification approaches, such as manual classification.

Wang et al. [49] compared the performance of a pre-trained contextual language representation obtained with BERT to the performance achieved by traditional deep learning models. Experimenting on the issues collected from GitHub, they found that BERT outperforms traditional deep learning models when a large amount of training data is used. Conversely, CNNs perform better than BERT in the presence of small-size training data, consisting of less than 5,000 issues. Instead of a multi-class issue label classification setting, they formulated their approach to perform multi-label issue classification, that is recommending $k$ possible labels for any issue report. As such, the performance of their approach is measured using F1-score@$k$, which limits direct comparison with the performance obtained in our study where a multi-class classification problem is addressed.

A BERT variant, RoBERTa, was leveraged by Izadi et al. [50] for predicting the label of a newly created issue report. They fine-tuned the RoBERTa model on a dataset of 818K issues labeled as bug, enhancement, or support/documentation, collected from GitHub. Similar to what we applied in our study, they relied on the textual information contained in each issue title and issue body, and achieved an overall accuracy of 82%, with F1-score values of 85%, 84%, and 67% in bug, enhancement, and support/documentation classes, respectively.

Kallis et al. [15] introduced a dataset, NLBSE'22 dataset, of 803K issue reports collected from GitHub which were categorized into one of three labels: bug, enhancement, or question. On this dataset, they obtained an accuracy of 81.8% with the fastText approach they previously developed [13] and extended later [14]. Several studies [51, 52, 53, 54, 55] experimented with this dataset, where most of the proposed models developed based on BERT [39], including its different variants, namely RoBERTa [19], CodeBERT [40], ALBERT [41], and seBERT [42].

All of the proposed models outperformed the initial study [15] on overall accuracy. While the classical models (MLP and logistic regression) achieved higher performance, the improvements were small (0.011 and 0.004). The remaining classifiers, based on BERT, achieved a comparable classification performance, having accuracy values ranging between 0.855 and 0.872. The model proposed by Izadi [51] achieved the highest overall accuracy, potentially due to the additional data preprocessing steps applied, specifically the removal of duplicate issue reports in the training set.

Based on the error analysis of the proposed model [53], Colavito et al. [57] experimented with different data quality criteria applied to the NLBSE'22 dataset to evaluate the impact of such data quality filters on model performance. However, they stated that none of their attempts had a significant effect on model performance. In addition to the NLBSE'22 dataset, they also employed a Jira dataset, introduced by previous work [58], which already satisfied the determined data quality criteria, involving 2.2M issue reports having a label of bug, enhancement, or question. After, the same preprocessing steps

[53] were applied to the Jira dataset, the fine-tuned RoBERTa model achieved a micro-average F1-score of 91.36% on this dataset. However, the smaller number of instances in the question class had a significant impact on the model performance, resulting in lower precision, recall, and F1-score values for all classes, except for the bug.

In this thesis study, we first reproduced the previous work [51] adopting a BERT-based approach, RoBERTa [19], by applying similar preprocessing steps and model parameters on the same dataset, NLBSE'22 dataset, and as expected, obtained similar performance results. Since the NLBSE'22 dataset was collected using only a time-based criterion for inclusion, without any concerns on the popularity or activity of the projects, we constructed an additional dataset as part of this thesis study using the REST API provided by GitHub, GitHub'23 dataset, with the aim of obtaining a more representative dataset reflecting the nature of large-scale OSS development projects. We fine-tuned the RoBERTa model with GitHub'23 dataset by applying the same preprocessing steps and model parameters as in [51]. Although we observed improvements in all three classification metrics for the (worst-performing) question class (13% in precision, 5% in recall, and 9% in F1-score), while other metrics being approximately the same, overall accuracy was still 88%.

This result seems to be aligned with the experiments conducted in [57] which determined a set of quality criteria, involving the number of stars, repository age, and issues having more than single label, to filter out uncertain data in the NLBSE'22 dataset. Actually, we utilized these filters (the number of stars and repository age) and several others to construct a more representative dataset of large-scale repositories. In line with [57], we also did not observe any significance performance improvement with RoBERTa in this new dataset. However, what we revealed is that eliminating the questionable issue reports from the test set actually improved the classification performance of the RoBERTa model by **8%**, achieving an accuracy of **.9458** on the NLBSE'22 dataset. Additionally, all class-based performance metrics—precision, recall, and F1-score—were improved across all three classes.

# CHAPTER 7

# CONCLUSION AND FUTURE WORK

In this section, the contributions of this thesis study are summarized. Next, the implications of the study for theory and practice are discussed. Then, threats to validity are listed and finally, potential directions for future research are suggested.

## 7.1 Contributions

Although the accurate assignment of issue labels is crucial for effective issue management, only a few studies have investigated the problem of mislabeling. Existing research has primarily focused on issue reports from more traditional issue tracking systems, such as Bugzilla, with no studies specifically addressing the reliability of issue labels on GitHub. The main objective and motivation of this thesis study is to investigate the reliability of issue labels in OSS development projects, aiming to improve issue management processes.

The research methodology employed in the study involved several steps. First, two datasets were collected which involved issue reports created in the OSS development projects hosted on GitHub. Second, we experimented with state-of-the-art open source LLMs, specifically with a BERT-based model (RoBERTa), and the Llama 3 model, with its 8B and 70B variants, for issue label classification. The RoBERTa model was fine-tuned with training datasets, whereas for Llama 3, only inference was performed. The performance of the models were compared. Next, we manually classified the issue reports for which at least one of the models classified the issue with a label different from the assigned one. We performed this classification relying only on the issue content, involving the issue title and the issue body. The patterns where a mismatch between the assigned issue labels and the actual content of the issue reports were identified. We commented on the manual classification process that we performed, highlighting concerns about the reliability of the issue labels in OSS development projects, as well as utilizing LLMs in issue label classification.

The primary contributions of this thesis are listed below:

- In a representative random sample taken from OSS development projects hosted on GitHub, it is found that 11% of the issue reports have actually questionable issue labels.
- Experiments with state-of-the-art open source LLMs were conducted for issue label classification and their classification tendencies were investigated.
- The patterns observed in questionable issue reports are identified.

## 7.2 Implications for Theory

This section discusses the potential theoretical implications of the research findings. They can be listed as follows:

**1. Contribution to the reliability of issue labels in OSS development projects:**

The study investigates the reliability of issue labels in issue tracking systems, particularly in OSS development projects. The findings of the study contribute to the identification and understanding of patterns inherent the in issue reports that have a mismatch between their labels and contents. The discovery that 11% of issues have indeed questionable labels highlights a gap in the accuracy and consistency of labeling practices, which can inform future studies on the reliability and effectiveness of issue labeling mechanisms.

**2. Impact on machine learning performance:**

The improvement in RoBERTa's classification performance, particularly in overall accuracy (from 87.64% to 94.58%) after removing questionable issues suggests that the quality of data plays a crucial role in the performance of LLMs. This finding underscores the importance of data quality and could lead to new theoretical discussions on the impact of data quality on model performance in NLP tasks.

**3. Refinement of theoretical models for issue classification:**

The research may prompt a reevaluation of existing theoretical models that assume issue labels as the ground truth. The findings of the study suggest that such assumptions may lead to suboptimal results, thus encouraging the development of more robust models that account for or correct labeling inaccuracies.

## 7.3 Implications for Practice

This section discusses the potential practical implications of the research findings. They can be listed as follows:

**1. Improving issue labeling practices in OSS development projects:**

The results of the manual classification indicate a need for more rigorous issue label assignment practices in OSS development projects. Encouraging OSS community members to adopt an agreed-upon policy for issue label assignment could help ensure that issue labels accurately reflect the content, thereby enhancing the efficiency of issue tracking and resolution.

**2. Enhancing the accuracy of machine learning models in real-world applications:**

The findings of the study suggest that preprocessing steps to identify and eliminate mislabeled data could significantly improve the performance of machine learning models, like RoBERTa. This observation has practical implications for any application of machine learning in OSS development projects, recommending that practitioners implement data cleansing steps to maximize model accuracy.

**3. Automated detection and correction of mislabeled issues:**

Based on our findings, there is a potential to develop or integrate automated recommender systems that detect and suggest corrections for questionable issues during issue label assignment. Such tools could be particularly valuable in large-scale OSS development projects where manual review of all issue labels is impractical, thereby improving issue management processes.

