TU WIEN Informatics

Verlässlichkeit von Drittanbieter-Anwendungen in Open Source Forschungs-Software

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Stefan Csukker, BSc
Matrikelnummer 01326273

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Wien, 25. März 2025

Stefan Csukker	Andreas Rauber

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

On Reliability Of Third-Party Applications For Open Source Research Software

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Stefan Csukker, BSc
Registration Number 01326273

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Vienna, March 25, 2025

Stefan Csukker	Andreas Rauber

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Stefan Csukker, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 25. März 2025

Stefan Csukker

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte meiner Familie und meinen Freunden, ganz besonders meinen Eltern, für ihre unerschütterliche Unterstützung und Ermutigung im Laufe der Jahre danken.

Jeder von euch hat eine entscheidende Rolle dabei gespielt, dass ich diesen Meilenstein erreichen konnte. Ich bin zutiefst dankbar für eure Unterstützung und euren Glauben an mich.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to thank my family and friends, especially my parents, for their unwavering support and encouragement throughout the years.

Each of you has played a vital role in helping me reach this milestone. I am profoundly thankful for your support and belief in me.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Keywords Open source research software, Softwarefehler, github, gitlab

Die Eigenschaften von Open-Source Software, nämlich reduzierte Kosten, Anpassbarkeit und die Fähigkeit zur Zusammenarbeit und dem Teilen von Code, machen sie sehr nützlich für Forscher. Wenn Software wiederverwendet wird, können sich allerdings bereits die kleinsten Änderungen auf Forschungsergebnisse auswirken. Aspekte wie die Fehlerhäufigkeit, Stabilität und Leistung sind essenziell, wenn Forscher auf der Suche nach einer verlässlichen Software sind. Der aktuelle Stand der Forschung in Bezug auf Fehler und Fehlerkorrekturen in Forschungssoftware ist nicht bekannt. Diese Diplomarbeit beschäftigt sich mit diesem Thema anhand einer empirischen Studie mit dem Fokus auf Fehlerkorrekturen in Open-Source Forschungssoftware. Als Basis wurden Datensätze vom Software Knowledge Graph und Papers With Code verwendet. Das Sammeln von zusätzlich benötigten Daten, sowie die Klassifizierung von Commit-Nachrichten wurden automatisiert. Die Klassifizierung von Commit-Nachrichten in Code-Änderungen und Fehlerkorrekturen weist eine Precision von 93,75% und einen Recall von 95,17% auf. Die Studie zeigte eine starke positive Korrelation zwischen der Anzahl von Mitwirkenden und der Anzahl an Commits, sowie eine schwache positive Korrelation zwischen der Anzahl von Mitwirkenden und dem Anteil an Fehlerkorrekturen in Commits. Softwareprojekte mit mindestens zwei Entwicklern tendieren zu einem höheren Anteil an Fehlerkorrekturen. Für die dritte Forschungsfrage wurde ein Workflow, welcher Bedingungen enthält, und zusätzliche Kriterien eingeführt anhand derer Code-Repositories gefiltert wurden. Von insgesamt 1,055 Repositories waren fünf ausführbar. Die erneute Berechnung führte in einem Fall zu unterschiedlichen Ergebnissen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Keywords Open source research software, software errors, github, gitlab

The characteristics of Open-Source Software, namely reduced cost, customizability, and the ability to collaborate and share code, make it a great fit for researchers. But when reusing software, even the smallest adjustments can affect research outcomes. Aspects like bug frequency, stability, and performance are essential when a researcher is looking for reliable software. The current state of research on bugs and their fixes in research software remains unclear. This thesis aims to address this topic by conducting an empirical study on bug fixes in Open-Source Research Software, for which I used the datasets from the Software Knowledge Graph and Papers With Code. In addition, I accomplished the automation of data gathering and the classification of commit messages from code repositories. The classification of commit messages into code changes and bug fixes achieved a precision of 93.75% and a recall of 95.17%. Overall, the study showed a strong positive correlation between the number of contributors and commits, as well as a weak positive correlation between the number of contributors and bug fix percentages within commits. Software projects with at least two developers tend to have a higher percentage of bug fixes. For the question of effects of bug fixes, I introduced a workflow consisting of conditions and additional criteria for which repositories are being excluded for further investigations. Out of 1,055 repositories five were executable. The recalculation showed different results in one case.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

Open-Source Software (OSS) has gained importance for public and private companies as well as in research. Open-source means that source code needs to be available for free. Free software projects, on the other hand, do not necessarily provide the code. In the 1980s, the amount of software that was available for free started growing. The concepts of "free" and "open source" still had to be distinguished. The examples of the browsers of Netscape and Microsoft have been provided by Fogel [Fog17], as they were both available for free in the 1990s. But "you couldn't get the source code, and even if you could, you didn't have the right to modify or redistribute it. The only thing you could do was download an executable and run it." To differentiate both variants, licensing models had to be introduced. The licenses that are used nowadays can be split into two types [SAP24]:

- Permissive licenses: Also known as "anything goes" licenses like MIT, Apache and BSD
- Copyleft licenses: When modifying Open-Source Software the resulting code has to be distributed under the same license. Examples are *GNU GPL 3.0*, *GNU LGPL 3.0* and *MPL 2.0*. [AMWH19]

Due to its collaborative nature and transparency, Open-Source Software is a great fit for researchers. The most commonly known and widely used applications include Python, R, RStudio, and MySQL. Those projects will be mentioned again later in this work, as they are part of a dataset that has been used. The advantages of using OSS for researchers are reduced cost, customizability of the software and frameworks, and the ability to collaborate and share code openly.

Despite the possibility of customizing code according to the requirements of researchers, it raises the issue of compatibility. Not every Open-Source Research Software (OSRS)

is compatible with what has been implemented before and potentially cannot be fully integrated. In addition, many OSS projects are receiving constant maintenance. Those changes can remove previous compatibility and therefore always need to be considered carefully. While it might still be compatible, the update can also just include a few changes and bug fixes. But even the smallest adjustment can affect research outcomes. This brings me to the topic of reliability of OSRS. Aspects like bug frequency, stability, and performance are essential when a researcher is looking for reliable software. As I have said, errors or unstable code can significantly affect research outcomes as well as the reproducibility of results.

1.1 Problem Statement

The reliability and correctness of Open-Source Research Software are critical for ensuring the accuracy and reproducibility of scientific research. Despite the widespread adoption of OSRS, there is limited understanding of how often bugs occur in these applications and the potential impact these bugs may have on research outcomes. This gap in knowledge raises important questions about the overall reliability of OSRS.

In this thesis, I address this problem by conducting an analysis of code repositories and related scientific papers. Specifically, I perform an empirical study focusing on the frequency of bugs in these applications. In addition, I examine commit messages and provide insights into the correlation between metrics of software projects and bug fix percentages in commits.

The findings from this research will contribute to a better understanding of the reliability of OSRS. Ultimately, this work aims to enhance awareness when developing and reusing research software, thereby supporting the integrity and reproducibility of scientific research.

1.2 Research Objectives

In the course of the thesis the following research questions will be regarded:

RQ1: To what extent can commit messages be categorized into functional bug fixes and feature changes automatically?
The answer to this question serves as a base for further investigations within this thesis. Different types of data collections as well as repositories that do not provide changes in the form of commits and ultimately identifying functional bug fixes hide several challenges that need to be addressed. A filter that is based on the number of citations is going to be applied to the list of repositories. The remaining repositories will be iterated, for which the commit messages are to be gathered via API calls and classified into bug fixes and code changes. The automatically generated classification is going to be verified manually by picking a subset of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

repositories and thereby commit messages. After the manual verification, precision and recall will be calculated.

RQ2: To what degree do the size of the project/repository (i.e., number of commits, number of contributors) and the percentage of bug fixes correlate?
The aim of this question is to investigate the relationship between the total number of commits, the number of contributors per repository, and the bug fix percentages. It remains unclear whether the size of software projects directly influences bug fix percentages. Certain metrics of repositories, like the total number of commits, will be reused from the first research question. In addition, the number of contributors will be retrieved from APIs. After gathering the necessary data, it will then be compared to the bug fix percentages. The overall results are to be presented as graphs and plots showing the correlation between the metrics. The types of plots will be listed in the next section, 1.3 *Methodology*, in more detail.

RQ3: To what extent do functional bugs, that have been fixed, affect results of OSRS?
Not every bug fix influences results in research, which requires evaluating the effects of bug fixes in detail. I will define a workflow that includes conditions for suitability in terms of reproducibility. In addition, specified criteria, such as the availability of dependencies, the runtime, and the size of the dataset, will be applied to a set of repositories. If possible, the results of the ones that remain after this step will then be recalculated.

1.3 Methodology

1. Query the triples of the SoftwareKG [SZK20] and extract open source (research) software. In addition, find relevant papers and repositories on paperswithcode[1].

2. Empirical study
   The study will provide an overview of the distribution of bug fix percentages in Open-Source Research Software, focus on the domain of data analytics, and include the following steps:

   - Define a minimum number of repositories for the sample size used within the study.
   - Define a threshold for the citation count at which papers are being considered and included in the study.
   - While accessing the code repositories via API to get the commits, count the total number of commits per repository.

[1] https://paperswithcode.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Use a heuristic that analyzes the commit messages and classifies them
  The commits will be checked by a heuristic for keywords like “bug”, ”fix”, or “error”. Further, the keywords need to be improved and adjusted in the course of the thesis, since this heavily depends on the available repositories and how commit messages have been written.
- Based on the gathered data, create the following plots to compare the metrics of the code repositories:
  - Bar charts that show the distribution of bug fix percentages across code repositories (one for a single contributor and another one for several contributors)
  - Box plots that show the commits per repository having a single contributor
  - A scatter plot that shows the relationship between the number of contributors and the number of commits per repository
  - A scatter plot that shows the relationship between bug fix percentages and the number of contributors per repository
  - A scatter plot that shows the relationship between bug fix percentages and the number of commits per repository

3. Define criteria based on which repositories are included or excluded as candidates to rebuild the software and recreate the results.
4. Find repositories that had bug fixes after the publication.
5. If possible, rebuild the software and compare results from before and after the bug fix has been applied. In addition, evaluate if the results within the paper have changed due to the bug fix.

1.4 Structure of the Thesis

The thesis will be split into the following chapters:

Related Work In this chapter, I will explore existing work that deals with the same or similar topics. The literature is grouped into the areas of *Trustworthiness and Reliability*, consisting of *Based on Social Aspects* and *Based on Code*, *Gathering of Data*, and *Reproducibility*.

Study Design and Implementation Study design and implementation give an insight into topics that are specific to this thesis. It starts with the gathering and preparation of datasets that describe the data of SoftwareKG and Papers With Code. Second, the study design of the thesis will be explained. It gives an overview of how the data has been processed. The last section deals with the iteration and filtering of data, the patterns of the spaCy framework, as well as how additional data was retrieved.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Evaluation In this chapter, the three research questions and, thereby, the results will be elaborated. Each section is directly related to the respective question.

Summary and Conclusion In the end, I will sum up the results and discuss the outcome of the research questions. In addition, I will discuss whether the problem statement of the thesis has been resolved. Further, the limitations of applied methods and used data will be discussed. In the end, potential future topics that can be based on this work will be proposed.

1.5 Summary

This thesis deals with the reliability of third-party applications for open-source research software. The main focus is on bug fixes and the effects of such fixes. Accurate and reproducible results are essential in research. Being dependent on third-party applications can interfere with this substantial aspect. The research questions are dealing with gathering the necessary data, automatically classifying commit messages from code repositories, investigating the relationship between the size of software projects and bug fix percentages, and to what extent functional bug fixes are affecting the results of research software. What follows is the state-of-the-art research literature on this topic and aligning areas of research.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Related Work

This chapter will provide an overview of the state-of-the-art literature on the topic of the reliability of third-party applications for open source research software. It will be split into the sections *Trustworthiness and Reliability*, describing factors of software projects that have an influence on them, *Gathering of Data*, describing how other research projects approached the topic of this thesis, and *Reproducibility*, describing the challenge of reproducing and rebuilding research software.

I started the search off on Google[1] to get an overview about the topics *bugs in research software*, *empirical study on bugs in OSRS*, *paper withdrawals due to incorrect findings*, and *reproducibility issues*. I used the ACM Digital Library[2] for the in-depth search, as it provides different types of research literature focusing on the field of computing. The advanced search offers the option to search for specific keywords in the title, the abstract, and the full text. The results can also be limited to a certain time period in terms of publication dates. In addition, I used ChatGPT[3] to get an overview of available papers on the mentioned topics.

The first search was based on the keywords *error* and *bug* combined with *research software*, *open source software*, or *open source research software*. It has been executed while having *Title*, as well as *Full text* selected for the field *Search within*. In addition, I was specifically looking for literature that is dealing with software repositories like *GitHub* and *GitLab*. The second phase included the word *reliability* instead of the keywords *error* and *bug*, and it was again combined with *research software*, *open source software*, or *open source research software*. What followed was a search for empirical studies on open source (research) software.

[1] https://www.google.com/
[2] https://dl.acm.org/
[3] https://chatgpt.com/

The selected time period in terms of the publication date was 2000 onwards. In addition, no filters have been applied in regard to specific types of literature, like books, journals, or magazines. The selected papers expressed their relevance by their number of citations and downloads. For example, the paper "Open source software reliability model: an empirical approach" [ZD05] has 65 citations and 2,473 downloads as of September 2024. As mentioned above, the literature has been grouped into the areas of *Trustworthiness and Reliability*, *Gathering of Data*, and *Reproducibility*.

2.1 Trustworthiness and Reliability

2.1.1 Based on Social Aspects

There has been an analysis of software reliability following reported bugs by Zhou and Davis [ZD05]. They used criteria like project duration and activity in their work on software reliability. Activity is based on the number of downloads and the number of times the project has been mentioned in the forum of SourceForge.net. Combining those measures resulted in an activity index. SourceForge provided a "Most active" list, which was their starting point. Their results for the bug arrival trend showed that it slowly increases until it reaches a peak before it decreases and stabilizes at a lower level. Furthermore, they checked the correlation between bug arrival patterns and statistics from SourceForge such as page view and download patterns. Out of the six projects that they considered for their work only one project showed a strong correlation between the monthly bug arrival rate and both monthly page view and downloads numbers. They concluded that those metrics rather measure a project's user acceptance level, than serving as quality indicator.