**4. Guidance for data annotation in NLP:**

For practitioners involved in training NLP models, the research highlights the critical importance of accurate data annotation. This could lead to better training protocols, including more thorough reviews of labeled data and the adoption of tools or techniques that help identify and correct mislabeled instances before these are used in model training.

## 7.4 Threats to Validity

This section describes the threats to validity of the study. They can be listed as follows:

**1. External validity:**

Threats to external validity relate to the generalizability of the study. To mitigate these kinds of threats, we included two different datasets in our research containing 803K and 105K issue reports, respectively, collected from OSS projects hosted on GitHub. Moreover, we manually classified 1,541 issue reports for which at least one of the models performed a classification different from the assigned issue label. However, this sample may not be adequately representative of all GitHub projects.

**2. Internal validity:**

Internal validity refers to how well a study is designed and executed so that the conclusions drawn from the study can be confidently attributed to the effect of the variables being tested, rather than other factors. In this study, we experimented with LLMs, utilizing GPUs, which perform parallel floating-point calculations to optimize speed. These calculations can result in tiny differences in the last few significant digits, leading to slightly different results. This variability can be seen as a threat to internal validity, as it introduces randomness that makes it difficult to attribute the model's output to the specific input conditions given. Instead, some of the variability in responses might result from the model's inherent stochastic nature, which acts like a confounding variable, making it challenging to determine if the results are due to the inputs or to this uncontrolled variability.

**3. Construct validity:**

This study aimed to assess the reliability of issue labels used in OSS development projects hosted on GitHub, but the reliance on only the issue title and the issue body for manual classification introduces a potential threat to construct validity. Specifically, this approach might not fully capture the context of the issue reports, without knowing the current functionalities supported by the projects and their characteristics. Without considering these additional elements, the manual classification performed might not fully reflect the actual content of the issues, potentially limiting the accuracy of the findings. This limitation suggests that the study may not entirely measure the true reliability of the issue labels, as important contextual factors could be overlooked.

## 7.5 Future Research

This thesis study presents several opportunities for future research, which can be listed as follows:

**1. Identifying plausible issue reports with Llama 3:**

Inspired by the results of hypothesis testing, which indicate that the confidence levels in classifications of issues considered plausible are greater than those considered questionable, the 8B or 70B variants of the Llama 3 model could be utilized to evaluate confidence levels and identify plausible issue reports, thus increasing data quality in training sets. The performances of the models with new training sets could then be compared with the original results.

**2. Determining related features to identify questionable issue reports:**

Related features of the issue reports could be investigated to develop automated approaches for identifying questionable issue reports, such as the number of tokens in the issue reports or the issue author association. For instance, experiments could be conducted on the number of tokens in the issue reports. The issue reports with a low number of tokens might either be ignored or defined as a feature, as they may not be sufficiently informative.

**3. Investigating questionable issue reports:**

The questionable issue reports identified in this study could be further investigated to understand why they were misclassified. The patterns in the contents of these issue reports and the potential reasons for misclassifications could be identified.

**4. Performing manual classification on the GitHub'23 dataset:**

We manually classified issue reports from a random sample of the NLBSE'22 test set to investigate the extent to which the assigned issue labels are relevant to the content of the issue reports. This analysis could be performed on the GitHub'23 dataset as well. As these datasets have different characteristics, the differences and similarities between the results could provide a deeper understanding on the reliability of issue labels.

**5. Applying data quality filters on the training sets:**

Some filters could be applied to the datasets to improve data quality. For example, users new to OSS development may lack experience in creating issue reports, leading to lower-quality submissions. Therefore, only issue reports created by users who have submitted a minimum number of issues could be included in the dataset to avoid introducing bias.

**6. Expanding the size of datasets:**

We utilized the Llama 3 models on small datasets as these models require high computational power and processing time, particularly for the Llama 3 70B variant. In future work, experiments could be performed on larger datasets.

**7. Expanding the issue labels:**

This thesis study focused on issue reports associated with the three most commonly used issue labels in OSS development projects: bug, enhancement, and question. Future experiments could expand this label set to compare results and incorporate a broader range of issue labels for reliability analysis and issue label classification.

**8. Conducting qualitative studies:**

Future research could be enriched by qualitative methods, such as surveys and in-depth interviews with OSS practitioners. This approach would provide valuable insights into their perspectives and experiences regarding the reliability of issue labels used in OSS development projects.

**9. Developing recommender systems:**

Automated tools could be developed to guide users and developers in determining the appropriate label for an issue report. This would help foster a common consensus among OSS community members.

# REFERENCES


[1] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open-source software development: What we know and what we do not know," *ACM Comput. Surv.*, vol. 44, no. 2, 2008.

[2] G. K. Lee and R. E. Cole, "From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development," *Organization Science*, vol. 14, no. 6, pp. 633–649, 2003.

[3] M. Gerosa, I. Wiese, B. Trinkenreich, G. Link, G. Robles, C. Treude, I. Steinmacher, and A. Sarma, "The shifting sands of motivation: Revisiting what drives contributors in open source," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1046–1058, 2021.

[4] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillère, J. Klein, and Y. L. Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from github," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 188–197, 2013.

[5] S. Panichella, G. Canfora, and A. Di Sorbo, ""won't we fix this issue?" qualitative characterization and automated identification of wontfix issues on github," *Information and Software Technology*, vol. 139, p. 106665, 2021.

[6] Q. Fan, Y. Yu, G. Yin, T. Wang, and H. Wang, "Where is the road for issue reports classification based on text mining?," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 121–130, 2017.

[7] P. Hooimeijer and W. Weimer, "Modeling bug report quality," *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, pp. 34–43, 2007.

[8] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?," *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 308–318, 2008.

[9] J. Cabot, J. L. C. Izquierdo, V. Cosentino, and B. Rolandi, "Exploring the use of labels to categorize issues in open-source software projects," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, 2015.

[10] J. L. C. Izquierdo, V. Cosentino, B. Rolandi, A. Bergel, and J. Cabot, "Gila: Github label analyzer," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, 2015.

[11] Z. Liao, D. He, Z. Chen, X. Fan, Y. Zhang, and S. Liu, "Exploring the characteristics of issue-related behaviors in github using visualization techniques," *IEEE Access*, vol. 6, 2018.

[12] A. Zeller, *Perspectives on the Future of Software Engineering*, ch. Can we trust software repositories?, pp. 209–215. Springer, 2013.

[13] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 406–409, 2019.

[14] R. Kallis, A. D. Sorbo, G. Canfora, and S. Panichella, "Predicting issue types on github," *Science of Computer Programming*, vol. 205, 2021.

[15] R. Kallis, O. Chaparro, A. Di Sorbo, and S. Panichella, "Nlbse'22 tool competition," in *2022 IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pp. 25–28, 2022.

[16] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement? a text-based approach to classify change requests," in *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, CASCON '08, pp. 304–318, 2008.

[17] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: How misclassification impacts bug prediction," in *2013 35th International Conference on Software Engineering (ICSE)*, pp. 392–401, 2013.

[18] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," *Journal of Software: Evolution and Process*, vol. 28, 2016.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, 2019.

[20] AI@Meta, "Llama 3," 2024. [Online]. Available: `https://github.com/meta-llama/llama3` [Accessed: Aug. 19, 2024].

[21] J. W. Castro Llanos and S. T. Acuña Castillo, "Differences between traditional and open source development activities," in *Product-Focused Software Process Improvement*, pp. 131–144, 2012.

[22] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," 2018.

[23] J. F. Chen, Q. H. Do, and H. N. Hsieh, "Training artificial neural networks by a hybrid pso-cs algorithm," *Algorithms*, vol. 8, pp. 292–308, 2015.

[24] A. A. Heidari, H. Faris, S. Mirjalili, I. Aljarah, and M. Mafarja, *Ant lion optimizer: Theory, literature review, and application in multi-layer perceptron neural networks*, vol. 811, pp. 23–46. 2020.

[25] Y. S. Park and S. Lek, *Artificial Neural Networks: Multilayer Perceptron for Ecological Modeling*, vol. 28, ch. 7, pp. 123–140. Elsevier, 2016.

[26] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, vol. 1, 2014.

[27] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[28] D. P. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*. USA: John Wiley & Sons, Inc., 2001.

[29] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, 2015.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, 1997.

[31] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[32] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 4, 2014.

[34] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[35] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, 2015.

[36] K. S. Kalyan, "A survey of gpt-3 family large language models including chatgpt and gpt-4," *Natural Language Processing Journal*, vol. 6, 2024.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.

[38] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[39] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[40] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "Codebert: A pre-trained model for programming and natural languages," in *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, 2020.

[41] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[42] J. von der Mosel, A. Trautsch, and S. Herbold, "On the validity of pre-trained transformers for natural language processing in the software engineering domain," 2021. arXiv preprint arXiv:2109.04738.

[43] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[44] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, 2021.

[45] N. Pandey, D. K. Sanyal, A. Hudait, and A. Sen, "Automated classification of software issue reports using machine learning techniques: an empirical study," *Innovations in Systems and Software Engineering*, vol. 13, 2017.

[46] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 3, 2014.

[47] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, vol. 2, pp. 3111–3119, 2013.

[48] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 2, 2017.

[49] J. Wang, X. Zhang, and L. Chen, "How well do pre-trained contextual language representations recommend labels for github issues?," *Knowledge-Based Systems*, vol. 232, 2021.

[50] M. Izadi, K. Akbari, and A. Heydarnoori, "Predicting the objective and priority of issue reports in software repositories," *Empirical Software Engineering*, vol. 27, 2022.

[51] M. Izadi, "Catiss: An intelligent tool for categorizing issues reports using transformers," in *Proceedings - 1st International Workshop on Natural Language-Based Software Engineering, NLBSE 2022*, 2022.

[52] S. Bharadwaj and T. Kadam, "Github issue classification using bert-style models," in *Proceedings - 1st International Workshop on Natural Language-Based Software Engineering, NLBSE 2022*, 2022.

[53] G. Colavito, F. Lanubile, and N. Novielli, "Issue report classification using pre-trained language models," in *Proceedings - 1st International Workshop on Natural Language-Based Software Engineering, NLBSE 2022*, 2022.

[54] M. L. Siddiq and J. C. Santos, "Bert-based github issue report classification," in *Proceedings - 1st International Workshop on Natural Language-Based Software Engineering, NLBSE 2022*, 2022.

[55] A. Trautsch and S. Herbold, "Predicting issue types with sebert," in *Proceedings - 1st International Workshop on Natural Language-Based Software Engineering, NLBSE 2022*, 2022.

[56] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[57] G. Colavito, F. Lanubile, N. Novielli, and L. Quaranta, "Impact of data quality for automatic issue classification using pre-trained language models," *Journal of Systems and Software*, vol. 210, 2024.

[58] L. Montgomery, C. Luders, and W. Maalej, "An alternative issue tracking dataset of public jira repositories," in *Proceedings - 2022 Mining Software Repositories Conference, MSR 2022*, 2022.

[59] E. Kalliamvakou, L. Singer, G. Gousios, D. M. German, K. Blincoe, and D. Damian, "The promises and perils of mining github," in *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 2014.

[60] H. Borges and M. Tulio Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[61] J. Jiang, D. Lo, X. Ma, F. Feng, and L. Zhang, "Understanding inactive yet available assignees in github," *Information and Software Technology*, vol. 91, no. C, pp. 44–55, 2017.

[62] S. Nielebock, R. Heumüller, and F. Ortmeier, "Programmers do not favor lambda expressions for concurrent object-oriented code," *Empirical Software Engineering*, vol. 24, pp. 103–138, 2019.

[63] D. Castro and M. Schots, "Analysis of test log information through interactive visualizations," *IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pp. 156–166, 2018.

[64] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," *31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 426–437, 2016.

[65] Student, "The probable error of a mean," *Biometrika*, vol. 6, 1908.

[66] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, 1965.

[67] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, 1947.

# APPENDIX A

# LISTS OF REPOSITORIES

This chapter presents the repositories that are in the initial list and the selected repositories that are included in the GitHub'23 dataset.

## A.1 Initial List of Repositories

The initial list of the repositories returned from the query explained in Section 4.1.2 can be seen in Table 38.

Table 38: The repositories that are in the initial list.

| Number | Repository | Number | Repository |
|---|---|---|---|
| 1 | Snailclimb/JavaGuide | 87 | JeffLi1993/springboot-learning-example |
| 2 | facebook/react-native | 88 | medcl/elasticsearch-analysis-ik |
| 3 | iluwatar/java-design-patterns | 89 | williamfiset/Algorithms |
| 4 | macrozheng/mall | 90 | material-components/material-components-android |
| 5 | spring-projects/spring-boot | 91 | android10/Android-CleanArchitecture |
| 6 | elastic/elasticsearch | 92 | dyc87112/SpringBoot-Learning |
| 7 | kdn251/interviews | 93 | antlr/antlr4 |
| 8 | TheAlgorithms/Java | 94 | prestodb/presto |
| 9 | spring-projects/spring-framework | 95 | baomidou/mybatis-plus |
| 10 | google/guava | 96 | android-hacker/VirtualXposed |
| 11 | ReactiveX/RxJava | 97 | hdodenhof/CircleImageView |
| 12 | square/retrofit | 98 | thingsboard/thingsboard |
| 13 | apache/dubbo | 99 | ben-manes/caffeine |
| 14 | PhilJay/MPAndroidChart | 100 | alibaba/ARouter |
| 15 | skylot/jadx | 101 | mockito/mockito |
| 16 | eugenp/tutorials | 102 | alibaba/DataX |
| 17 | airbnb/lottie-android | 103 | Tencent/QMUI_Android |
| 18 | dbeaver/dbeaver | 104 | Konloch/bytecode-viewer |

Table 38 (cont.)

| Number | Repository | Number | Repository |
|---|---|---|---|
| 19 | bumptech/glide | 105 | apache/hadoop |
| 20 | alibaba/arthas | 106 | eclipse-vertx/vert.x |
| 21 | Blankj/AndroidUtilCode | 107 | arduino/Arduino |
| 22 | netty/netty | 108 | orhanobut/logger |
| 23 | zxing/zxing | 109 | elastic/logstash |
| 24 | xkcoding/spring-boot-demo | 110 | lgvalle/Material-Animations |
| 25 | ityouknow/spring-boot-examples | 111 | Curzibn/Luban |
| 26 | alibaba/easyexcel | 112 | Bigkoo/Android-PickerView |
| 27 | halo-dev/halo | 113 | google/tink |
| 28 | apolloconfig/apollo | 114 | apache/pulsar |
| 29 | SeleniumHQ/selenium | 115 | deeplearning4j/deeplearning4j |
| 30 | alibaba/nacos | 116 | JessYanCoding/AndroidAutoSize |
| 31 | alibaba/druid | 117 | seaswalker/spring-analysis |
| 32 | dromara/hutool | 118 | GoogleContainerTools/jib |
| 33 | crossoverJie/JCSprout | 119 | pinpoint-apm/pinpoint |
| 34 | alibaba/canal | 120 | java-decompiler/jd-gui |
| 35 | alibaba/spring-cloud-alibaba | 121 | Netflix/zuul |
| 36 | lenve/vhr | 122 | apache/druid |
| 37 | wuyouzhuguli/SpringAll | 123 | questdb/questdb |
| 38 | apache/kafka | 124 | LuckSiege/PictureSelector |
| 39 | JakeWharton/butterknife | 125 | sqshq/piggymetrics |
| 40 | TeamNewPipe/NewPipe | 126 | greenrobot/greenDAO |
| 41 | alibaba/fastjson | 127 | youth5201314/banner |
| 42 | xuxueli/xxl-job | 128 | zhihu/Matisse |
| 43 | termux/termux-app | 129 | daimajia/AndroidViewAnimations |
| 44 | proxyee-down-org/proxyee-down | 130 | quarkusio/quarkus |
| 45 | greenrobot/EventBus | 131 | daimajia/AndroidSwipeLayout |
| 46 | scwang90/SmartRefreshLayout | 132 | projectlombok/lombok |
| 47 | signalapp/Signal-Android | 133 | google/guice |
| 48 | Netflix/Hystrix | 134 | Netflix/eureka |
| 49 | doocs/leetcode | 135 | codecentric/spring-boot-admin |
| 50 | DrKLO/Telegram | 136 | pagehelper/Mybatis-PageHelper |
| 51 | google/gson | 137 | neo4j/neo4j |
| 52 | apache/skywalking | 138 | Tencent/VasSonic |
| 53 | libgdx/libgdx | 139 | Yalantis/uCrop |
| 54 | apache/flink | 140 | DuGuQiuBai/Java |
| 55 | redisson/redisson | 141 | apache/zookeeper |
| 56 | jenkinsci/jenkins | 142 | janishar/mit-deep-learning-book-pdf |
| 57 | alibaba/Sentinel | 143 | realm/realm-java |
| 58 | bazelbuild/bazel | 144 | redis/jedis |
| 59 | google/ExoPlayer | 145 | zaproxy/zaproxy |
| 60 | apache/rocketmq | 146 | frank-lam/fullstack-tutorial |
| 61 | ReactiveX/RxAndroid | 147 | pxb1988/dex2jar |