Without focusing on code or the implementation of Free Open-Source Software (FOSS), Alami [Ala20] defined three enablers of software quality attributes, namely code review, Pull Request (PR) governance, and continuous improvement, as well as two attributes (active participation and retention of developer, as well as active user resources). His reasoning was that those factors bring motivation among the participants and passion for the project. In addition, his interviews and survey showed that the mentioned enablers and attributes have a positive influence.

The papers showed that aspects like page views and download numbers, according to Zhou and Davis, are not a good fit for quality indication, while others, like code review and active participation, proved to have a positive impact. In contrast, I will focus on measurements that are solely based on the code of projects, as the investigation regarding the correlation of social aspects and reliability showed difficulties. In addition, the correlation of hard facts of OSRS and bug fix percentages will be evaluated.

Another aspect to define the reliability of OSS has been introduced by Schueller et al. [SW24]. The team carried out a study on a broader basis. Their previous work on this topic served as a data source, where they created a database for Rust projects that is able to present technical dependencies and developer contributions. It is based on "over

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

five million distinct contributions of over 72 thousand developers, contributing to over 74 thousand libraries over eight years." [SWS+22] Their main focus was the share of contributions by developers, calculated by the number of commits per repository, and how different projects can rely on a single person. The results showed that within the 100 most downloaded libraries, there are "individual developers [that] would introduce a correlated shock impacting the future functionality of the system." [SW24] In addition, the team was investigating the correlation between the importance of libraries and certain metrics of repositories like dependencies and GitHub stars. While the dependencies showed a moderate correlation, the correlation considering GitHub stars was weaker. While Schueller et al. also examined relationships between projects, I will focus on the repositories as an "enclosed environment". The research question regarding the correlation between the size of projects and bug fix percentages only considers numbers within the respective repositories.

2.1.2 Based on Code

Lavazza et al. [LMTT10] used a different approach to assess the trustworthiness of software. They used the so-called Elementary Code Assessment rules that are solely based on code analysis without requiring external data or historical information. These rules target common coding issues that can degrade software quality, such as excessive class length, missing braces in control structures, and the presence of unused code. By automatically detecting and correcting these types of issues, the ECA method aims to improve both the readability and maintainability of the code and thereby implicitly also the reliability. In addition, Lavazza et al. highlighted that adhering to ECA rules can lead to fewer defects and a more robust codebase, particularly in open-source software (OSS), where the absence of formal certification processes makes code quality critical to establishing trust.

Another option on how to improve the reliability of software is the so-called Continuous Code Quality (CCQ), which is a Continuous Integration (CI) principle. In CI, code changes are continuously integrated into a project by using an automated build process. CCQ involves automatic tests and static/dynamic code analysis. A study by Vassallo et al. [VPBG18] shows that only 11% of builds use CCQ principles. One of their conclusions is that "code quality is not the top priority for developers, who prefer not to improve the existing code for different reasons, including time pressure or laziness."

To sum up, the mentioned approaches to improve the trustworthiness of software projects are based on code and code quality. They include rules that consider coding issues, as well as CCQ, which automatically tests the code. Opposing this approach, I will focus on social aspects that were mentioned in the previous section. On the other hand, the empirical study within this work will include bug fixes, which can be seen as a hard fact based on code. It is a metric that will be considered for the investigation on the correlation of certain metrics.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.2 Gathering of Data

Yang et al. [YBLL17] tried to predict the severity of bugs within open source projects. This goes a step further than what is to be worked on within this thesis. The team decided to focus on Eclipse, Android, and JBoss. All of those projects have tracking systems from which they collect the reports via JSON and CSV. The classification of bugs was based on an emotion words-based dictionary and considered the title and description of bug reports. They managed to get a recall of 90% (Eclipse), 97% (Android), and 79% (JBoss). Unfortunately, no reason has been provided why those open source projects were chosen for their work. Opposing this approach, I will consider a sample of at least 1,000 projects and repositories.

Schwittek and Eicker [SE13] carried out an empirical study on the reuse of commercial-off-the-shelf (COTS) and Open-Source Software. They implemented a tool called the "Component Decision ETL Pipeline". The tool focuses on Java web projects and downloads web archive files from the respective websites. It has not been mentioned how the web applications were selected. The information regarding the criteria was that the chosen websites are focusing on serving enterprises. In the end, 36 web applications were part of their study and included websites like Jenkins and Nexus. The tool was searching for Java archives within the application. The result was that "the number of reused components ranges from 16 to 161 components with an average number of about 70." Again, the study was based on a small sample of 36 web applications. As mentioned before, I will carry out the study within this work on a broader basis and also try to automatise as much as possible.

The Retraction Watch (RW) database [4] contains scientific articles that have been retracted for various reasons. In total the search form provides 107 reasons for retraction as of October 2024. The following list contains a few examples:

- Concerns/Issues about Data
- Concerns/Issues about Results
- Error in Methods
- Error in Results/Conclusions
- Unreliable Data
- Unreliable Results

Schindler et al. [SYSK23] investigated the correlation between errors in software and the retraction of papers. They used retraction notices from the RW database and also the SoftwareKG [SZK20] to gather data. The latter contains software mentions for all

[4] http://retractiondatabase.org/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

PubMed Central[5] publications covering 11.8 million software mentions in 3.2 million publications. The entries of the SoftwareKG with publication type "retraction" were marked for further investigation. Linking those to the source of PubMed Central led to 1,924 retraction notices. Schindler et al. obtained the plain text of the publications, which they got from S2ORC [LWN+20], a corpus consisting of metadata, abstracts, and bibliographic references. Their dataset contained articles that were published between 2000 and 2019, as there were either too few data samples (before 2000) or the full-text document was not available (after 2019). The most commonly used software, a total of 243 tools, was analyzed manually and enriched with information on whether the software is free or commercial. It has not been done for the full dataset, because, according to Schindler et al., it was a considerable manual effort. Further, the reason for the retraction of each publication was analyzed. The reasons were grouped into *Software Errors*, *Study Design*, *Journal Policy*, and *Reporting Error*. The team identified 16 retraction notices (out of 1,924) for which the software was related to the retraction of the publication. The most common reasons were biased study results caused by errors in software usage (6), as well as source code errors leading to data processing errors (3). In addition, for four instances, the study design, including the usage of specific software, led to unreliable results. Two notices violated the journal policy, and one notice had a wrongly reported version of SPSS as contributing reason. Overall, their results showed that in retracted articles, less FOSS is employed in contrast to commercial software. This is also the case within specific retraction reasons like *Error*, *Investigation*, *Misconduct*, and *Self Plagiarism*. The approach of Schindler et al. was to access retracted articles directly and start from there. Even though the information of the retraction was known, analyzing the articles had to be done manually. Opposing this approach, I will consider published articles and rebuild their results in case the used software contained errors or bug fixes.

The International Conference on Mining Software Repositories is analyzing data within software repositories in order to gather information about software systems and projects. [MSR23] The following papers have been published as conference papers and deal with the topic of bug fixes in OSRS. In the case of Ponta et al. [PPS+19], data about vulnerabilities in OSS has been gathered manually. Their self-developed tool is focusing on code, opposing the approach of other tools that are purely gathering metadata. As the manual identification of fixes is expensive and time-consuming, the team tried to introduce an automated approach. For every new vulnerability that has been acknowledged within their company, they reviewed the available information and tried to find the fixes in the related code repository. Over 70% of the National Vulnerability Database vulnerabilities did not have any link to a source code repository. All the steps have been carried out manually, including identifying the fix for the vulnerability. For that they relied on the text of the advisory, the commit messages, and the code changes. The last step required expert knowledge and, therefore, had to be carried out manually. Similar to the work within this article, it requires a lot of time to find bug fixes in code repositories. That is why the approach within this thesis is to automate finding the fixes of errors and

[5] https://pmc.ncbi.nlm.nih.gov/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

to mark those accordingly. Based on that, the results will be rebuilt manually as the software projects are different in terms of the building process. This aligns with the manual verification in the mentioned paper that required expert knowledge.

Similar to the work of Yang et al. [YBLL17], Wang et al. [WBJS20] based their empirical study on specific projects instead of using a dataset consisting of repositories or bugs. The chosen merged PRs came from the OSS repositories of Apache, Mozilla, Google, and Facebook. With their work focusing on regular expressions, as well as selecting only Java, JavaScript, and Python, they found 195 repositories in total that are part of the mentioned organizations. The inspection of the merged PRs has been carried out manually. While they searched for *regular expression* and *regex* via GitHub GraphQL API and got 664 results, their manual verification resulted in 350 PRs, which amounts to only 53%. In the end, they discussed that their sample of PRs was not large and there is an opportunity to automate some of the applied steps. As mentioned before, the approach of my work is to use a list of repositories in the first place and to automate the search and marking of the bug fixes. Automating this step within this thesis is supposed to result in a higher accuracy in finding bug fixes.

2.3 Reproducibility

The third research question of this work requires reproducing results of papers. This includes rebuilding and executing the provided software. Ivie et al. [IT18] described reproducibility as carrying out the same tasks of the original researcher while differing in some ways, like non-availability of software or hardware. Reproducibility is given when this still leads to the same results. Within this section I will elaborate on the state of research in the area of reproducibility.

Collberg [Col14] performed a study on reproducible research in which he scanned papers for code and related links to code repositories. Every time code was found related to a paper, it was assigned to a student research assistant. They set a limit of 30 minutes (excluding download and install time) to complete the task of running the software. In case, after the 30 minutes, there were unresolved make or compilation errors, the software was marked as “did not compile”. In case it compiled but did not run, it was marked as “compiles but does not run”, and after a successful execution without errors, it was marked as “runs”. Out of 410 code repositories (not including 98 non-practical papers and 105 repositories that were “not requested”), they were able to execute 102 (24.9%) applications. The most common reasons why they failed to run the code were as follows:

- Prerequisite failed to build (16.9%): Time limit of 30 minutes.
- Missing third-party package (16.2%)
- Distribution is missing files (15.4%)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Contrary to the mentioned paper, the starting point within this work will be an existing dataset containing links to code repositories. On the other hand, a specific time limit when rebuilding the software will help to reduce wasting time on potentially non-executable code.

Boettiger [Boe15] dealt with the topic of reproducibility in the area of computer systems research. He stated that reproducible research is becoming more important and introduced the concept of Docker, combining several aspects of computer systems. In his work, he discussed four technical challenges that Docker aims to solve or at least reduce the risk of studies being irreproducible:

- Dependency hell: The study of Collberg [Col14] showed that less than 50% of software could be built or installed, and only 24.9% was successfully executed. In some cases it is possible to overcome missing dependencies by applying domain knowledge. Those differences can ultimately lead to differing results. Docker provides an image in which the software has been installed, including potentially outdated dependencies.
- Imprecise documentation: Dockerfiles overcome the issue of imprecise documentation by providing an automated build process including the installation of all the necessary dependencies and required steps to run the software.
- Code rot: Regular updates can change the results. Even though the underlying code has not changed, the evolved ecosystem surrounding the application can influence the software, which is known as *code rot*. Since Docker images still receive updates for their underlying operating system, the challenge of *code rot* can be overcome by providing a binary copy of the initial image, which can be created as a portable tarball file.
- Barriers to adoption and reuse in existing solutions: There is a significant barrier to learning additional tools that would help to reduce the risk of irreproducibility. While Docker requires some time to learn, it eliminates the need for additional tools by providing everything necessary to create fully functional, portable environments.

As stated above, reusing a Docker image to build and run the research software would reduce the risk of irreproducibility. Unfortunately, the content of the datasets that will be used within this thesis cannot be influenced. Creating a Docker image with the given code and dependencies does not guarantee a successful execution of the software. In addition, when creating an image for an unknown project, the necessary time effort can be significantly higher than the defined time frame of 30 minutes by Collberg [Col14].

2.4 Summary

When defining the reliability of software, the existing literature used social aspects like project duration, code review, and PR governance. In addition, other teams were using

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

predefined quality measurements like the "Elementary Code Assessment" and continuous code quality. While Schueller et al. [SW24] carried out an empirical study on a broader base focusing on Rust projects, there is no work that is focusing on bug fixes. The team was exploring the contributions of developers and how much the projects were depending on single contributors. The study within this thesis will be based on the Software Knowledge Graph and Papers With Code and aims to have a sample size of at least 1,000 repositories. In several papers that were mentioned above, most of the steps have been carried out manually, which the researchers also blamed to be time-consuming. The iteration through the code repositories as well as the classification of bug fixes will be automated within this thesis. In addition, the most important take-away from this section will be the 30-minute time limit used by Collberg [Col14]. It can be difficult and time-consuming dealing with unknown applications and their dependencies. In his work, the team was able to successfully execute only 24.9% of the software. The most common reasons included missing third-party packages, as well as missing files.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Study Design and Implementation

This chapter provides an overview of the processing of the data that was necessary to carry out the empirical study. First, the preparation steps will be discussed in detail. The section is split into the datasets of the SoftwareKG as well as Papers With Code. Second, a description of the study design will be provided that explains the workflow of the data. In the end, Section 3.3 *Explanation of Thesis Specific Implementation* gives insights into the iteration and filtering of data, spaCy patterns, and how the data was gathered for the plots that will be discussed in Chapter 4 *Evaluation*.

3.1 Preparation of Datasets

3.1.1 SoftwareKG

The Software Knowledge Graph is a combination of two projects from the areas of social and medical sciences. It "contains information about software mention statements from more than 51,000 scientific articles". It helps to work with reused research software and provides information about whether the source code is available. [SZK24]

Initially, I parsed the data locally by downloading the full dataset of the SoftwareKG [SZK19]. It contains nearly four million triples, of which 20,227 refer to software. The software is available for 133 entries, including both free and applications that are not free. At first, the *rdflib.graph* package was used to access the local dataset directly. It took around 90 seconds to complete the parsing of the complete file. On the website [1] of the SoftwareKG project, a SPARQL query editor can be found. The same queries can be built in Python using the *SPARQLWrapper* framework. Compared to *rdflib.graph*, the parsing of the data is significantly faster.

[1] https://data.gesis.org/softwarekg/sparql

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

First, I verified the numbers that can be found on the website for "source code available" by executing the following query:

```
    sparql = SPARQLWrapper(endpoint=
            'https://data.gesis.org/softwarekg/sparql')
    sparql.setReturnFormat(JSON)
    sparql.setQuery("""
        SELECT count(?sourceAvailable) as
        ?count ?sourceAvailable WHERE { ?s
        <http://data.gesis.org/softwarekg/sourceAvailable>
        ?sourceAvailable . }
        GROUP BY ?sourceAvailable
    """)

    ret = sparql.queryAndConvert()

    for r in ret["results"]["bindings"]:
        print(r)
```

Listing 1: Query to find applications for which source code is available

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The results aligned with what has been provided on the website. The query returned 52 applications (S_{skg}) having available repositories and 81 not offering the code publicly. Second, I printed the triples for which the source code is available:

```
sparql = SPARQLWrapper(endpoint=
        'https://data.gesis.org/softwarekg/sparql')
sparql.setReturnFormat(JSON)
sparql.setQuery("""
    SELECT ?s WHERE { ?s
    <http://data.gesis.org/softwarekg/sourceAvailable>
    "true"^^<http://www.w3.org/2001/XMLSchema#boolean>
    .}
""")

ret = sparql.queryAndConvert()

for r in ret["results"]["bindings"]:
    print(r)
```

Listing 2: Query to print triples for which source code is available

This prints the results in the form of: *{'s': 'type': 'uri', 'value': '<link to repository>'}*. A concrete example for a triple can be found in Listing 3. Further, Table 3.1 shows the content of S_{skg}, providing the name of the repository, the postfix of the SoftwareKG URI, and the link to the code repository. The commits from the repositories have been used in the classification, which will be explained in Section 3.3.

```
's':
{
    'type': 'uri',
    'value': 'http://data.gesis.org/softwarekg/software/0'
}
```

Listing 3: SoftwareKG SPARQL query result example

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

ID	Name	SoftwareKG URI postfix	Repository URL
1*	Audacity	182	https://github.com/audacity/audacity
2*	Bioconductor	107	https://github.com/grimbough/code.bioconductor.org
3*	BEAST	26	https://github.com/beast-dev/beast-mcmc
4	BLAST	18	no repository available
5*	Cytoscape	57	https://github.com/cytoscape/cytoscape
6	DnaSP	76	no repository available
7*	EEGLAB	52	https://github.com/sccn/eeglab
8*	EpiInfo	222	https://github.com/Epi-Info/Epi-Info-Community-Edition
9*	FigTree	99	https://github.com/rambaut/figtree
10*	GATK	70	https://github.com/broadinstitute/gatk
11	GenAIEx	86	no repository available
12*	GENEPOP	78	https://gitlab.mbb.univ-montp2.fr/francois/genepop
13*	GeoDA	263	https://github.com/GeoDaCenter/geoda/
14*	Ggplot2	97	https://github.com/tidyverse/ggplot2/
15	GIFT	219	https://sourceforge.net/projects/gift/
16	Haploview	74	https://sourceforge.net/projects/haploview/
17*	HMMER	358	https://github.com/EddyRivasLab/hmmer
18*	Igraph	113	https://github.com/igraph/igraph
19*	ImageJ	9	https://github.com/imagej/ImageJ
20	Java	67	no repository available
21	limma	256	part of Bioconductor
22*	lme4	24	https://github.com/lme4/lme4/
23*	MAFFT	71	https://gitlab.com/sysimm/mafft
24*	MaxEnt	37	https://github.com/mrmaxent/Maxent
25*	MEGA	23	https://github.com/meganz/sdk
26*	MrBayes	51	https://github.com/NBISweden/MrBayes
27*	MuMIn	118	https://github.com/munin-monitoring/munin
28*	MUSCLE	58	https://github.com/rcedgar/muscle/tree/main
29	MySQL	307	https://github.com/mysql/mysql-server
30	nlme	88	read only mirror; no meaningful commits available
31	OpenBUGS	585	no repository available
32	OxCal	171	no repository available
33*	PAML	206	https://github.com/abacus-gene/paml
34	PHYLIP	144	no repository available

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

35*	Plink	21	https://github.com/chrchang/plink-ng
36*	Praat	49	https://github.com/praat/praat
37*	Primer-E Primer	93	https://github.com/primer3-org/primer3
38	Psychophysics	56	part of Psychophysics Toolbox
39*	Psychophysics Toolbox	381	https://github.com/Psychtoolbox-3/Psychtoolbox-3
40*	PsychoPy	243	https://github.com/psychopy/psychopy
41	Psychtoolbox	126	part of Psychophysics Toolbox
42	Python	22	https://github.com/python/cpython
43*	QGIS	83	https://github.com/qgis/QGIS
44*	QIIME	152	https://github.com/qiime2/qiime2
45	R	0	https://github.com/r-hub/rhub/
46*	RAxML	85	https://github.com/stamatak/standard-RAxML
47	RStudio	96	https://github.com/rstudio/rstudio
48*	SPM8	14	https://github.com/spm/spm
49	stats	154	part of R
50*	STRUCTURE	61	https://github.com/rajanil/fastStructure
51	TCS	237	no repository available
52*	Tracer	54	https://github.com/beast-dev/tracer

Table 3.1: 52 triples of the SoftwareKG for which the source code is available for free. The first column includes a mark (*) for the entries that have been retained for the study. I will provide the reasons for excluding certain triples in Section 3.2. The third column shows the postfix of the URI *http://data.gesis.org/softwarekg/software/*
e.g. http://data.gesis.org/softwarekg/software/0

3.1.2 Papers With Code

Papers With Codes' [2] mission is "to create a free and open resource with Machine Learning papers, code, datasets, methods, and evaluation tables". It offers papers in the areas of machine learning, computer science, physics, mathematics, astronomy, and statistics. The team is using machine learning and scraping websites to provide papers and software that has been implemented in the course of the research project.

In order to have more relatable topics and understandable software, the data from the area of computer science within Papers With Code has been chosen. The respective data is available under https://cs.paperswithcode.com/about. "Links between papers and code" provides a JSON file containing links to code repositories and related papers. The most important key-value pairs of a single entry are as follows:

- paper_url: Link to the paper

[2] https://paperswithcode.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
{
    "paper_url": "https://paperswithcode.com/
    paper/<paper title>"
    "paper_title" : "<paper title>",
    "paper_arxiv_id" : "1234.5689",
    "paper_url_abs": "https://arxiv.org/
    abs/<paper_arxiv_id>",
    "paper_url_pdf": "https://arxiv.org/pdf/
    <paper_arxiv_id>.pdf",
    "repo_url": "https://github.com/<github user>/
    <repo name>",
    "is_official": true,
    "mentioned_in_paper": true,
    "mentioned_in_github": true,
    "framework": "tf"
}
```

Listing 4: JSON "Links between papers and code" from Papers With Code showing entry for paper with link to code repository

- paper_arxiv_id: Identifier of the arXiv archive website (DOI). DOI is used to uniquely identify papers online.
- repo_url: Link to the code repository
- is_official: A flag that is true when the code repository is the main (initial) implementation.

Besides the mentioned key-value pairs, it provides *paper_title*, *paper_url_abs*, *paper_url_pdf*, *mentioned_in_paper*, *mentioned_in_github*, and *framework*. A generalized example entry can be looked up in Listing 4. The file contains duplicates within the fields *paper_url* and *paper_arxiv_id*, which I had to restrict in order to not consider a code repository more than once. Having duplicates in the result set would lead to inconsistencies in the study later on. One approach was to use the *is_official* flag. But, when filtering for *is_official* = *true*, the first batch of 10,000 entries returned only a single result. Thus, applying the filter on the whole dataset (around 235,000 papers) would only provide roughly 235 results. Since the target sample size was 1000, using the mentioned flag did not provide enough repositories.

The alternative solution was to use the link to code repositories as a unique identifier. Even though links sometimes occur more than once in the file, it does not interfere with the study results. This just means that several papers have been cited by the paper

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

related to the implementation. Using the repository URL instead, the first batch of 10,000 entries returned 77 results. In some cases, *repo_url* contained a link to a single file within a code repository. This means that one file contains the implementation of the paper. In some cases, the repository consisted of additional files that contained implementations of other papers.

In order to keep the samples comparable – most of the links are referring to repositories – those occurrences have been filtered out. Considering single files bears several issues:

- Considering single files does not align with the overall study design, which focuses on repositories.
- Additional logic is required to retrieve related commits.
- The first appearances did not contain a documentation for single files, but rather an overall documentation for the repository.

The arXiv ID was used to retrieve the citation count of papers. Entries without the identifier had to be excluded because it was not possible to uniquely identify the paper and obtain data such as the number of citations. The reason for considering papers that exceed a certain threshold of citations is that software cannot cause any issues for other research applications if it has not been reused. The citation count was retrieved from the API of datacite.org [3], which provides general metadata for research papers, including the *citationCount* field. A pseudocode and description can be found in Section 3.3.1 *Iteration and Filtering of Datasets*.

3.2 Study Design of Thesis

Initially, two datasets from Papers With Code and the SoftwareKG have been used as starting point. As mentioned in Section 3.1.2, the dataset from Papers With Code contained a list of papers containing links to code repositories. In total, it provided 234,166 entries.[4] On the other hand, the SoftwareKG results had 3,998,194 triples, out of which 52 had source code publicly available (S_{skg}). Out of those, 8 did not provide repositories (line numbers 4, 6, 11, 20, 31, 32, 34 and 51 in Table 3.1). In addition, four projects were part of others and have not been considered as well (line numbers 21, 38, 41 and 49 in Table 3.1). Furthermore, three of the repositories were outdated and did not provide commits or a version history of files (line numbers 15, 16 and 30 in Table 3.1). The rows 15 and 16 contain the projects *GIFT* and *Haploview* and were uploaded to Sourceforge. Another decision was to exclude the following repositories (line numbers 29, 42, 45 and 47 in Table 3.1):

- https://github.com/mysql/mysql-server

[3] https://api.datacite.org/

[4] The dataset has been downloaded from https://cs.paperswithcode.com/about on May 8, 2024

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- https://github.com/python/cpython
- https://github.com/r-hub/rhub/
- https://github.com/rstudio/rstudio

The focus of this work was on Python but also on Open-Source Research Software in general. Those repositories do not provide results that have been published in papers and therefore have been excluded from further steps.

While having 33 repositories left from SoftwareKG, the target sample size for the statistics in RQ2 was 1,000 repositories. To exceed this number, the first 125,000 entries of Papers With Code had to be considered. As mentioned in Section 3.1.2, a threshold of citations has been implemented, which amounts to 20 citations for every paper. With the repositories from SoftwareKG, the resulting sample (S_i) contained 1,055 entries. Those have been used for the work on RQ1 and RQ2.

What followed were API calls to gather commit messages for all the repositories. The classification of commit messages has been done separately for each project and will be described in detail in the next section 3.3. To verify the results, a manual verification process has been carried out. A continuous block of 50 repositories has been selected for which the actual number of bug fixes, as well as precision and recall, were calculated. The exact repositories and results will be mentioned later. The work up to this point concluded the answers for RQ1 and RQ2. For the third research question, a workflow consisting of conditions and a list of criteria has been defined, which includes and excludes repositories. Since the first approach did not result in any repositories that would allow to perform the intended evaluation, a second and third approach will be introduced.

3.2.1 First Approach to Elaborate Effects of Functional Bug Fixes

The content of the first research question was to determine a dataset that allows to address RQ3. The first approach to answer the third research question included the mentioned set of repositories coming from RQ1. In general, visual and sound-related papers were excluded due to the lack of domain knowledge. At first, projects with no bug fixes were removed, as a comparison of different versions would not show any differences. Second, the remaining repositories have been checked manually. The criteria to include or exclude code repositories were as follows:

- **Outliers** – in terms of bug fix percentages and the number of commits – are desired and can provide a deeper insight into potential effects.
- **Dependencies** must be available for free.
- The **estimated runtime** must not be too long (less than one hour is preferred). This criteria only applied to applications that provided the runtime in the documentation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- The size of the necessary **dataset** must not exceed 1 gigabyte.
- Clearly defined **tasks** in the form of a how-to or documentation must be provided.
- Special **hardware** must not be needed.

Ultimately, the software was supposed to be rebuilt with the latest available version and a former version like the one that has been used for the publication of the related paper. The expected outcome was that the two builds were creating different results in case the bug fixes that took place in between were affecting it.

3.2.2 Second Approach to Elaborate Effects of Functional Bug Fixes

The second approach for research question 3 was to find code repositories that contained bug fixes after the related paper had been published. The **full** data set of Papers With Code has been considered, including a filter for the *is_official* flag. Entries that have the value "true" are the official implementation of the paper. Since I was not able to find an API that provides the exact publication date for arXiv[5] papers, I decided to retrieve the publication year from the datacite API. I added January 1 ("01-01") to the publication year and compared it to the date of every (classified) bug fix. In case the bug fix has been applied after the paper has been published, the entry of Papers With Code has been saved for further analysis. Again, the necessary threshold of citations for every paper was 20. After that, the actual publication dates have been retrieved manually. In addition, the dates of the bug fixes have been compared to the exact publication date.

The resulting list of candidates was verified manually. The bug fixes and, thereby, dates have to be checked in the repository. In case the paper had bug fixes post-publication, the following values are to be presented as a timeline:

- Publication date of the paper
- Dates of bug fixes

3.2.3 Revisiting First Approach

For the third approach, I reused most of the criteria from the initial approach. The significant adjustment was to include visual and sound-related papers and thereby results (see Section 3.2.1). Rather than initially excluding them, a paper has been considered for the reproducibility whenever it contained any kind of numerical results. This also included graphs (e.g., showing error rates or accuracy) that can be reproduced by going through the steps in the documentation. In order to reduce mistakes during the process of reproduction, it has been split into two phases. The first phase was considering the paper and the documentation in the code repository. The second phase included

[5] https://arxiv.org/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

double-checking the bug fixes, as well as verifying whether bug fixes took place after the initial publication of the paper.

The first phase has been visualized in Figure 3.1. The dataset (S_i) that has been created during RQ1 was reused for this process. At first, every repository without bug fixes was removed. Next, I checked whether the dataset of 3.1.2, the readme within the code repository or the website of the project contained a link to the original paper. This has been achieved by using the search string *"https://github.com/oellop/facial_expression",\n....."is_official": true* in the text editor Notepad++ while searching within the dataset of Papers With Code. In this example, *oellop* denotes the name of the GitHub user and *facial_expression* was the name of the repository. The four dots are depicting spaces. In case the repository was located on GitLab, *github.com* was replaced by *gitlab.com*, as the structure of the link and the entry within the dataset was the same otherwise.

If the original paper was available, it was searched for numerical results as mentioned before. The next step was then to check whether the code repository contained documentation in the form of a readme file or a link to external documentation. Lastly, the following criteria had to be fulfilled:

- **Dependencies, additionally required software, and datasets** must be available for free. This also included general availability.
- The **estimated runtime** must not be too long (less than one hour is preferred). Before the execution, this criteria only applied to applications that provided the runtime in the documentation.
- The size of the necessary **dataset** must not exceed 1 gigabyte.
- Special **hardware** or **software** must not be needed.

The second phase included double checking bug fixes. On the one hand, they had to be marked as true positive, but it was also necessary that at least one took place after the first publication of the paper.

3.3 Explanation of Thesis Specific Implementation

After iterating through the JSON file from Papers With Code and the results from the SoftwareKG SPARQL query, the next step was to retrieve the commit messages from the sample S_i. It is important to mention that some repositories can be found on *GitLab*, while the majority reside on *GitHub*. Since the API endpoints are different, two separate implementations were needed to cover both variants.[6]

[6]The code has been written in Python (version 3)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

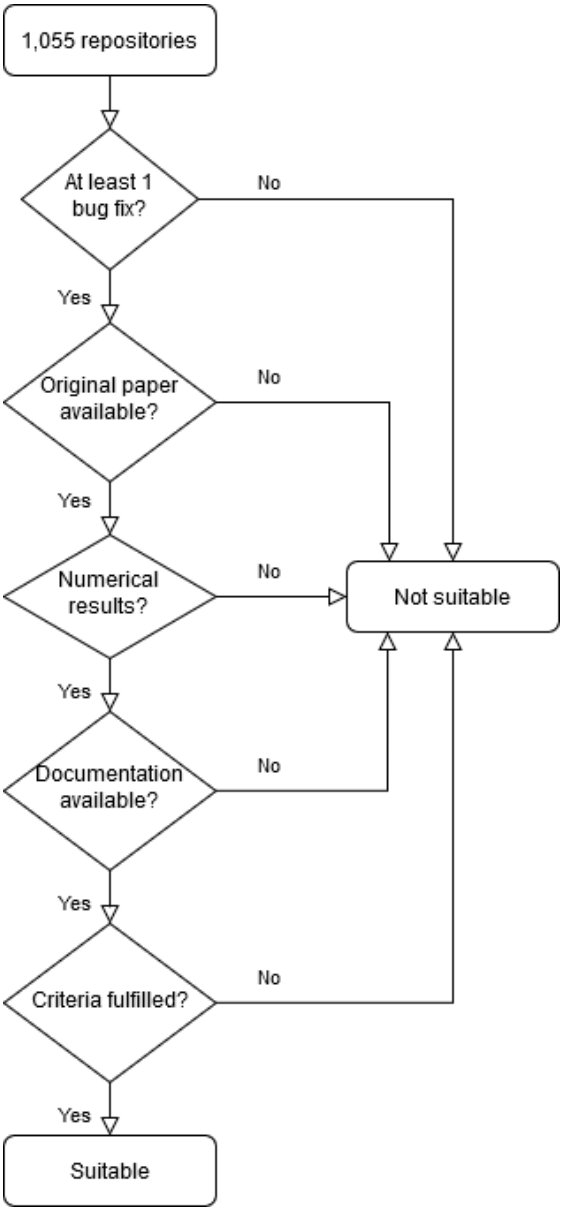


Figure 3.1: Evaluation of suitability to reproduce results

Due to the low rate limits for unauthenticated API requests on both platforms, [7,8] the usage of authentication was required. In both cases, it has been achieved by using access tokens.

[7] GitHub allows 60 unauthenticated and 5000 authenticated requests per hour [Git24a]
[8] GitLab allows 400 unauthenticated and 2000 authenticated requests within 10 minutes [Git24b]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
# using paperswithcode.json
for loop:
    if arxiv_id is not None:
        if get_citation_count(arxiv_id) >= 20:
            # stored "username/repo name" in a list
            if is_not_duplication:
                if is_not_single_file_implementation:
                    # perform classification


get_citation_count(arxiv_id):
    # GET request to
    # https://api.datacite.org/dois
    # /10.48550/arXiv.{arxiv_id}

    return response.json()['data']
        ['attributes']['citationCount']
```