Table 38 (cont.)

| Number | Repository | Number | Repository |
|---|---|---|---|
| 62 | didi/DoKit | 148 | permissions-dispatcher/PermissionsDispatcher |
| 63 | EnterpriseQualityCoding/FizzBuzzEnterpriseEdition | 149 | jfeinstein10/SlidingMenu |
| 64 | Anuken/Mindustry | 150 | code4craft/webmagic |
| 65 | oracle/graal | 151 | gyf-dev/ImmersionBar |
| 66 | CarGuo/GSYVideoPlayer | 152 | careercup/CtCI-6th-Edition |
| 67 | mybatis/mybatis-3 | 153 | H07000223/FlycoTabLayout |
| 68 | apache/shardingsphere | 154 | lingochamp/FileDownloader |
| 69 | brettwooldridge/HikariCP | 155 | mission-peace/interview |
| 70 | Baseflow/PhotoView | 156 | grpc/grpc-java |
| 71 | linlinjava/litemall | 157 | Netflix/conductor |
| 72 | dianping/cat | 158 | android-async-http/android-async-http |
| 73 | OpenAPITools/openapi-generator | 159 | daniulive/SmarterStreaming |
| 74 | keycloak/keycloak | 160 | jeasonlzy/okhttp-OkGo |
| 75 | yudaocode/SpringBoot-Labs | 161 | aosp-mirror/platform_frameworks_base |
| 76 | forezp/SpringCloudLearning | 162 | alibaba/COLA |
| 77 | iBotPeaches/Apktool | 163 | tbruyelle/RxPermissions |
| 78 | openjdk/jdk | 164 | square/javapoet |
| 79 | facebook/fresco | 165 | amitshekhariitbhu/android-interview-questions |
| 80 | Tencent/tinker | 166 | apereo/cas |
| 81 | nostra13/Android-Universal-Image-Loader | 167 | jhy/jsoup |
| 82 | shuzheng/zheng | 168 | google/auto |
| 83 | winterbe/java8-tutorial | 169 | JessYanCoding/MVPArms |
| 84 | openzipkin/zipkin | 170 | languagetool-org/languagetool |
| 85 | LMAX-Exchange/disruptor | 171 | clojure/clojure |
| 86 | Tencent/APIJSON | | |

### A.2 List of Repositories Included in the GitHub'23 Dataset

The list of the repositories that are included in the GitHub'23 dataset is provided in Table 39.

Table 39: The repositories included in the GitHub'23 dataset.

| Number | Repository | Number | Repository |
|---|---|---|---|
| 1 | Anuken/Mindustry | 16 | neo4j/neo4j |
| 2 | apache/druid | 17 | Netflix/conductor |
| 3 | apache/dubbo | 18 | OpenAPITools/openapi-generator |

Table 39 (cont.)

| Number | Repository | Number | Repository |
|---|---|---|---|
| 4 | apache/pulsar | 19 | openzipkin/zipkin |
| 5 | apache/shardingsphere | 20 | pinpoint-apm/pinpoint |
| 6 | apache/skywalking | 21 | quarkusio/quarkus |
| 7 | bazelbuild/bazel | 22 | questdb/questdb |
| 8 | codecentric/spring-boot-admin | 23 | ReactiveX/RxJava |
| 9 | dbeaver/dbeaver | 24 | redis/jedis |
| 10 | elastic/elasticsearch | 25 | SeleniumHQ/selenium |
| 11 | elastic/logstash | 26 | spring-projects/spring-boot |
| 12 | facebook/react-native | 27 | spring-projects/spring-framework |
| 13 | GoogleContainerTools/jib | 28 | TeamNewPipe/NewPipe |
| 14 | grpc/grpc-java | 29 | thingsboard/thingsboard |
| 15 | mockito/mockito | 30 | zaproxy/zaproxy |

# APPENDIX B

# LABEL MAPPING RULE SETS

This chapter presents the details of the constructed rule sets in each round of the label mapping.

## B.1 Rule Set for Round 1

The rule set for Round 1 is presented in Table 40.

Table 40: Rule Set for Round 1.

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 1 | facebook/react-native | Bug | bug |
| 2 | facebook/react-native | Impact: Bug | bug |
| 3 | facebook/react-native | Type: Enhancement | enhancement |
| 4 | spring-projects/spring-boot | type: bug | bug |
| 5 | spring-projects/spring-boot | type: enhancement | enhancement |
| 6 | elastic/elasticsearch | >bug | bug |
| 7 | elastic/elasticsearch | >enhancement | enhancement |
| 8 | elastic/elasticsearch | >feature | enhancement |
| 9 | spring-projects/spring-framework | type: bug | bug |
| 10 | spring-projects/spring-framework | type: enhancement | enhancement |
| 11 | ReactiveX/RxJava | Bug | bug |
| 12 | ReactiveX/RxJava | Enhancement | enhancement |
| 13 | ReactiveX/RxJava | Feature-Request | enhancement |
| 14 | apache/dubbo | type/bug | bug |
| 15 | apache/dubbo | type/enhancement | enhancement |
| 16 | apache/dubbo | type/proposal | enhancement |
| 17 | dbeaver/dbeaver | bug | bug |
| 18 | dbeaver/dbeaver | enhancement | enhancement |
| 19 | dbeaver/dbeaver | feature request | enhancement |
| 20 | SeleniumHQ/selenium | bug | bug |
| 21 | SeleniumHQ/selenium | I-defect | bug |
| 22 | SeleniumHQ/selenium | I-enhancement | enhancement |
| 23 | TeamNewPipe/NewPipe | bug | bug |

Table 40 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 24 | TeamNewPipe/NewPipe | feature request | enhancement |
| 25 | apache/skywalking | bug | bug |
| 26 | apache/skywalking | enhancement | enhancement |
| 27 | apache/skywalking | feature | enhancement |
| 28 | bazelbuild/bazel | query bugs | bug |
| 29 | bazelbuild/bazel | type: bug | bug |
| 30 | bazelbuild/bazel | type: feature request | enhancement |
| 31 | Anuken/Mindustry | bug | bug |
| 32 | apache/shardingsphere | type: bug | bug |
| 33 | apache/shardingsphere | type: enhancement | enhancement |
| 34 | apache/shardingsphere | type: new feature | enhancement |
| 35 | OpenAPITools/openapi-generator | Enhancement: CI/Test | enhancement |
| 36 | OpenAPITools/openapi-generator | Enhancement: Code Cleanup | enhancement |
| 37 | OpenAPITools/openapi-generator | Enhancement: Code format | enhancement |
| 38 | OpenAPITools/openapi-generator | Enhancement: Compatibility | enhancement |
| 39 | OpenAPITools/openapi-generator | Enhancement: Feature | enhancement |
| 40 | OpenAPITools/openapi-generator | Enhancement: General | enhancement |
| 41 | OpenAPITools/openapi-generator | Enhancement: New generator | enhancement |
| 42 | OpenAPITools/openapi-generator | Enhancement: Performance | enhancement |
| 43 | OpenAPITools/openapi-generator | Enhancement: Security | enhancement |
| 44 | OpenAPITools/openapi-generator | Feature: Authentication | enhancement |
| 45 | OpenAPITools/openapi-generator | Feature: Composition / Inheritance | enhancement |
| 46 | OpenAPITools/openapi-generator | Feature: Documentation | enhancement |
| 47 | OpenAPITools/openapi-generator | Feature: Enum | enhancement |
| 48 | OpenAPITools/openapi-generator | Feature: Generator | enhancement |
| 49 | OpenAPITools/openapi-generator | Feature: OAS 3.0 spec support | enhancement |
| 50 | OpenAPITools/openapi-generator | Feature: OAS 3.1.0 spec support | enhancement |
| 51 | OpenAPITools/openapi-generator | General: Suggestion | enhancement |
| 52 | OpenAPITools/openapi-generator | Issue: Bug | bug |
| 53 | openzipkin/zipkin | bug | bug |
| 54 | openzipkin/zipkin | enhancement | enhancement |
| 55 | thingsboard/thingsboard | bug | bug |
| 56 | thingsboard/thingsboard | enhancement | enhancement |
| 57 | thingsboard/thingsboard | feature | enhancement |
| 58 | mockito/mockito | bug | bug |
| 59 | mockito/mockito | enhancement | enhancement |
| 60 | mockito/mockito | new feature | enhancement |
| 61 | elastic/logstash | >bug | bug |
| 62 | elastic/logstash | bug | bug |
| 63 | elastic/logstash | enhancement | enhancement |
| 64 | apache/pulsar | type/bug | bug |
| 65 | apache/pulsar | type/enhancement | enhancement |
| 66 | apache/pulsar | type/feature | enhancement |