Listing 5: Pseudo code for preparation of Papers With Code dataset

3.3.1 Iteration and Filtering of Datasets

I iterated the datasets without making any changes or applying filters beforehand. The number of citations for S_{skg} was retrieved manually, as the dataset did not contain an identifier like the arXiv ID. For the entries of S_i, it was possible to get the information in an automated manner. It has been achieved by calling the endpoint of https://api.datacite.org with a GET request using the arXiv ID. The exact URL can be found in Listing 5. The citation count was then retrieved from the JSON response by going through the elements of *data*, *attributes*, and finally *citationCount*. As mentioned above, a threshold of 20 citations per paper was necessary for it to be considered further. Together with the other filters mentioned in Section 3.1.2, the process reached the target amount of 1,000 for the sample. Since the dataset of Papers With Code contained duplicates in terms of links to code repositories, I stored every repository in a list and checked for each if it is part of it already. Furthermore, the last check was then for the type of implementation. As mentioned above, in case the implementation was done within a single file, which is located in a repository with other implementations, it has not been considered for further usage. If every condition was satisfied, the classification of commit messages has been performed, which will be explained in the next section.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
nlp = sp.load("en_core_web_sm")

matcher = Matcher(nlp.vocab)
pattern = [[{'LOWER': {'IN': ['defect', 'error']}}],
           [{'LOWER': 'bug'}, {'LEMMA': 'fix'}],
           [{'LEMMA': 'fix'}, {'LOWER': 'bug'}],
           [{'TEXT': {'IN': ['fixes']}}]]

anti_pattern =
    [[{'LEMMA': 'fix'}, {'LOWER':
        {'NOT_IN': ['readme', 'grammar', 'test']}}],
    [{'LOWER': {'NOT_IN': ['readme',
        'grammar', 'test']}}, {'LEMMA': 'fix'}]]

matcher.add("Pattern", pattern)
matcher.add("AntiPattern", anti_pattern)
```

Listing 6: First version of spaCy patterns

3.3.2 Classification of Commit Messages

SpaCy [9] is a Python Natural Language Processing framework. It was used for the classification of commit messages. SpaCy supports more than 75 languages and offers "components for named entity recognition, [...] sentence segmentation, text classification, lemmatization [...] and more." In this work, rule-based matching has been used to classify commit messages from code repositories. Token matching is one of the capabilities of spaCy. Not only can the framework handle patterns, but it can also handle anti-patterns. Despite spaCy offering phrase matching, it is also possible to define tokens – a single word – and the conjunction of such. The patterns and anti-patterns have been implemented as follows:

First, the English language model was set. After that, four patterns have been defined in the following order:

- Exact matching of the words "defect" and "error"
- Exact matching of the word "bug" followed by the word "fix" covering variations like "fixes", "fixed", and "fixing"
- Same rule as the one before with reverse order

[9] https://spacy.io/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
nlp = sp.load("en_core_web_sm")

matcher = Matcher(nlp.vocab)
pattern = [[{'LOWER': {'IN': ['defect', 'error',
             'bugfix']}}],
           [{'LOWER': 'bug'}, {'LEMMA': 'fix'}],
           [{'LEMMA': 'fix'}, {'LOWER': 'bug'}],
           [{'TEXT': {'IN': ['fixes', 'Fixes']}}]]

anti_pattern =
    [[{'LEMMA': {'IN': ['fix', 'Fix']}}, {'LOWER':
        {'NOT_IN': ['readme', 'readme.md', 'grammar',
        'test', 'compile']}}],
    [{'LOWER': {'NOT_IN': ['readme', 'readme.md',
        'grammar', 'test', 'compile']}},
        {'LEMMA': {'IN': ['fix', 'Fix']}}]]

matcher.add("Pattern", pattern)
matcher.add("AntiPattern", anti_pattern)
```

Listing 7: Second version of spaCy patterns

- Single-word matching for “fixes”. This has not been covered in rules 2 and 3, because the word “bug” is obligatory. The exact matching in the first bullet point can consist of additional words, which are not allowed in this case.

The anti-patterns are supposed to rule out certain phrases. Using just the lemma “fix” can result in including phrases like “fix readme”, which had to be excluded as it does not contain a fix for functional bugs. The attribute *NOT_IN* can be used to restrict the matching for the provided list of words. The anti-pattern consists of two rules:

- Covering variations of the word “fix” like “fixes”, ”fixed”, and “fixing” followed by the words “readme”, ”grammar”, or “test”
- Same rule as the first one with reverse order

After the first full verification, which will be explained in Section 4.1, I have made some adjustments to the patterns and anti-patterns.

The changes were as follows:

- Added “bugfix” to the exact matching of words

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- Added “Fixes” to the single word matching
- Added “Fix” to the anti-pattern lemma, as well as “readme.md” and “compile”

After grouping the reasons for false positives during the verification of the automatic classification, it showed that the largest one was the fixing of typos. The anti-pattern explicitly avoids matches of commit messages like “fix readme.md”. Some of the matches still consisted of commits containing only the “readme.md” file. Because of that, I have implemented a check that counts the number of files in a match, and in case only “readme.md” has changed, the match is not considered as such.

3.3.3 Gathering of Data for Plots

During the preparation of data for the automatic classification of commit messages, the number of commits per repository has been counted. There is no explicit API endpoint that returns the exact number, which is why this solution was necessary. Additionally, the number of contributors was needed to answer the second research question. Both GitHub and GitLab offer explicit endpoints in their APIs that return a list of contributors:

- GitHub: https://api.github.com/repos/<user>/<repo>/contributors
- GitLab: https://gitlab.com/api/v4/projects/<user>/<repo>/members

<user> is the username, and *<repo>* is the name of the code repository. Since contributors are called differently, a separate logic depending on the type of repository was necessary. While GitHub calls them contributors, GitLab is using the term members, as can be seen in the provided URLs above.

3.4 Summary

In this chapter, I discussed how the data for the empirical study has been prepared. One of the two datasets came from the SoftwareKG and consists of nearly four million triples. The dataset provided 52 software repositories. S_{skg} has been further decreased due to being outdated or not providing commits in a necessary format. In the end, 33 software repositories were left.

In addition, the data from Papers With Code was used. It provides “links between papers and code” as a JSON file. In total, it consists of nearly 235,000 papers. Because the file contains duplicates, the URL of the code repository has been used as a unique identifier.

The target sample size of 1,000 has been filled with repositories from Papers With Code. In order to consider a paper from that dataset, a threshold of 20 for the number of citations has been defined. The last step for RQ2 was to gather commit messages via API calls to GitHub and GitLab. The strings were then classified with spaCy patterns and anti-patterns. Primarily, the words “bug”, “fix”, ”defect”, and “error” and thereby

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

variations were the main focus. While the number of commits has been counted during the gathering of commit messages, additional API calls were necessary to get the number of contributors. The first approach to RQ3 was to go through the list of repositories manually and try to rebuild the software with two different versions with bug fixes in between. It was expected to receive differing results. As none of the repositories fit the defined criteria, which will be discussed in the next chapter, 4 *Evaluation*, a second approach and third approach were defined. The second approach was considering the full dataset of Papers With Code and comparing the exact publication date with the dates of the bug fixes. In the event that at least one bug fix occurred after publication, the application has been included in the result set. Since this approach resulted only in a few repositories for which the criteria did not fit, the initial approach has been reused and changed. I introduced a workflow that checked the suitability of results to be reproduced, which included the reworked criteria of the first approach.

What follows is an evaluation of the gathered data, considering the research questions. The prepared dataset will be used to investigate the relationship between the size of code repositories and bug fix percentages.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Evaluation

This chapter describes the evaluation of the research questions, including intermediary steps. As a data source, the files from Papers With Code[1] and the Software Knowledge Graph have been used. The manual verification process validated the automated classification of commit messages. What follows are the sections "Classification of Commit Messages with spaCy" (related to RQ1), "Correlation of Bug Fix Percentages and the Size of Code Repositories" (related to RQ2), and "Effects of Functional Bug Fixes" (related to RQ3).

4.1 Classification of Commit Messages with spaCy

The purpose of the verification is to calculate the ratio of falsely positive and falsely negative classified commit messages. From the full sample, a block of 50 repositories has been collected and verified manually. The list of repositories can be found in the Appendix. It contains a link to the repository, the total number of commits, the number of bug fixes found by spaCy, and the actual number of bug fixes. The manual verification process contains the following parts:

- **Counting total number of bug fixes**
- **Counting false positives:** Commit messages that have been marked as bug fixes by spaCy, but are *not* considered as such.
- **Counting false negatives:** Commit messages that have *not* been marked as bug fixes by spaCy, but were expected to be

[1] https://paperswithcode.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Category	Description	Amount
Duplication	Bug fixes have been considered twice when a merge pull request contained the same commit message that has been classified already	13
Typo	Fixes of typos in comments or changes in the README.md file that do not have an effect on calculations	5
Syntax	Fixes of syntax errors that cause the compilation to fail and prohibit an execution of the code	1
Docker	Changes to the Dockerfile that are affecting the metadata (label) of the Docker image but do not have an effect on calculations	1
Git	Changes to the *.gitignore* file that do not have an effect on calculations	1

Table 4.1: Categories of falsely positive classified commit messages

Precision and recall are critical metrics for evaluating the effectiveness of classification models. Precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.1}$$

This metric represents the ratio of correctly predicted positive observations to the total predicted positives.

Recall, on the other hand, is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.2}$$

Recall measures the ratio of correctly predicted positive observations to all observations in the actual positive class.

The results of the verification showed a precision of 93.75% and a recall of 95.17%. Within the mentioned 50 repositories for manual verification, there were 21 false positives (out of 336 total bug fixes, according to spaCy). Table 4.1 shows the reasons in categories why those messages should not have been considered a bug fix. It displays that most of the false positives were either a duplication (13) or a fix of a typo (5), followed by syntax fixes (1) and miscellaneous reasons (2).

On the one hand, it is difficult to classify commit messages that contain a fix for a typo, as it can indeed be a fix for an error that occurs during runtime. For example, loading a file using a string to locate it does not break compilation but will only show

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

up when the file is being opened and used. Having a typo in the string of the file name causes a *FileNotFoundError*. On the other hand, fixing a typo in a comment or in the documentation in general, like the README.md file, does not have an effect on the code or any kind of calculation. Differentiating those two types of typos just based on the commit message was not possible in cases where the developer used a message like "*fix typo*" as it covered both cases.

The issue with commits being counted twice occurred because merge pull requests have been created for fixes and contained the same message as the initial bug fix. Since the full commit message is being considered, both of them are triggering a match. To provide a clear picture of this, I selected one concrete example from the GitHub user *ayoungkang* and the underlying repository pytorch_Yolov4_lane-detection [2].

- Commit 1[3]: "Merge pull request #192 from YongWookHa/master fix: deprecated 'size_average' "
- Commit 2[4]: "fix: deprecated 'size_average' "

The first commit ("Commit 1") contains the merge pull request string and the initial commit message. The second one ("Commit 2") contains just the initial message. The so-called "anti pattern" in the matching rules (see 7) for spaCy triggered the match due to the token *[{'LEMMA': 'fix'}]* in the first anti pattern rule. The rule is triggering a match for every string that contains "fix" and **does not** contain the words "readme", "grammar" and so on. The difficulty here is that merge pull requests cannot be excluded in the first place since there are also repositories that do not include the initial commit (in the example above, "Commit 2"). Ignoring the merge request message would then produce a false negative. To avoid duplication, additional logic outside of the spaCy matching would be necessary.

4.1.1 Limitations of spaCy

As I have mentioned before, spaCy is a NLP framework. When it comes to the processing of text, it offers a wide variety of capabilities. On the other hand, going beyond dealing with strings is not possible. The classification of commit messages, as it was the case within this work, required additional logic, which had to be implemented separately. The category "typo" in Table 4.1 contains cases in which the fixing of typos was recognized as a bug fix. Excluding the commits that only consisted of a typo fix in the readme can reduce the number of false positives. This has been achieved by examining the number of files as well as verifying the file type. When the changes contained only a

[2]https://github.com/ayoungkang/pytorch_Yolov4_lane-detection
[3]https://github.com/ayoungkang/pytorch_Yolov4_lane-detection/commit/a132f275d62b52d3e829720996f11c03ed5f4256
[4]https://github.com/ayoungkang/pytorch_Yolov4_lane-detection/commit/ac43ec2a8c8846691b46b435ce2ac431a89a1fdf

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

single "readme.md" file, the match by spaCy was ignored. This improved the precision from 88.99% (37 false positives) to 93.75% (21 false positives).

In some cases, the specified patterns did not work as expected and caused inconsistencies. The reworked anti-patterns, as shown in 7, included a check for the word "compile" and variations. This excluded some of the previous matches, but not all of them. One commit that used to be part of the result set contained the message "*Another compile fix*" [5]. It has been excluded successfully. On the other hand, the commits "*Compile fix on Windows*" [6] and "*Another compile fix)*" [7] (sic) were still considered a match after the change. The latter shows that the capitalization was not the cause of the false positives. Due to the limited time available, this issue has not been further investigated.

4.2 Correlation of Bug Fix Percentages and the Size of Code Repositories

The classification and results from spaCy have been used to create the charts that will be discussed in this section. Within S_i, 662 (62.75%) had only a single contributor, and 346 (32.8%) had at least two. For 47 repositories, the API either returned an error or zero. Only repositories with at least one contributor have been included in Figure 4.1. The two separate figures show the distribution of bug fix percentages across code repositories and have exactly one contributor on the left-hand side and more than one on the right-hand side. As I have mentioned before, due to the high number of repositories with exactly zero bug fixes, a separate bar has been added. In the first version of the bar charts (see Figure 4.1), the lowest/leftmost bar showed the percentage from 0% to 9%. Since repositories with a single contributor had a significant number of cases with no bug fixes (0%) a separate bar for those cases has been added (see chart 4.1a).

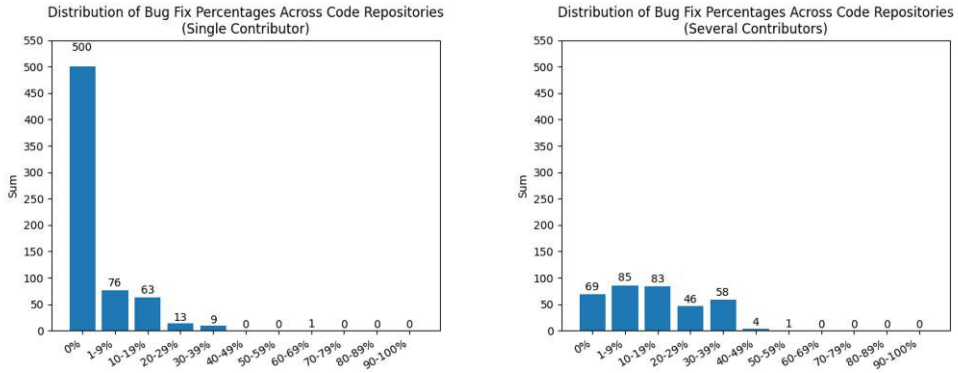
Within 662 repositories of single contributors, 500, which amounts to 75.53%, did not have a single bug fix. 76 had 1-9%, 63 had 10-19%, 13 had 20-29%, 9 had 30-39%, and the commits of 1 repository consisted of 60-69% of bug fixes. For repositories with more than one contributor, the range of bug fix percentages was similar, but the distribution was more even. 69 had zero bug fixes, 85 had between 1 and 9%, 83 had between 10 and 19%, 46 had between 20 and 29%, 58 had between 30 and 39%, 4 had between 40 and 49%, and 1 repository had between 50 and 59% bug fixes within commits.

In comparison, repositories with a single contributor tend to have no bug fixes at all. Repositories with at least two contributors tend to have a higher percentage in the range of 20% to 39%. I believe that this occurs due to the higher complexity of those projects. This will be further discussed by showing figures comparing the size of repositories, considering the number of contributors and commits, to bug fix percentages. I would like to point out that repositories of single developers that do not contain a bug fix have a

[5] https://github.com/peter8301120/darknet_t/commit/2790464de1185869dbc6ff565294a4cd34e8db71
[6] https://github.com/peter8301120/darknet_t/commit/1b15e2f8df5794a8c81fdfafddb3863c9a1a9e16
[7] https://github.com/peter8301120/darknet_t/commit/596311e2c79dd9c75fa55278580c8d575779dbe6

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

lower average of commits compared to repositories containing bug fixes. This underscores the fact that repositories with a single contributor tend to lack bug fixes due to their size and consequently lower complexity. Further, code bases with bug fixes not only have a higher average but also have a wider range for the number of commits. Figure 4.2 exemplifies this statement.



(a) Single contributor (b) Several contributors

Figure 4.1: Comparison of bug fix percentages within commits between repositories having one contributor and several contributors

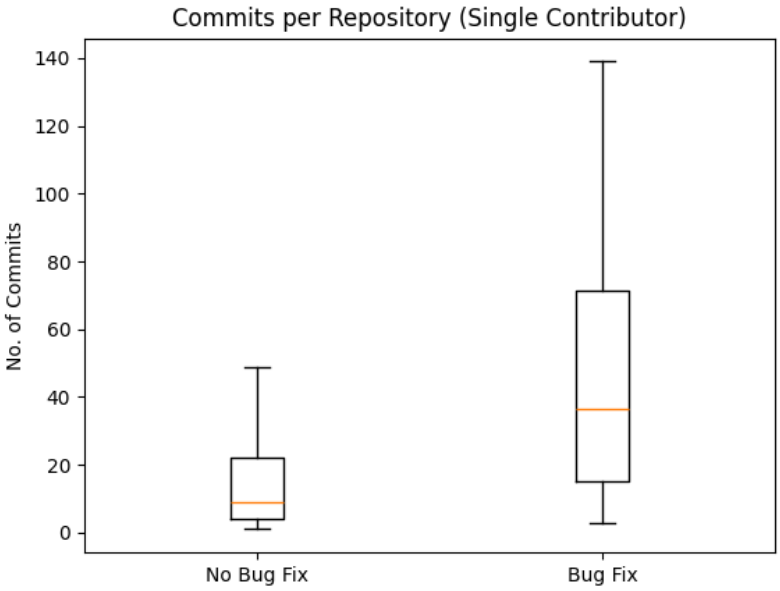


Figure 4.2: Commits per repository having exactly one contributor split into repositories with no bug fix and repositories with bug fixes

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Larger and thereby more complex projects can be defined by the number of commits and contributors. As I have just mentioned, complex applications tend to have a higher percentage of bug fixes. Figure 4.3 shows that there is a strong positive correlation between the number of contributors and the number of commits per repository. This can be explained by the fact that complex software requires more developers. A higher number of contributors then leads to a higher number of commits.

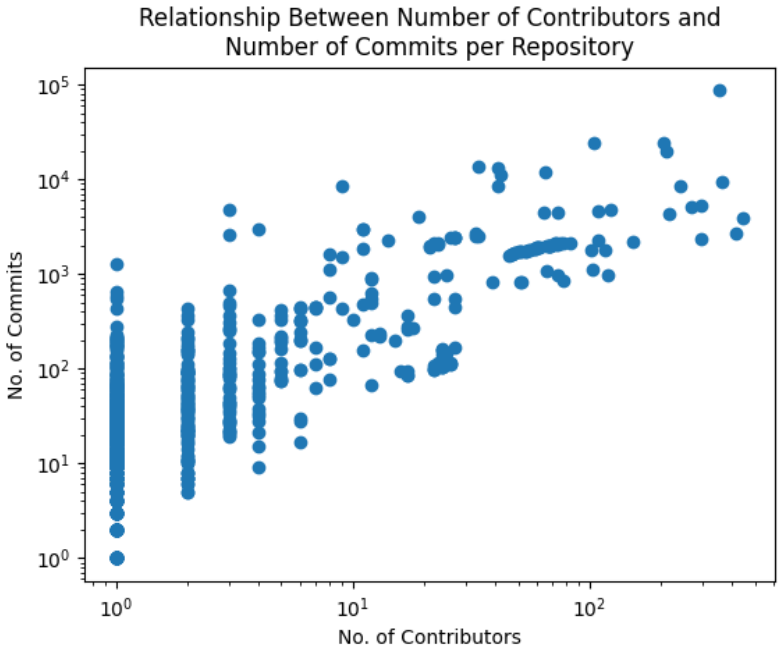


Figure 4.3: Relationship between the number of contributors and the number of commits

A similar tendency can be seen when comparing bug fix percentages with the number of contributors. While mid-sized repositories between 10 and 100 contributors have a ratio below 40%, projects with more than 100 contributors always have at least 15%. While projects with less than 100 contributors tend to have a lower number of commits, the bug fix percentage also resides below 40%. It is easier for a repository with 10 commits to reach 50%, than larger ones with several thousand commits. The relationship between the number of contributors and the bug fix ratio shows a weak positive correlation (see Figure 4.4).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

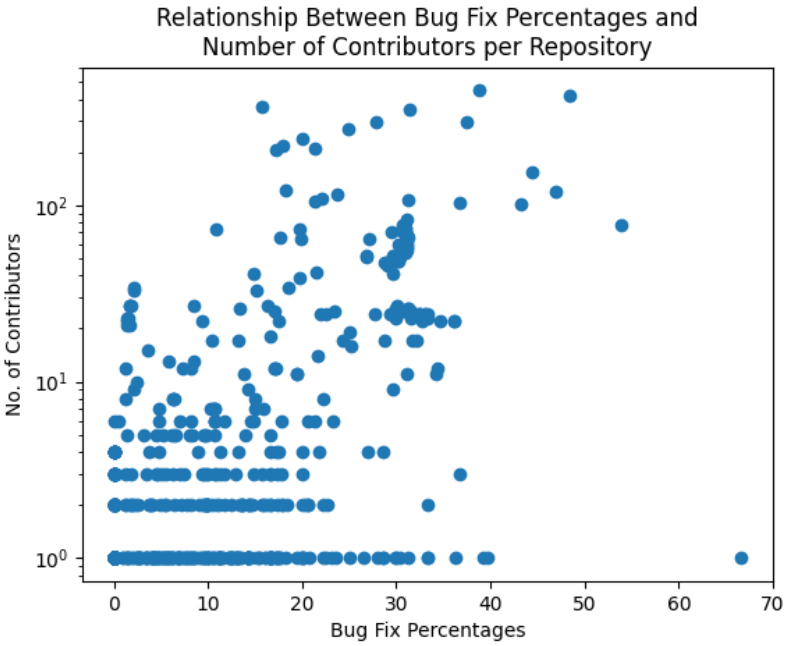


Figure 4.4: Bug fix percentages compared to number of contributors per repository

A weak positive correlation shows Figure 4.5 that presents bug fix percentages on the x-axis and the number of commits on the y-axis. Again, a significant number of repositories do not have any bug fixes. Most of the repositories gather in the area of 0% to 25% while having 10 to 500 commits. On the other hand, applications with more than 1,000 commits tend to have a higher percentage of bug fixes in the range of 10 to 50%.