Table 40 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 67 | GoogleContainerTools/jib | enhancement | enhancement |
| 68 | GoogleContainerTools/jib | proposal | enhancement |
| 69 | GoogleContainerTools/jib | type: bug | bug |
| 70 | GoogleContainerTools/jib | type: feature request | enhancement |
| 71 | pinpoint-apm/pinpoint | bug | bug |
| 72 | pinpoint-apm/pinpoint | enhancement | enhancement |
| 73 | pinpoint-apm/pinpoint | proposal | enhancement |
| 74 | apache/druid | Bug | bug |
| 75 | apache/druid | Feature/Change Description | enhancement |
| 76 | apache/druid | Feature | enhancement |
| 77 | apache/druid | Improvement | enhancement |
| 78 | apache/druid | Proposal | enhancement |
| 79 | questdb/questdb | Bug | bug |
| 80 | questdb/questdb | Bug 0 | bug |
| 81 | questdb/questdb | Enhancement | enhancement |
| 82 | questdb/questdb | New feature | enhancement |
| 83 | questdb/questdb | Performance | enhancement |
| 84 | quarkusio/quarkus | kind/bug | bug |
| 85 | quarkusio/quarkus | kind/enhancement | enhancement |
| 86 | quarkusio/quarkus | kind/extension-proposal | enhancement |
| 87 | quarkusio/quarkus | kind/new-feature | enhancement |
| 88 | codecentric/spring-boot-admin | bug | bug |
| 89 | codecentric/spring-boot-admin | enhancement | enhancement |
| 90 | neo4j/neo4j | bug | bug |
| 91 | neo4j/neo4j | enhancement | enhancement |
| 92 | neo4j/neo4j | feature | enhancement |
| 93 | redis/jedis | bug | bug |
| 94 | redis/jedis | enhancement | enhancement |
| 95 | redis/jedis | feature | enhancement |
| 96 | zaproxy/zaproxy | bug | bug |
| 97 | zaproxy/zaproxy | enhancement | enhancement |
| 98 | zaproxy/zaproxy | Type-Defect | bug |
| 99 | grpc/grpc-java | bug | bug |
| 100 | grpc/grpc-java | End-to-end bugs | bug |
| 101 | grpc/grpc-java | enhancement | enhancement |
| 102 | grpc/grpc-java | Type: Bug | bug |
| 103 | grpc/grpc-java | Type: Feature | enhancement |
| 104 | Netflix/conductor | enhancement | enhancement |
| 105 | Netflix/conductor | type: bug | bug |
| 106 | Netflix/conductor | type: feature | enhancement |

### B.2 Rule Set for Round 2

The rule set for Round 2 is presented in Table 41.

Table 41: Rule Set for Round 2.

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 1 | facebook/react-native | Resolution: Locked | other |
| 2 | facebook/react-native | Ran Commands | other |
| 3 | facebook/react-native | Type: Question | question |
| 4 | facebook/react-native | Resolution: For Stack Overflow | question |
| 5 | spring-projects/spring-boot | type: dependency-upgrade | other |
| 6 | spring-projects/spring-boot | type: task | other |
| 7 | spring-projects/spring-boot | status: invalid | other |
| 8 | spring-projects/spring-boot | status: forward-port | other |
| 9 | spring-projects/spring-boot | type: documentation | other |
| 10 | spring-projects/spring-boot | status: duplicate | other |
| 11 | spring-projects/spring-boot | for: stackoverflow | question |
| 12 | spring-projects/spring-boot | status: declined | other |
| 13 | spring-projects/spring-boot | for: external-project | other |
| 14 | spring-projects/spring-boot | status: superseded | other |
| 15 | elastic/elasticsearch | >test-failure | other |
| 16 | elastic/elasticsearch | >docs | other |
| 17 | elastic/elasticsearch | discuss | question |
| 18 | elastic/elasticsearch | >test | other |
| 19 | spring-projects/spring-framework | status: invalid | other |
| 20 | spring-projects/spring-framework | status: declined | other |
| 21 | spring-projects/spring-framework | type: task | other |
| 22 | spring-projects/spring-framework | type: documentation | other |
| 23 | spring-projects/spring-framework | status: duplicate | other |
| 24 | spring-projects/spring-framework | type: backport | other |
| 25 | spring-projects/spring-framework | type: regression | bug |
| 26 | ReactiveX/RxJava | Question | question |
| 27 | ReactiveX/RxJava | Documentation | other |
| 28 | ReactiveX/RxJava | Information | other |
| 29 | ReactiveX/RxJava | Discussion | question |
| 30 | ReactiveX/RxJava | StackOverflow | question |
| 31 | apache/dubbo | type/question | question |
| 32 | apache/dubbo | type/feature | enhancement |
| 33 | apache/dubbo | type/discussion | question |
| 34 | apache/dubbo | type/refactor | other |
| 35 | dbeaver/dbeaver | question | question |
| 36 | dbeaver/dbeaver | duplicate | other |
| 37 | TeamNewPipe/NewPipe | duplicate | other |
| 38 | TeamNewPipe/NewPipe | question | question |

Table 41 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 39 | TeamNewPipe/NewPipe | discussion | question |
| 40 | TeamNewPipe/NewPipe | wontfix | other |
| 41 | apache/skywalking | question | question |
| 42 | apache/skywalking | invalid | other |
| 43 | apache/skywalking | wontfix | other |
| 44 | apache/skywalking | duplicate | other |
| 45 | apache/skywalking | documentation | other |
| 46 | apache/skywalking | discussion | question |
| 47 | bazelbuild/bazel | type: documentation (cleanup) | other |
| 48 | bazelbuild/bazel | type: process | other |
| 49 | apache/shardingsphere | status: invalid | other |
| 50 | apache/shardingsphere | type: question | question |
| 51 | apache/shardingsphere | in: test | other |
| 52 | apache/shardingsphere | type: duplicate | other |
| 53 | apache/shardingsphere | type: refactor | other |
| 54 | apache/shardingsphere | in: document | other |
| 55 | OpenAPITools/openapi-generator | General: Question | question |
| 56 | OpenAPITools/openapi-generator | Announcement | other |
| 57 | OpenAPITools/openapi-generator | General: Discussion | question |
| 58 | openzipkin/zipkin | question | question |
| 59 | openzipkin/zipkin | docs | other |
| 60 | openzipkin/zipkin | duplicate | other |
| 61 | thingsboard/thingsboard | Question | question |
| 62 | thingsboard/thingsboard | Bug | bug |
| 63 | thingsboard/thingsboard | Feature | enhancement |
| 64 | thingsboard/thingsboard | invalid | other |
| 65 | thingsboard/thingsboard | Docs enhancement | other |
| 66 | mockito/mockito | docs | other |
| 67 | mockito/mockito | question | question |
| 68 | mockito/mockito | wontfix | other |
| 69 | mockito/mockito | invalid | other |
| 70 | mockito/mockito | refactoring | other |
| 71 | elastic/logstash | docs | other |
| 72 | elastic/logstash | test failure | other |
| 73 | elastic/logstash | discuss | question |
| 74 | elastic/logstash | tests-infra | other |
| 75 | apache/pulsar | component/test | other |
| 76 | apache/pulsar | flaky-tests | other |
| 77 | apache/pulsar | doc-required | other |
| 78 | apache/pulsar | question | question |
| 79 | apache/pulsar | doc | other |
| 80 | apache/pulsar | doc-complete | other |
| 81 | GoogleContainerTools/jib | question | question |