The two curves in the lower part of the plot are related repositories. Some projects can be found several times in the sampled dataset of the plot, like “bert”, ”darknet”, or “yolo”. My first thought was that this was caused by reusing code and forking other projects. When doing so, GitHub offers the option to “copy the master branch only”. Initially, I was assuming that the commits of bug fixes were not part of the new repository since bug fix percentages were decreasing. While having a closer look, I was focusing on the isolated dot having 20% bug fixes and 5 commits marked with a red rectangle in Figure 4.5. Searching in the dataset provided three repositories:

- mjhucla/mRNN-CR
- mindspore-courses/DeepNLP-models-MindSpore
- tmbaoloc/darknet

This aligns with the initial assumption of reoccurring project names – in this case, “darknet”. The following repositories are further examples:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- peter8301120/darknet_t
- tanmayj000/darknet
- CRIGIM/darknet
- hirohiro23/Darknet
- xuanloct4/darknet
- AmeerHamza111/yolo-darknet
- SAIGANESH-Collab/DARKNET

While dealing with RQ3, I had to go through code repositories in order to find out if those fit the defined criteria. During this process, it showed that certain projects have indeed been forked within GitHub. During the first approach to answering RQ3, the result set contained several "bert" repositories, all containing the same initial commits. As I have mentioned, a copy of the master/main branch has been created in those cases. Upon that, some adjustments have been made to the base code. While the number of commits increased, there were no additional bug fixes, which led to a decrease in bug fix percentages, resulting in the curves in Figure 4.5.

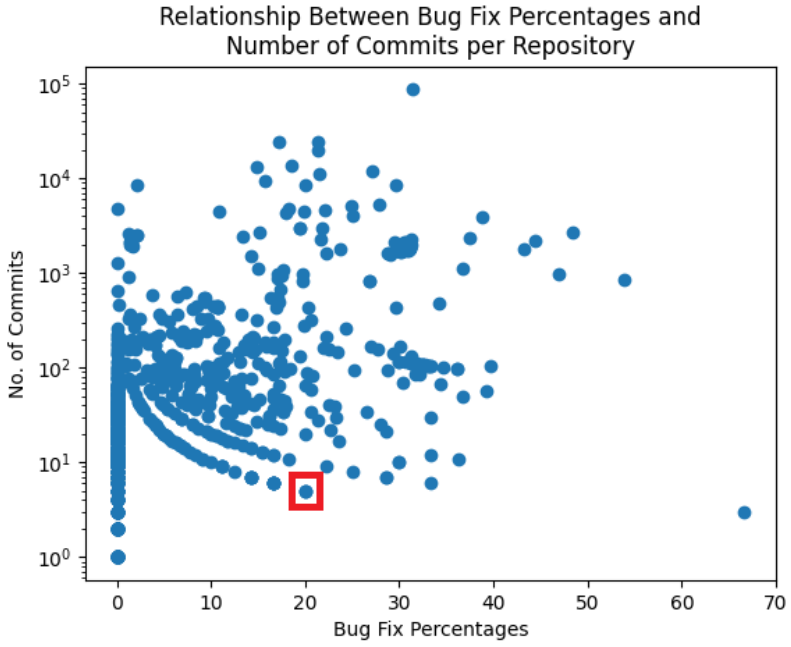


Figure 4.5: Bug fix percentages compared to number of commits per repository

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.2.1 Difficulties of Selected Datasets and APIs

While the documentation of datacite described the published field as "the issued date or publication year" [Dat24], it did not provide the full date. Using the field *created* instead is not an alternative, since it shows the datacite-internal creation of a "DOI record in iso8601" [Dat24]. Comparing it to other websites (e.g., https://arxiv.org/) shows a significant difference. In addition, when a paper has several versions, the publication year from datacite seems to show the year of the first publication. For example:

- The paper *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [DCLT18] had two versions published. The publications took place on November 10, 2018 and May 24, 2019.
- Datacite returns 2018 for the *published* field. This is also the case for the automatically generated BibTex reference.

The dataset of Papers With Code also includes links to code repositories that are not official implementations. This can be detected by the *is_official* flag presented in Listing 4. When this flag is set to false, it means that methods or contents of the linked paper have been used in the code. During the work of this thesis, the unofficial implementations have mostly been discovered as forked repositories and thus have been excluded from the second approach of RQ3, as those are reused code. Besides that, not all the official implementations in the dataset were exactly that. As discussed in Section 4.3, the repository "DeepNLP-models-MindSpore" of the account "mindspore-courses" linked papers in the readme file. Some models and methods from those papers have been used in the code. Because of that, the repository has been marked as an official implementation of the paper "Efficient Estimation of Word Representations in Vector Space" [MCCD13].

4.3 Effects of Functional Bug Fixes

As discussed in Section 3.2 *Study Design of Thesis*, I have defined criteria for code repositories to be considered or excluded in order to answer RQ3. Due to the difficulty in comparing graphical and sound-related results of papers and finding differences, those have been excluded during the first approach. Later on, those results were not explicitly excluded anymore due to the limiting effect on the results of this work. The dataset of Papers With Code mostly contained projects that dealt with image recognition. Besides that, within S_i, 613 (58.1%) did not contain a bug fix. This reduced the candidates for RQ3 significantly. After consideration of all the other criteria during the first approach, no code repositories were left to be evaluated. Because of that, it was not possible to investigate the effects of bug fixes directly.

This made it necessary to introduce another approach for RQ3. While I initially only worked with a subset of the Papers With Code data, the second approach considered the full dataset of 235,000 entries. Six papers had bug fixes after publication. What follows are descriptions of those projects:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

AlexeyAB/darknet [8] The repository "darknet" of user "AlexeyAB" is the base repository of several forks within the Papers With Code dataset. It has been forked itself from the repository https://github.com/pjreddie/darknet, in which the user "AlexeyAB" was a contributor. "Darknet is a tool [...] which may be used for computer vision. It is used to both train neural networks, and then run images or video frames through those neural networks. YOLO is one of many possible configurations files that Darknet uses." [Boc24]

The paper was published on April 23, 2020. Despite the publication being four years ago, the code still receives updates and fixes. Since April 23, 2020, there have been 157 bug fixes in total. The latest ones occurred at the end of August/early September 2023 and on May 3, 2024.

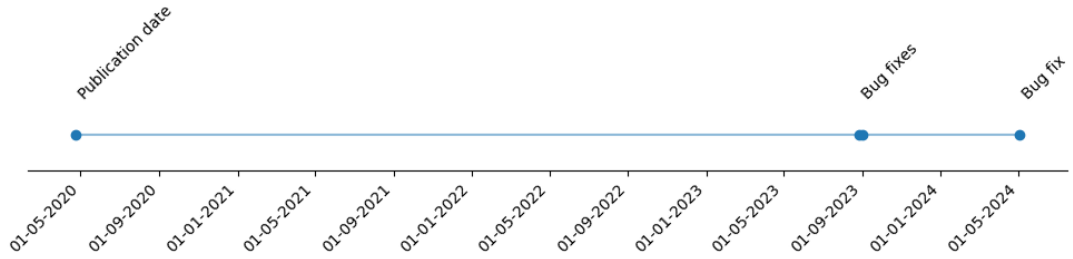


Figure 4.6: Timeline of paper [BWL20] and repository "darknet" of user "AlexeyAB"

lmcinnes/umap [9] UMAP stands for Uniform Manifold Approximation and Projection, which "is a dimension reduction technique that can be used for visualization similarly to t-SNE but also for general non-linear dimension reduction." An external documentation [10] outside of GitHub exists for the project. Even though it also contains a description on how to reproduce results in general [11], there is no description on how to reproduce the results of UMAP itself. The paper was published on February 9, 2018.

[8] https://github.com/AlexeyAB/darknet
[9] https://github.com/lmcinnes/umap
[10] https://umap-learn.readthedocs.io/en/latest/
[11] https://umap-learn.readthedocs.io/en/latest/reproducibility.html

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

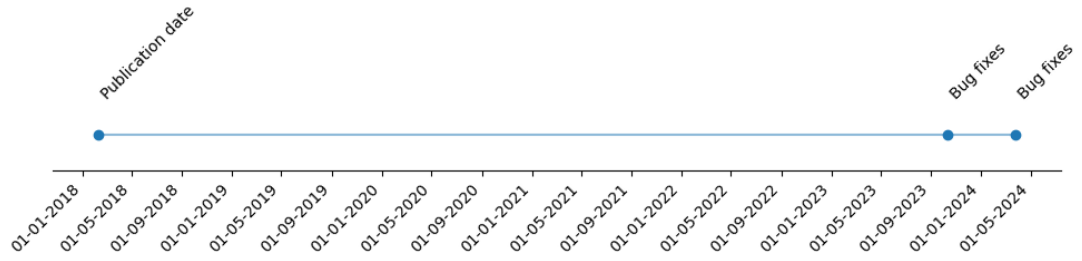


Figure 4.7: Timeline of paper [MHSG18] and repository "umap" of user "lmcinnes"

lh3/bwa [12] This repository "is a software package for mapping DNA sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW, and BWA-MEM". The first commit was created on January 14, 2011. While the paper was published on May 16, 2013, the software is still receiving changes in 2024. In total, my implementation found 76 bug fixes. The timeline (see Figure 4.8) consists of bug fixes around March 23, 2020, and July 30, 2017, as well as one bug fix on April 07, 2018. The project was implemented by using C as the programming language.

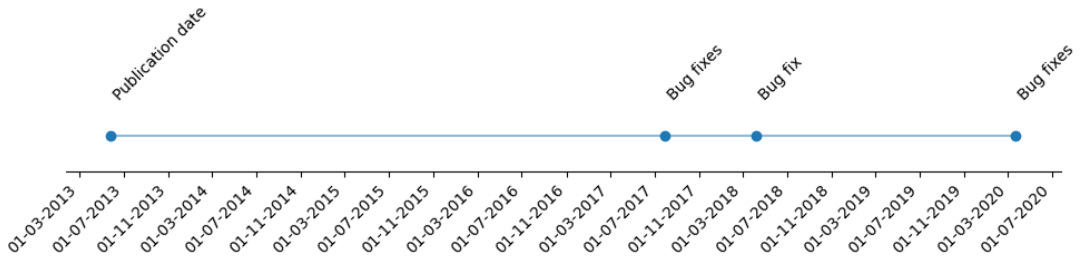


Figure 4.8: Timeline of paper [Li13] and repository "bwa" of user "lh3"

mindspore-courses/DeepNLP-models-MindSpore [13] The GitHub repository is not the official implementation of the paper. This project has been implemented to provide examples for "MindSpore implementations of various Deep NLP models". It was part of a lecture at Stanford University. The paper was linked in the readme file for the first model example, "Skip-gram-Naive-Softmax".

jlmelville/uwot [14] This repository is an R implementation of the paper [MHSG18]. It had 20 bug fixes after the publication on September 18, 2020. Since I have created a timeline for the Python implementation above, I did not provide another visual presentation for it.

[12] https://github.com/lh3/bwa
[13] https://github.com/mindspore-courses/DeepNLP-models-MindSpore
[14] https://github.com/jlmelville/uwot

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

ekg/freebayes [15] Despite the paper having been published on July 20, 2012, the code repository still receives updates. The last commit was pushed on April 28, 2024 (as of July 11, 2024). Because of this, there were also recent bug fixes that took place on October 12, 2022, February 13, 2023, and April 23, 2024. In total, since July 17, 2012, 92 bug fixes have taken place. The programming language C++ was used for the implementation.

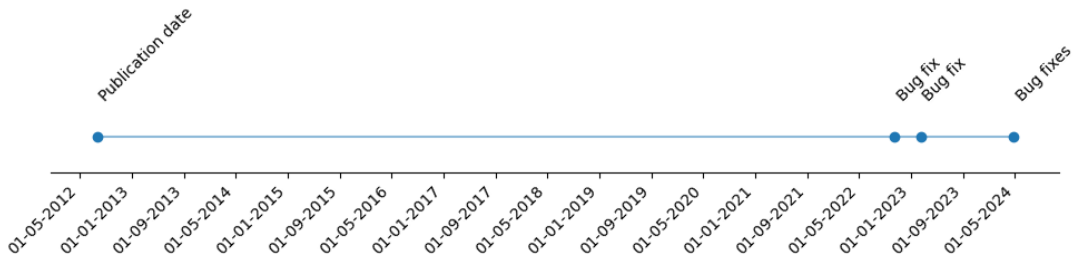


Figure 4.9: Timeline of paper [GM12] and repository "freebayes" of user "ekg"

In addition, two code repositories were part of the result set when the year of publication – adding January 01 – was used for the comparison. After it was replaced by the exact publication date, they did not have any relevant bug fix dates anymore.

- tensorflow/tensor2tensor[16]
- google-research/bert[17]

4.3.1 Revisiting First Approach

Like in the first approach to RQ3, the resulting dataset of RQ1 and RQ2 (S_i) was used. I created a workflow to evaluate the suitability of every repository as explained in Section 3.2 *Study Design and Implementation* and displayed in Figure 3.1. At first, I removed all the repositories that did not contain any bug fix, which amounts to 613. This resulted in 442 repositories that contained at least one bug fix. Opposing the first approach to RQ3, I did not remove graphical and sound-related results, but still considered those suitable if the paper contained any kind of numerical results. Every time, a paper has been considered as not suitable the reason was documented. In 343 cases, neither the repository nor the datasets of Papers With Code and SoftwareKG contained a URL to the original paper for which the code has been implemented. As mentioned above, common cases were the papers and repositories of *Attention Is All You Need* [VSP+17] and *Google BERT* [DCLT18]. The repositories were often forked and either contained

[15] https://github.com/ekg/freebayes
[16] https://github.com/tensorflow/tensor2tensor
[17] https://github.com/google-research/bert

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Reason	Description	Amount
No original paper	Neither the datasets of Papers With Code and SoftwareKG, nor the code repository contained a URL to the paper for which the code has been implemented	343
No numerical results	The paper for which the code has been implemented did not contain any numerical results	39
No documentation	No documentation has been provided	14
No bug fixes	The marked bug fixes from RQ1 were either false positives or took place before the publication of the paper	9
Duplication	The repository has been forked by one of the researchers or the software has been implemented for another paper and was reused	3
Runtime	The runtime criteria was not fulfilled	2
Hardware required	Special hardware was required in order to reproduce the results	2
Software required	Special software (e.g., operating system) was required in order to reproduce the results	1

Table 4.2: Reasons for excluding repositories in the first phase of answering RQ3

the same amount of commits or had a few additional commits. 39 papers did not contain any numerical results. There was no documentation available within 14 repositories on how to execute the software. 9 repositories did not have a relevant bug fix, since they were either false positive or no bug fix took place after the initial publication of the paper. I marked three repositories as duplication as they have been forked by one of the participating researchers or the repository was copied and reused. In the latter case, the repository referred to its own and original paper, but linked the tutorial on how to execute the software of another repository that was also part of S_i. The remaining reasons were that the runtime criteria was not fulfilled (2), the execution required certain hardware (2), and that the execution of the software required a specific operating system (1). An overview of the reasons has been provided in Table 4.2.