Table 41 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 82 | GoogleContainerTools/jib | type:documentation | other |
| 83 | GoogleContainerTools/jib | release | other |
| 84 | pinpoint-apm/pinpoint | question | question |
| 85 | pinpoint-apm/pinpoint | dependencies | other |
| 86 | pinpoint-apm/pinpoint | release | other |
| 87 | apache/druid | Area - Testing | other |
| 88 | apache/druid | Area - Documentation | other |
| 89 | apache/druid | Discuss | question |
| 90 | apache/druid | Flaky test | other |
| 91 | apache/druid | Release Notes | other |
| 92 | questdb/questdb | Schrödinger's bug | other |
| 93 | questdb/questdb | Question | question |
| 94 | questdb/questdb | Documentation | other |
| 95 | quarkusio/quarkus | kind/question | question |
| 96 | quarkusio/quarkus | area/housekeeping | other |
| 97 | quarkusio/quarkus | triage/invalid | other |
| 98 | quarkusio/quarkus | triage/out-of-date | other |
| 99 | quarkusio/quarkus | area/documentation | other |
| 100 | quarkusio/quarkus | kind/epic | enhancement |
| 101 | codecentric/spring-boot-admin | invalid | other |
| 102 | codecentric/spring-boot-admin | duplicate | other |
| 103 | codecentric/spring-boot-admin | wontfix | other |
| 104 | codecentric/spring-boot-admin | documentation | other |
| 105 | codecentric/spring-boot-admin | discussion | question |
| 106 | codecentric/spring-boot-admin | for-external-project | other |
| 107 | neo4j/neo4j | question | question |
| 108 | neo4j/neo4j | docs | other |
| 109 | redis/jedis | question | question |
| 110 | redis/jedis | will not fix | other |
| 111 | redis/jedis | documentation | other |
| 112 | redis/jedis | maintenance | other |
| 113 | redis/jedis | testing | other |
| 114 | zaproxy/zaproxy | Type-Enhancement | enhancement |
| 115 | zaproxy/zaproxy | question | question |
| 116 | zaproxy/zaproxy | invalid | other |
| 117 | zaproxy/zaproxy | Type-Task | other |
| 118 | zaproxy/zaproxy | duplicate | other |
| 119 | grpc/grpc-java | question | question |
| 120 | grpc/grpc-java | experimental API | other |
| 121 | grpc/grpc-java | docs | other |
| 122 | Netflix/conductor | question | question |
| 123 | Netflix/conductor | type: docs | other |
| 124 | Netflix/conductor | invalid | other |

Table 41 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 125 | Netflix/conductor | duplicate | other |

### B.3 Rule Set for Round 3

The rule set for Round 3 is presented in Table 42.

Table 42: Rule Set for Round 3.

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 1 | spring-projects/spring-boot | type: regression | bug |
| 2 | spring-projects/spring-boot | type: wiki-documentation | other |
| 3 | spring-projects/spring-boot | type: epic | enhancement |
| 4 | elastic/elasticsearch | >non-issue | other |
| 5 | spring-projects/spring-framework | for: stackoverflow | question |
| 6 | spring-projects/spring-framework | type: dependency-upgrade | other |
| 7 | spring-projects/spring-framework | for: external-project | other |
| 8 | spring-projects/spring-framework | status: superseded | other |
| 9 | spring-projects/spring-framework | in: test | other |
| 10 | ReactiveX/RxJava | Test-Failures | other |
| 11 | ReactiveX/RxJava | Invalid | other |
| 12 | apache/dubbo | type/unit-test | other |
| 13 | dbeaver/dbeaver | won't do | other |
| 14 | dbeaver/dbeaver | other | other |
| 15 | apache/skywalking | test | other |
| 16 | apache/shardingsphere | status: won't fix | other |
| 17 | apache/shardingsphere | type: discussion | question |
| 18 | openzipkin/zipkin | techdebt | other |
| 19 | openzipkin/zipkin | wontfix | other |
| 20 | elastic/logstash | invalid | other |
| 21 | GoogleContainerTools/jib | type:bug | bug |
| 22 | quarkusio/quarkus | triage/duplicate | other |
| 23 | quarkusio/quarkus | triage/wontfix | other |
| 24 | codecentric/spring-boot-admin | move-to-stackoverflow | question |
| 25 | zaproxy/zaproxy | wontfix | other |
| 26 | zaproxy/zaproxy | unittest | other |
| 27 | grpc/grpc-java | highly flaky | other |
| 28 | grpc/grpc-java | duplicate | other |
| 29 | facebook/react-native | Stale | other |
| 30 | spring-projects/spring-framework | status: bulk-closed | other |
| 31 | apache/pulsar | lifecycle/stale | other |
| 32 | apache/pulsar | Stale | other |
| 33 | pinpoint-apm/pinpoint | stale | other |

Table 42 (cont.)

| Rule # | Repository | Original Label | Mapped Label |
|---|---|---|---|
| 34 | apache/druid | stale | other |
| 35 | zaproxy/zaproxy | historic | other |

# APPENDIX C

# POST-PROCESSING APPLIED TO THE RESPONSES OF THE LLAMA 3 MODELS

This section presents the details of the manually corrected responses and summarized issue reports for each dataset.

## C.1 Corrected Responses of Llama 3 8B on the Random Sample of the NLBSE'22 Test Set

For below responses, manual corrections were performed.

**Issue Index: 15678**
**Correction:** The classification was corrected as "bug" and additional percent signs `(%)` were removed from the confidence of prediction and the level of issue understandability.

```
**Classification:** Bug
**Prediction Confidence:** 100%
**Issue Understandability:** 100%

The issue is that the code has inconsistent formatting and
spacing, which can lead to errors or make it harder to
read. The `black` tool is designed to automatically format
Python code according to a set of rules, making it easier
to maintain and understand.

To fix this issue, you can install the `black` tool using
pip: `pip install black`, then run it on your code: `black
/path/to/your/code.py`. This will automatically format
your code according to the default settings.
```

**Issue Index: 69657**
**Correction:** The classification was corrected as "bug".

```
Classification: **bug**
Prediction Confidence: 80
```

```
Issue Understandability: 90
```

**Issue Index: 8732**
**Correction:** Additional percent signs `(%)` were removed from the confidence of prediction and the level of issue understandability.

```
Classification: Enhancement
Prediction Confidence: 90%
Issue Understandability: 95%

The issue is about creating a rule that combines the Smart
Shift and Forward buttons to produce the Super_L keystroke
when both are pressed simultaneously. The user has already
created individual rules for each button, but this
enhancement request aims to create a new rule that takes
into account the combination of these two buttons.
```

After these corrections, the distribution of responses for each of the three tasks are provided below.

**Label Classification**

The responses received for the classification of issues are presented in Table 43.

Table 43: Classification of labels by Llama 3 8B on the random sample of the NLBSE'22 test set.

| **Classification** | **Count** |
|---|---|
| enhancement | 2,438 |
| bug | 2,422 |
| question | 120 |
| | 19 |
| other | 1 |
| Total | 5,000 |

**Confidence of Classification**

The responses received for the confidence levels of classifications are presented in Table 44.

Table 44: Confidence levels by Llama 3 8B on the random sample of the NLBSE'22 test set.

| **Prediction Confidence** | **Count** |
|---|---|
| 80 | 3,147 |
| 90 | 1,522 |
| 95 | 180 |
| 100 | 115 |

Table 44 (cont.)

| Prediction Confidence | Count |
|---|---|
| | 19 |
| 60 | 10 |
| 50 | 3 |
| 70 | 2 |
| 20 | 1 |
| 0 | 1 |
| Total | 5,000 |

**Issue Understandability**

The responses received for the understandability levels of issues are presented in Table 45.

Table 45: Understandability levels by Llama 3 8B on the random sample of the NLBSE'22 test set.

| Issue Understandability | Count |
|---|---|
| 90 | 2,680 |
| 80 | 1,271 |
| 20 | 372 |
| 95 | 233 |
| 60 | 208 |
| 100 | 88 |
| 50 | 87 |
| 0 | 21 |
| 70 | 21 |
| | 19 |
| Total | 5,000 |

### C.2 Corrected Responses of Llama 3 70B on the Random Sample of the NLBSE'22 Test Set

For below responses, manual corrections were performed.

**Issue Index: 32970**
**Correction:** The classification was corrected as "enhancement".

```
Classification: patch/enhancement
Prediction Confidence: 90
Issue Understandability: 80
```

**Issue Index: 46667**
**Correction:** The classification was corrected as "other".

```
Classification: task list
```

```
Prediction Confidence: 90
Issue Understandability: 80
```

**Issue Index: 31866**
**Correction:** The confidence of prediction was corrected as "90".

```
Classification: bug
Prediction Confidence:  , 90
Issue Understandability: 95
```

**Issue Index: 79951**
**Correction:** The confidence of prediction was corrected as "90" and the level of issue understandability was corrected as "80".

```
Classification: bug
Prediction Confidence:  , 90
Issue Understandability: , (80)
```

**Issue Index: 16158**
**Correction:** The confidence of prediction was corrected as "90" and the level of issue understandability was corrected as "95".

```
Classification: bug
Prediction Confidence: , (90)
Issue Understandability:  (95)
```

**Issue Index: 76042**
**Correction:** The confidence of prediction was corrected as "95".

```
Classification: bug
Prediction Confidence:  , 95
Issue Understandability: 90
```

**Issue Index: 28110**
**Correction:** The level of issue understandability was corrected as "0".

```
Classification: other
Prediction Confidence: 100
Issue Understandability:  (not understandable)
```

**Issue Index: 7042**
**Correction:** The level of issue understandability was corrected as "100".

```
Classification: question
Prediction Confidence: 90
Issue Understandability:  of 100
```

**Issue Index: 28573**
**Correction:** The back-tick characters (`) were removed from the issue body and the prompt was asked once again to the model. Based on the new response, the level of issue understandability was corrected as "90".

```
Classification: enhancement
Prediction Confidence: 80
Issue Understandability: <|begin_of_text|>201
```

**Issue Index: 28573 - 2nd response**

```
Classification: enhancement
Prediction Confidence: 80
Issue Understandability: , of 90
```

After these corrections, the distribution of responses for each of the three tasks are provided below.

**Label Classification**

The responses received for the classification of issues are presented in Table 46.

Table 46: Classification of labels by Llama 3 70B on the random sample of the NLBSE'22 test set.

| Classification | Count |
|---|---|
| bug | 2,470 |
| enhancement | 2,138 |
| question | 262 |
| other | 111 |
| | 19 |
| Total | 5,000 |

**Confidence of Classification**

The responses received for the confidence levels of classifications are presented in Table 47.

Table 47: Confidence levels by Llama 3 70B on the random sample of the NLBSE'22 test set.

| Prediction Confidence | Count |
|---|---|
| 90 | 3,015 |

Table 47 (cont.)

| Prediction Confidence | Count |
|---|---|
| 100 | 824 |
| 80 | 578 |
| 95 | 559 |
| | 19 |
| 0 | 4 |
| 70 | 1 |
| Total | 5,000 |

**Issue Understandability**

The responses received for the understandability levels of issues are presented in Table 48.

Table 48: Understandability levels by Llama 3 70B on the random sample of the NLBSE'22 test set.

| Issue Understandability | Count |
|---|---|
| 95 | 1,651 |
| 80 | 1,460 |
| 90 | 1,280 |
| 20 | 206 |
| 60 | 138 |
| 50 | 129 |
| 100 | 50 |
| 70 | 36 |
| 40 | 23 |
| | 19 |
| 0 | 7 |
| 98 | 1 |
| Total | 5,000 |

### C.3 Summarized Issue Reports in the Random Sample of the NLBSE'22 Test Set

The indexes of the issue reports for which the issue body were summarized with the Llama 3 70B variant can be seen in Table 49.

Table 49: The indexes of the summarized issue reports in the random sample of the NLBSE'22 test set.

| Number | Issue Index | Number | Issue Index |
|---|---|---|---|
| 1 | 48736 | 11 | 2073 |
| 2 | 1975 | 12 | 2629 |
| 3 | 17453 | 13 | 20277 |
| 4 | 39588 | 14 | 1083 |

Table 49 (cont.)

| Number | Issue Index | Number | Issue Index |
|---|---|---|---|
| 5 | 22018 | 15 | 31806 |
| 6 | 63597 | 16 | 71661 |
| 7 | 37745 | 17 | 24021 |
| 8 | 74882 | 18 | 76006 |
| 9 | 35297 | 19 | 69591 |
| 10 | 30873 | | |

### C.4 Corrected Responses of Llama 3 8B on the Random Sample of the GitHub'23 Test Set

For below responses, manual corrections were performed.

**Issue Index: 3755**
**Correction:** The classification was corrected as "enhancement" and additional percent signs (%) were removed from the confidence of prediction and the level of issue understandability.

```
**Classification:** Enhancement

**Prediction Confidence:** 80%

**Issue Understandability:** 90%

The issue is related to the deployment of a Spring Boot application
on WebLogic server, which fails due to a Jersey (JAX-RS)
configuration error. The error message indicates that the
ResourceConfig instance does not contain any root resource classes,
which prevents the application from starting.

To resolve this issue, you may need to review your Jersey
configuration and ensure that you have correctly configured the
RootResourceUriRules or added root resource classes to your
application.
```

**Issue Index: 7744**
**Correction:** The classification was corrected as "bug" and additional percent signs (%) were removed from the confidence of prediction and the level of issue understandability.

```
**Classification:** Bug Report

**Prediction Confidence:** 90%

**Issue Understandability:** 80%
```

```
The issue report describes a problem where the
ShardingSphereDataSourceFactory fails to create a data source,
resulting in an exception. The error message indicates that the
data source is rejected due to too many connections. The report
provides code snippets and configuration details, which helps
understand the context of the issue.

However, without more information about the specific environment,
configuration, or usage scenario, it's difficult to pinpoint the
exact cause of the problem.
```

**Issue Index: 9427**
**Correction:** Additional percent signs (%) were removed from the confidence of prediction and the level of issue understandability.

```
Classification: Bug

Prediction Confidence: 95%

Issue Understandability: 80%
```

**Issue Index: 5297**
**Correction:** Additional percent signs (%) were removed from the confidence of prediction and the level of issue understandability.

```
Classification: Enhancement
Prediction Confidence: 80%
Issue Understandability: 90%

The issue is about using `rest-client-reactive-jackson` togethe
with `quarkus-keycloak-admin-client` in a single project, but it
seems that there's a conflict between the two libraries. The user
wants to use both libraries together without any issues.
```

After these corrections, the distribution of responses for each of the three tasks are provided below.

**Label Classification**

The responses received for the classification of issues are presented in Table 50.

Table 50: Classification of labels by Llama 3 8B on the random sample of the GitHub'23 test set.

| Classification | Count |
|---|---|
| bug | 2,910 |

Table 50 (cont.)

| Classification | Count |
|---|---|
| enhancement | 1,981 |
| question | 81 |
| | 27 |
| other | 1 |
| Total | 5,000 |

**Confidence of Classification**

The responses received for the confidence levels of classifications are presented in Table 51.

Table 51: Confidence levels by Llama 3 8B on the random sample of the GitHub'23 test set.

| Prediction Confidence | Count |
|---|---|
| 80 | 2,590 |
| 90 | 1,896 |
| 95 | 367 |
| 100 | 103 |
| | 27 |
| 60 | 11 |
| 0 | 3 |
| 20 | 3 |
| Total | 5,000 |

**Issue Understandability**

The responses received for the understandability levels of issues are presented in Table 52.

Table 52: Understandability levels by Llama 3 8B on the random sample of the GitHub'23 test set.

| Issue Understandability | Count |
|---|---|
| 90 | 2,797 |
| 80 | 1,600 |
| 95 | 390 |
| 20 | 65 |
| 60 | 58 |
| 100 | 52 |
| | 27 |
| 50 | 4 |
| 70 | 4 |
| 0 | 3 |
| Total | 5,000 |

### C.5 Corrected Responses of Llama 3 70B on the Random Sample of the GitHub'23 Test Set

For below responses, manual corrections were performed.