The second phase included the remaining 29 repositories. Table 4.3 lists the reasons for exclusion. 11 repositories provided a general documentation, but did not explain how the results can be reproduced. Due to my lack of domain knowledge it was not possible to find the correct parameters or arguments for the applications in order to reproduce the numerical results. The installation of the software or the dependencies failed for 7 repositories. In four cases the execution of the software failed and two times the data was not available at all or the data criteria was not fulfilled.

The remaining five applications, which will be listed shortly, have been executed successfully. While it was possible to reproduce the results of *facebookresearch/anli*, the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Reason	Description	Amount
Incomplete documentation	The repository contained a general description on how to execute the software, but there was no description on how to reproduce the results	11
Installation failed	The installation of the software or dependencies failed	7
Execution failed	The execution of the software failed	4
Data	The required data was not available or the data criteria was not fulfilled	2

Table 4.3: Reasons for excluding repositories in the second phase of answering RQ3

execution of *WongKinYiu/yolov7* (for *pjreddie/darknet*) calculated different results.

pjreddie/darknet [18] As mentioned above in the second approach, this is the original implementation of the YOLO project. The lastest changes included the results and documentation for version 7. The related paper is called *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors* [WBL23]. According to the readme file, the source code of the paper can be found in a repository of the user *WongKinYiu* [19]. It is also supposed to be used when one is trying to reproduce the results. The documentation describes the process on how to reproduce the average precision and recall when testing the model. According to the readme file, the exact numbers for average precision are supposed to appear when executing the file test.py with certain parameters listed under *Testing*. The expected results, consisting of 12 lines, did not appear. After executing the command, I received only a single line displaying the results for the class *all*. In the documentation, it was called *area all* as pictured in Figure 4.10b. In addition, the results that I received only showed the values for *Average Precision IoU=0.50:0.95 area= all* and *Average Precision IoU=0.50 area= all*, which are the first two lines in Figure 4.10b. I received 69.1% for the *Average Precision IoU=0.50 area= all* (in contrast to 69.7%) and 49.7% for the *Average Precision IoU=0.50:0.95 area= all* (in contrast to 51.2%). The results that I received are displayed in Figure 4.11. The mentioned results were created with the latest available version of the code as of December 2024. It was not possible to recreate the results as of 6 July 2022, the date of the publication, as the first commit was created on the same day within the repository of the user *WongKinYiu*. The original repository of the user *pjreddie* does neither contain the file test.py, nor a description on how to execute the code.

[18] https://github.com/pjreddie/darknet
[19] https://github.com/WongKinYiu/yolov7

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Table 1: Comparison of baseline object detectors.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-31 [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

(a) Results within the paper

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.51206
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.69730
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.55521
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.35247
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.55937
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.66693
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.38453
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.63765
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.68772
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.53766
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.73549
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.83868
```

(b) Results within the repository

Figure 4.10: Comparison of results of [WBL23]

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	5000	36335	0.724	0.635	0.691	0.497

Figure 4.11: Reproduced results of [WBL23]

ruinmessi/RFBNet [20] I followed the described steps in the documentation of the code repository. This included the installation, the preparation of the datasets (both VOC and COCO), as well as the training. Unfortunately, the training was not finished within one hour. At that point, I canceled the execution due to the defined criteria in Section 3.2.3 *Revisiting First Approach*. When comparing the first version of the paper to the latest one – version 3 as of 12 January 2025 – there are differences in the results of the RFB Net300 method. The respective results can be found in Table 4 in version 1 and version 3 of the paper [LHW18]. Three bug fixes took place after the initial publication on 21 November 2017:

- 16 December 2017: *fix 512 prior error on* [21]
- 24 December 2017: *compatibility fix* [22]
- 5 August 2018: *fix the basic RFB structure in RFB_Net_vgg* [23]

facebookresearch/anli [24] In the Section *Pre-trained Models* of the readme file a code snippet can be found. It prints predicted probabilities for entailment, neutral and contradiction based on different models. As those values were not presented in the paper [NWD+20], I had to use another code snippet [25] that prints the accuracy of the model. After the execution of the code, I received an accuracy of 92.6% (see Figure 4.12) for the *RoBERTa* model. This aligns with the value that is available

[20] https://github.com/ruinmessi/RFBNet
[21] https://github.com/GOATmessi8/RFBNet/commit/a80171f635694f92dfe3ac428554b98ca43a9544
[22] https://github.com/GOATmessi8/RFBNet/commit/a1033fa63cfb84a97854955b27e4fbd23f2669af
[23] https://github.com/GOATmessi8/RFBNet/commit/47d7e971be5a9bf3815661692688c5f8a2410d29
[24] https://github.com/facebookresearch/anli
[25] https://github.com/facebookresearch/anli/blob/main/src/hg_api/interactive_eval.py

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

in the paper. In the first version of the paper the table can be found on page 5 (see Figure 4.13). The accuracy for the *RoBERTa* model stayed the same in the second version, that was published on 6 May 2020.