**Issue Index: 4621**
**Correction:** The confidence of prediction was corrected as "95".

```
Classification: bug
Prediction Confidence: , , 95
Issue Understandability: 90
```

**Issue Index: 3679**
**Correction:** The confidence of prediction was corrected as "95".

```
Classification: bug
Prediction Confidence: , , 95
Issue Understandability: 90
```

**Issue Index: 2527**
**Correction:** The confidence of prediction was corrected as "95" and the level of issue understandability was corrected as "90".

```
Classification: bug
Prediction Confidence:  a high level of confidence (95)
Issue Understandability: a high level of understandability (90)
```

**Issue Index: 1379**
**Correction:** The confidence of prediction was corrected as "100".

```
Classification: bug
Prediction Confidence: , ,100
Issue Understandability: 90
```

**Issue Index: 2554**
**Correction:** The confidence of prediction was corrected as "95".

```
Classification: bug
Prediction Confidence:  , 95
Issue Understandability: 90
```

**Issue Index: 2436**
**Correction:** The confidence of prediction was corrected as "100".

```
Classification: bug
Prediction Confidence:  , 100
Issue Understandability: 90
```

**Issue Index: 2595**
**Correction:** The horizontal rule characters (---) were removed from the issue body and the prompt was asked once again to the model. Based on the new response, the confidence of prediction was corrected as "80" and the level of issue understandability was corrected as "90".

```
Classification: bug
Prediction Confidence: <|begin_of_text|>20100
Issue Understandability:  (very high)
```

**Issue Index: 2595 - 2nd response**

```
Classification: bug
Prediction Confidence:  , 80
Issue Understandability: 90
```

**Issue Index: 7824**
**Correction:** The back-tick characters (`) were removed from the issue body and the prompt was asked once again to the model. Based on the new response, the confidence of prediction was corrected as "90" and the level of issue understandability was considered as "80".

```
Classification: bug
Prediction Confidence: <|begin_of_text|>201
Issue Understandability: 90
```

**Issue Index: 7824 - 2nd response**

```
Classification: bug
Prediction Confidence: 90
Issue Understandability: 80
```

**Issue Index: 467**
**Correction:** The tilde characters (~) were removed from the issue body and the prompt was asked once again to the model. Based on the new response, the level of issue understandability was corrected as "90".

```
Classification: bug
Prediction Confidence: 100
Issue Understandability: <|begin_of_text|>201
```

**Issue Index: 467 - 2nd response**

```
Classification: bug
Prediction Confidence: 100
Issue Understandability: 90
```

After these corrections, the distribution of responses for each of the three tasks are provided below.

**Label Classification**

The responses received for the classification of issues are presented in Table 53.

Table 53: Classification of labels by Llama 3 70B on the random sample of the GitHub'23 test set.

| Classification | Count |
|---|---|
| bug | 3,025 |
| enhancement | 1,674 |
| question | 265 |
| | 28 |
| other | 8 |
| Total | 5,000 |

**Confidence of Classification**

The responses received for the confidence levels of classifications are presented in Table 54.

Table 54: Confidence levels by Llama 3 70B on the random sample of the GitHub'23 test set.

| Prediction Confidence | Count |
|---|---|
| 90 | 2,918 |
| 100 | 1,247 |
| 95 | 616 |
| 80 | 191 |
| | 28 |
| Total | 5,000 |

**Issue Understandability**

The responses received for the understandability levels of issues are presented in Table 55.

Table 55: Understandability levels by Llama 3 70B on the random sample of the GitHub'23 test set.

| Issue Understandability | Count |
|---|---|
| 95 | 1,771 |

Table 55 (cont.)

| Issue Understandability | Count |
|---|---|
| 90 | 1,715 |
| 80 | 1,373 |
| 60 | 34 |
| | 28 |
| 50 | 24 |
| 100 | 23 |
| 20 | 14 |
| 70 | 13 |
| 40 | 3 |
| 0 | 1 |
| 98 | 1 |
| Total | 5,000 |

### C.6 Summarized Issue Reports in the Random Sample of the GitHub'23 Test Set

The indexes of the issue reports for which the issue body were summarized with the Llama 3 70B variant can be seen in Table 56. The issue report with the index of 3755 was summarized only for the Llama 3 70B variant.

Table 56: The indexes of the summarized issue reports in the random sample of the GitHub'23 test set.

| Number | Issue Index | Number | Issue Index |
|---|---|---|---|
| 1 | 3007 | 15 | 3126 |
| 2 | 9537 | 16 | 1494 |
| 3 | 3767 | 17 | 8426 |
| 4 | 10196 | 18 | 10140 |
| 5 | 7907 | 19 | 9704 |
| 6 | 3577 | 20 | 7528 |
| 7 | 8731 | 21 | 231 |
| 8 | 9111 | 22 | 2045 |
| 9 | 4398 | 23 | 8987 |
| 10 | 827 | 24 | 8231 |
| 11 | 8396 | 25 | 1697 |
| 12 | 7896 | 26 | 5048 |
| 13 | 1437 | 27 | 67 |
| 14 | 3284 | 28 | 3755 |

# APPENDIX D

# DISTRIBUTION OF THE ASSIGNED AND PREDICTED LABELS

This section presents the combinations of the assigned issue labels and the classifications of the models on the random sample taken from the NLBSE'22 test set. These combinations covered 1,541 issue reports where at least one model—either RoBERTa, Llama 3 8B, or Llama 3 70B—classified the issue with a label different from the assigned one. Among the possible combinations, 74 distinct patterns were observed, with their counts and percentages shown in Table 57.

Table 57: Combinations of the assigned and predicted labels.

| Case | Count | % |
|---|---|---|
| bbeb | 189 | 12.36 |
| qqeq | 122 | 7.92 |
| eebe | 117 | 7.59 |
| bbee | 94 | 6.10 |
| beee | 86 | 5.58 |
| qqbb | 76 | 4.93 |
| ebbb | 69 | 4.48 |
| ebee | 63 | 4.09 |
| qbbb | 53 | 3.44 |
| eeeo | 52 | 3.37 |
| bbbe | 49 | 3.18 |
| bebb | 46 | 2.99 |
| qqee | 42 | 2.73 |
| bqbb | 39 | 2.53 |
| bebe | 31 | 2.01 |
| qeee | 28 | 1.82 |
| eeqo | 27 | 1.75 |
| eebb | 27 | 1.75 |
| ebbe | 24 | 1.56 |
| eqee | 24 | 1.56 |
| eeeq | 20 | 1.30 |
| qqeb | 19 | 1.23 |
| ebeb | 16 | 1.04 |

Table 57 (cont.)

| Case | Count | % |
|---|---|---|
| bbeq | 14 | 0.91 |
| beeb | 13 | 0.84 |
| eqeq | 13 | 0.84 |
| eeeb | 13 | 0.84 |
| qeeq | 12 | 0.78 |
| bqeb | 12 | 0.78 |
| qbeb | 11 | 0.71 |
| eeqe | 9 | 0.58 |
| bbqo | 9 | 0.58 |
| bbqb | 9 | 0.58 |
| qbeq | 8 | 0.52 |
| eqbb | 7 | 0.45 |
| qebb | 6 | 0.39 |
| bbeo | 6 | 0.39 |
| eqqq | 5 | 0.32 |
| qqqb | 5 | 0.32 |
| bqeq | 5 | 0.32 |
| beeq | 4 | 0.26 |
| eqeb | 4 | 0.26 |
| ebeq | 4 | 0.26 |
| bbqq | 4 | 0.26 |
| eeqq | 4 | 0.26 |
| bqee | 3 | 0.19 |
| qqbq | 3 | 0.19 |
| beqo | 3 | 0.19 |
| qqbe | 3 | 0.19 |
| bqbq | 2 | 0.13 |
| qebe | 2 | 0.13 |
| qqqe | 2 | 0.13 |
| eebo | 2 | 0.13 |
| bqqq | 2 | 0.13 |
| qbqq | 2 | 0.13 |
| beqq | 2 | 0.13 |
| beqb | 2 | 0.13 |
| ebeo | 2 | 0.13 |
| qbbe | 2 | 0.13 |
| bbbo | 2 | 0.13 |
| beeo | 2 | 0.13 |
| bbbq | 2 | 0.13 |
| qbee | 2 | 0.13 |
| qbqo | 1 | 0.06 |
| ebqo | 1 | 0.06 |
| bqeo | 1 | 0.06 |

Table 57 (cont.)

| Case | Count | % |
|---|---|---|
| qeeb | 1 | 0.06 |
| bqqb | 1 | 0.06 |
| bebq | 1 | 0.06 |
| eeqb | 1 | 0.06 |
| eeoo | 1 | 0.06 |
| qeeo | 1 | 0.06 |
| eqbe | 1 | 0.06 |
| qeqo | 1 | 0.06 |
| Total | 1,541 | 100.00 |