```
Total / Correct / Accuracy: 9842 / 9109 / 0.9255232676285308
```

Figure 4.12: Reproduced results of [NWD+20]

Model	Training Data	A1	A2	A3	ANLI	ANLI-E	SNLI	MNLI-m/-mm
BERT	S,M[⋆1]	00.0	28.9	28.8	19.8	19.9	91.3	86.7 / 86.4
	+A1	44.2	32.6	29.3	35.0	34.2	91.3	86.3 / 86.5
	+A1+A2	57.3	45.2	33.4	44.6	43.2	90.9	86.3 / 86.3
	+A1+A2+A3	57.2	49.0	46.1	50.5	46.3	90.9	85.6 / 85.4
	S,M,F,ANLI	57.4	48.3	43.5	49.3	44.2	90.4	86.0 / 85.8
XLNet	S,M,F,ANLI	67.6	50.7	48.3	55.1	52.0	91.8	89.6 / 89.4
RoBERTa	S,M	47.6	25.4	22.1	31.1	31.4	92.6	90.8 / 90.6
	+F	54.0	24.2	22.4	32.8	33.7	92.7	90.6 / 90.5
	+F+A1[⋆2]	68.7	19.3	22.0	35.8	36.8	92.8	90.9 / 90.7
	+F+A1+A2[⋆3]	71.2	44.3	20.4	43.7	41.4	92.9	91.0 / 90.7
	S,M,F,ANLI	73.8	48.9	44.4	53.7	49.7	92.6	91.0 / 90.6

Figure 4.13: Model performance of [NWD+20]

mcsorkun/ChemPlot [26] The repository contains a description for the installation, as well as how to use *ChemPlot*. Even though the installation and the execution were successful, the implementation did not print results as expected. The code should print a table containing reduced dimensions and target values, as well as three plots. All of those components do not appear during execution.

bpaassen/ast2vec [27] On the one hand, the readme file contains a quickstart guide and on the other hand it is mentioning a file called *tutorial.ipynb*, which is using a mock data set of programs. The tutorial was executed successfully and also provided results. Unfortunately, the results that I received within the tutorial are different to the ones that were available in the paper [PMJ+21] and therefore not comparable. Since the quickstart guide is not extensive, it was not possible for me to reproduce the results.

4.3.2 Difficulties in Reproducing Results

Due to the fact that some of the research projects were implemented several years ago, the code, the requirements, and the dependencies have not been updated since. For

[26] https://github.com/mcsorkun/ChemPlot
[27] https://gitlab.com/bpaassen/ast2vec

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

example, the last change within the repository https://github.com/mjhucla/TF-mRNN took place in 2016. The documentation states Python version 2.7 as a requirement. Even though I used the listed version, the installation of other dependencies failed. In other cases, the outdated versions were still available, but the execution failed due to deprecated and incompatible code. For example, the last change of the repository https://github.com/Songlielie/DKT-Plus took place in 2020 and states that the code was only tested with Tensorflow 0.10. Even though the installation succeeded, the execution failed because of the implementation as mentioned.

I tried to install the dependencies of 29 repositories during the second phase. While the installation was often simple and quick, since the dependencies were either listed in the documentation or part of the code, the clean-up of such was never mentioned. Most of the time, it was an advantage to keep the dependencies because they were overlapping for different software projects, and the installation process was sometimes tedious and time-consuming. On the other hand, for a few exceptions, the implementation required specific versions, which then had to be taken care of separately. From my experience, for a small portion of the repositories, it is an advantage to start with a clean environment.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Summary and Conclusion

Open-Source Software has become increasingly important for the public and private sectors and research due to its free availability of source code and code repositories, distinguishing it from free software, which does not always provide the source code. OSS benefits researchers through reduced costs, customizability, and collaborative opportunities, but includes challenges like compatibility and reliability issues, as ongoing maintenance can affect stability and research outcomes.

While there has been work on the reliability of software considering social aspects and code, there has been no empirical study focusing on bug fixes. Schueller et al. [SWS+22] [SW24] created a database that served as a starting point for their study, but their focus was on dependencies and single developers influencing interdependent projects.

What follows is the summary of the results for the research questions:

RQ1: To what extent can commit messages be categorized into functional bug fixes and feature changes automatically?
The datasets of the SoftwareKG and Papers With Code have been used to answer this question. Both contain papers and related software repositories in which the code and additional metadata like commits and the number of contributors are available. After the preparation of the datasets, 33 repositories were left to be used from SoftwareKG. In order to reach a sample size of 1,000, it was necessary to consider 125,000 entries from Papers With Code. The total sample size of repositories was 1,055.

Because the dataset of Papers With Code contained duplicates, I considered the link to the code repositories as a unique identifier. Even though some repositories have occurred several times, they do not interfere with the use-case of this thesis. It simply means that methods from different papers have been considered for the implementation of the research software. Using a unique identifier helped to make

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

sure that the statistics within RQ2 do not contain duplicates, which would cause inconsistencies.

Another criteria was that the *paper_arxiv_id* must not be null. This ID has been used to retrieve the citation count of each paper from the API of datacite. In order to consider a paper, the required number of citations has been set to 20. I used spaCy, a NLP framework, for the automatic classification of commit messages. It showed that it is possible to classify commit messages from code repositories automatically. The results of the verification showed a precision of 93.75% and a recall of 95.17%.

RQ2: To what degree do the size of the project/repository (i.e., number of commits, number of contributors) and the percentage of bug fixes correlate?
The sample from RQ1 (S_i) has been reused to answer this question. The number of contributors and the number of commits show a strong positive correlation. In addition, there is a weak positive correlation between the number of contributors and bug fix percentages. Code repositories with more than 1,000 commits tend to have a higher percentage of bug fixes in the range of 10 to 50%. In general, this means that the bigger an application is, the more bug fixes it will contain. This can also be seen when comparing projects with a single contributor and projects with more than one contributor. Out of 662 repositories, with one contributor, 500 (75.53%) did not have any bug fixes. The weak positive correlation between the number of commits and bug fix percentages also exists for repositories with a single contributor. The results showed that repositories of a single contributor with no bug fix have a lower number of commits compared to repositories consisting of bug fixes. On the other hand, repositories with several contributors have more commits and show a higher percentage of bug fixes. Especially the range from 20% to 39% shows more occurrences compared to repositories with a single contributor.

RQ3: To what extent do functional bugs, that have been fixed, affect results of OSRS?
The first approach used the dataset S_i. It consisted of defined criteria and excluded visual and sound-related results. As mentioned, a threshold of 20 citations was necessary in order for a paper to be considered. In combination, this approach was too restrictive and did not result in any outcome.

The second approach used the full dataset of Papers With Code. The results from SoftwareKG have not been included because there is no direct link between the code repository and the paper, as it was the case with Papers With Code. Again, a citation count of 20 was required in order for a paper to be considered.

At first, the *created* field from the datacite API has been chosen to retrieve the publication date. This was necessary because *publicationYear* and *published* only provided the year of publication but not the full date. Unfortunately, this was not the right choice as it differs from the actual publication date. While the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

field *publicationYear* provides only the year of publication, it has been used for the automatic run to gather potential candidates. In a separate step, the actual publication date has been set manually to remove bug fix dates that took place before.

The initial result consisted of eight entries. Six repositories had at least one bug fix after the paper was published. One repository has been marked as false-positive after the manual verification. The readme file contained a link to the paper, but other than that, the repository was not related to the paper. One paper had two implementations in different languages. The Python implementation has been evaluated in detail. Overall, it was not possible to evaluate the affects of the bug fixes.

The third approach reused some of the criteria defined in the first one and included a workflow that was split into two phases. While the first phase focused on the paper and the repository as a whole, the second phase was focusing on the bug fixes and execution of the code. Out of 1,055 repositories only 442 had at least one bug fix. In addition, 413 further repositories were removed because of reasons like *no original paper*, *no numerical results*, *no documentation*, and *no bug fixes*. The remaining 29 repositories were investigated in more detail. Besides fives repositories, all the others were excluded because of an *incomplete documentation*, the *installation failed*, the *the execution failed*, and *data* was missing.

Five applications were executed successfully. The execution of three repositories either did not result in anything or the results were not comparable. The recalculation of *facebookresearch/anli* brought the same results. The repository *WongKinYiu/yolov7*, which was supposed to be used for the reproduction of *pjreddie/darknet*, calculated different results than what was published within the paper. The *Average Precision IoU=0.50 area= all* was 69.1% instead of 69.7% and 49.7% for the *Average Precision IoU=0.50:0.95 area= all* instead of 51.2%.

To sum up, this thesis contained three research questions dealing with OSRS. First, I explored the extent to which functional bug fixes can be automatically categorized, utilizing datasets from SoftwareKG and Papers With Code, resulting in S_i. Automated classification of commit messages achieved a precision of 93.75% and a recall of 95.17% using the spaCy NLP framework. Second, the study examined the correlation between project size (number of commits and contributors) and the percentage of bug fixes, finding a strong positive correlation between the number of contributors and commits and a weak positive correlation between contributors and bug fix percentages. Projects with more than 1,000 commits and those with multiple contributors tend to have higher percentages of bug fixes. Finally, the work on the third question revealed five repositories that fulfilled all the defined criteria and conditions of which one showed different results that what was published in the paper. What follows is a discussion of limitations and potential future topics to work on.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.1 Limitations and Future Work

The content of this work is based on the datasets of Papers With Code[1] and SoftwareKG. While the dataset from Papers With Code contained nearly 235,000 papers and repositories, the relevant part from SoftwareKG provided only 52 repositories. It offers an opportunity to test the implementation, but it is not large enough to provide a sufficient sample for an empirical study. In addition, some repositories are outdated and do not provide the data – especially commits – in a format that is needed in order to use it. Concrete examples were mentioned in the subsection *SoftwareKG* of Section 3.1. While the dataset from Papers With Code provides a sufficient size and up-to-date repositories, including APIs, the linked software is not always the official implementation.

As mentioned in Section 4.1.1, everything that goes beyond NLP requires additional logic outside of the spaCy framework. The fixing of a typo has been considered a bug fix. This also contains cases in which just the "readme.md" file has been changed. This caused false positives, which have been excluded by implementing a check for the number of files in the commit as well as the file type. Besides this specific issue, I have also discovered general inconsistencies. Despite changing the pattern, the result still contained false positives (see 4.1.1 for an example). In addition, while testing my implementation, I have tried to use different language models besides the one that I used ("*en_core_web_sm*").

The limited amount of time that I had, as well as the criteria for repositories that are suitable to be rebuilt, reduced the potential candidates for RQ3 significantly. As mentioned, the dataset of Papers With Code contains a lot of repositories that deal with visual computing and voice/speech-related topics. Having the necessary domain knowledge and focusing on the area of those projects can help you work with the repositories. If that is not possible, using different or additional datasets can provide alternatives to reaching a certain sample size. It would be sufficient to redefine the criteria when finding and using software that can deal with those domains, e.g., comparing and interpreting graphical results.

During this thesis, I have discovered potential topics that can be based on this work:

Consider forked repositories when investigating effects of bug fixes The dataset of Papers With Code contains forked repositories that are related to each other. When forking a code repository, a copy of a certain branch is created. Changes to the base repository are not propagated to the other repository automatically. As the results of RQ3 showed, the repository "darknet" of user "AlexeyAB" had 157 bug fixes after the related paper was published. The changes can be relevant for the forked code.

Find a software to compare graphical results The defined criteria in the first approach to answering RQ3 excluded repositories that consisted of graphical results. Removing the specific criteria will increase the number of relevant code repositories

[1] https://paperswithcode.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

in the datasets of Papers With Code and SoftwareKG. This can be achieved by finding and using software that is capable of comparing differences in graphical results.

Improve spaCy patterns The classification of commit messages with spaCy showed a partial inconsistency. Despite the need for additional logic outside of it, the patterns can still be improved. A specific example was given in the Section 4.1.1.

Use machine learning Even though the precision and recall for the automatic classification of commit messages were both above 93%, the capabilities of spaCy are limited to NLP. The result would not have been achieved without additional logic outside the classification. Using machine learning instead of the spaCy framework might be a better fit in this case. It needs to be explored if it can consider changed files in addition to commit messages.

Consider additional metrics of code repositories The size of code repositories has been defined by considering the number of commits and the number of contributors. The API of GitHub provides other metrics like *stargazers_count*, *watchers_count*, *forks_count*, as well as *subscribers_count*. Since the field *size* returns the size of a repository in kilobytes, it might not be reliable due to large files like pictures; a better indicator could be the lines of code. This indicator needs to be retrieved separately from the GitHub API or by using a third-party application. Using additional indicators for the size of code repositories can emphasize the results of RQ2.

Notification for bug fixes An automatic notification for papers that are reusing or citing results can help to prevent subsequent errors. It needs to be elaborated on how this can be implemented, especially how papers citing results can be found and notified.

Use data of Schueller et al. [SWS+22] While I have been focusing on the datasets of SoftwareKG and Paper With Code, the database of Schueller et al. contains useful metadata and shows dependencies of code repositories. This fact can be helpful when investigating the effects of bug fixes. Because their work is based on Rust projects, it makes sense to be familiar with this programming language, especially when investigating bug fixes in detail.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

QuillBot [2] Grammar Checker for the thesis (v16.0.1/18 January 2025)

ChatGPT [3] Summary of paper [LMTT10] partially reused (GPT v4o/13 October 2024)
Overview of papers for Chapter 2 *Related Work*
Summary of results of research questions partially reused (GPT v4o/21 July 2024)

[2] https://quillbot.com/grammar-check
[3] https://chatgpt.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Glossary

S_i The dataset coming from the results of RQ1 and RQ2 consisting of 1,055 repositories. 22, 24, 26, 34, 39, 42, 43, 50, 51

S_{skg} A subset of the SoftwareKG containing 52 applications for which the source code is available for free. 17, 21, 26, 29

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

CCQ Continuous Code Quality. 9

CI Continuous Integration. 9

DOI Digital Object Identifier. 20

ECA Elementary Code Assessment. 9

FOSS Free Open-Source Software. 8, 11

MSR Mining Software Repositories. 11

NLP Natural Language Processing. 27, 33, 50–53

NVD National Vulnerability Database. 11

OSRS Open-Source Research Software. xiii, 1–3, 7, 8, 11, 22, 50, 51

OSS Open-Source Software. xi, xiii, 1, 2, 8, 10–12, 49

PR Pull Request. 8, 12

RW Retraction Watch. 10

SoftwareKG Software Knowledge Graph. xi, xiii, 3, 4, 10, 11, 14, 15, 17, 21, 22, 24, 29, 31, 42, 43, 49–53, 61

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[Ala20] Adam Alami. The sustainability of quality in free and open source software. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, ICSE '20, page 222–225, New York, NY, USA, 2020. Association for Computing Machinery.

[AMWH19] Daniel A. Almeida, Gail C. Murphy, Greg Wilson, and Michael Hoye. Investigating whether and how software developers understand open source software licensing. *Empirical Software Engineering*, 24(1):211–239, Feb 2019.

[Boc24] Alexey Bochkovskiy. Darknet faq. `https://www.ccoderun.ca/programming/darknet_faq/`, 2024. Accessed: 2024-07-10.

[Boe15] Carl Boettiger. An introduction to docker for reproducible research. *SIGOPS Operating Systems Review*, 49(1):71–79, January 2015.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[Col14] Christian S. Collberg. Measuring reproducibility in computer systems research. In *Proceedings of the 2014 ACM International Symposium on Software Testing and Analysis*, ISSTA '14, page 1–12, New York, NY, USA, 2014. Association for Computing Machinery.

[Dat24] Datacite. Retrieving a single doi. `https://support.datacite.org/docs/api-get-doi`, 2024. Accessed: 2024-07-10.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[Fog17] Karl Fogel. *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly Media, second edition, January 2017. `http://www.producingoss.com/`.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[Git24a] GitHub. Github rate limits. https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28, 2024. Accessed: 2024-06-25.

[Git24b] GitLab. Gitlab rate limits. https://docs.gitlab.com/ee/administration/settings/rate_limit_on_projects_api.html, 2024. Accessed: 2024-06-25.

[GM12] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing, 2012. Preprint available on arXiv.

[IT18] Peter Ivie and Douglas Thain. Reproducibility in scientific computing. *ACM Computing Surveys (CSUR)*, 51(3), July 2018.

[LHW18] Songtao Liu, Di Huang, and Yunhong Wang. Receptive field block net for accurate and fast object detection. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 404–419, Cham, 2018. Springer International Publishing.

[Li13] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *ArXiv*, 1303, 03 2013.

[LMTT10] Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi. Predicting oss trustworthiness on the basis of elementary code assessment. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, New York, NY, USA, 2010. Association for Computing Machinery.

[LWN+20] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics.

[MCCD13] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[MSR23] MSR. Mining software repositories. http://www.msrconf.org/, 2023. Accessed: 2024-06-15.

[NWD+20] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

understanding. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July 2020. Association for Computational Linguistics.

[PMJ+21] Benjamin Paassen, Jessica McBroom, Bryn Jeffries, Irena Koprinska, and Kalina Yacef. Mapping python programs to vectors using recursive neural encodings. *Journal of Educational Data Mining*, 13(3):1–35, October 2021.

[PPS+19] Serena Elisa Ponta, Henrik Plate, Antonino Sabetta, Michele Bezzi, and Cédric Dangremont. A manually-curated dataset of fixes to vulnerabilities of open-source software. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 383–387, 2019.

[SAP24] LeanIX SAP. Understanding open-source licenses. `https://www.leanix.net/en/wiki/trm/open-source-licenses#Types-of-open-source-licenses`, 2024. Accessed: 2024-07-20.

[SE13] Widura Schwittek and Stefan Eicker. A study on third party component reuse in java enterprise open source software. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-Based Software Engineering*, CBSE '13, page 75–80, New York, NY, USA, 2013. Association for Computing Machinery.

[SW24] William Schueller and Johannes Wachs. Modeling interconnected social and technical risks in open source software ecosystems. *Collective Intelligence*, 3(1), 2024.

[SWS+22] William Schueller, Johannes Wachs, Vito Servedio, Stefan Thurner, and Vittorio Loreto. Evolving collaboration, dependencies, and use in the rust open source software ecosystem. *Scientific data*, 2022(9), 2022.

[SYSK23] David Schindler, Erjia Yan, Sascha Spors, and Frank Krüger. Retracted articles use less free and open-source software and cite it worse. *Quantitative Science Studies*, 4, 11 2023.

[SZK19] David Schindler, Benjamin Zapilko, and Frank Krüger. Softwarekg. `https://doi.org/10.5281/zenodo.3571350`, December 2019. Accessed: 2024-06-12.

[SZK20] David Schindler, Benjamin Zapilko, and Frank Krüger. Investigating software usage in the social sciences: A knowledge graph approach. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web*, pages 271–286, Cham, 2020. Springer International Publishing.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[SZK24] David Schindler, Benjamin Zapilko, and Frank Krüger. Softwarekg. https://data.gesis.org/softwarekg/softwarekg-social/, 2024. Accessed: 2024-07-25.

[VPBG18] Carmine Vassallo, Fabio Palomba, Alberto Bacchelli, and Harald C. Gall. Continuous code quality: are we (really) doing that? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18, page 790–795, New York, NY, USA, 2018. Association for Computing Machinery.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WBJS20] Peipei Wang, Chris Brown, Jamie A. Jennings, and Kathryn T. Stolee. An empirical study on regular expression bugs. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 103–113, New York, NY, USA, 2020. Association for Computing Machinery.

[WBL23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475, 2023.

[YBLL17] Geunseok Yang, Seungsuk Baek, Jung-Won Lee, and Byungjeong Lee. Analyzing emotion words to predict severity of software bugs: a case study of open source projects. In *Proceedings of the Symposium on Applied Computing*, SAC '17, page 1280–1287, New York, NY, USA, 2017. Association for Computing Machinery.

[ZD05] Ying Zhou and Joseph Davis. Open source software reliability model: an empirical approach. In *Proceedings of the Fifth Workshop on Open Source Software Engineering*, 5-WOSSE, page 1–6, New York, NY, USA, 2005. Association for Computing Machinery.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Appendix

Repository	Total Commits	Bug Fixes spaCy	Bug Fixes Actual
https://github.com/SindhuMadi/FakeNewsDetection	8	0	0
https://github.com/nicolashernandez/readi-lrec22	98	18	18
https://github.com/tugstugi/mongolian-bert	86	4	4
https://github.com/ayoungkang/pytorch_Yolov4_lane-detection	231	19	18
https://github.com/hhk7734/tensorflow-yolov4	632	46	47
https://github.com/Zehui127/SQUAD_BERT	4	0	0
https://github.com/sarus-tech/tf2-published-models	45	6	6
https://github.com/sadicLiu/yolov3	1	0	0
https://github.com/VinishUchiha/Object-Detection-Using-Yolo4	14	0	0
https://github.com/harsh2011/Yolov3-Detector	2461	43	45
https://github.com/jianmingwuhasco/yolov3	1	0	0
https://github.com/liuqiangict/lamb_optimizer	1	0	0
https://github.com/vwegn/dm	9	2	2
https://github.com/susomena/DeepSlowMotion	26	0	0
https://github.com/pengshuyuan/Bert	2	0	0
https://github.com/adaisti/fin-eval	4	0	0
https://github.com/crx934080895/Bert-CRF_New2	2	0	0
https://github.com/airsplay/vimpac	7	0	0
https://github.com/jiweibo/imagenet	21	0	0
https://github.com/adam123wu/edge-yolo	7	0	0
https://github.com/capstone-w3/trash_parent_repo	196	1	1
https://github.com/Robotmurlock/Deepwalk-and-Node2vec	28	0	0
https://github.com/mim-solutions/bert_for_longer_texts	74	7	6
https://github.com/enalisnick/stick-breaking_dgms	11	0	0
https://github.com/palmagro/gg2vec	115	13	14
https://github.com/Tools4Project/4501Project	53	2	2
https://github.com/rohith2506/word_embeddings	29	0	0
https://github.com/tbmoon/LANL_Earthquake_Prediction	8	0	0
https://github.com/StillKeepTry/Transformer-PyTorch	46	4	4

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

https://github.com/RuoyuGuo/Visualising-Image-Classification-Models-and-Saliency-Maps	2	0	0
https://github.com/Sage-Bionetworks/PythonEmbedInR	572	37	32
https://github.com/jarrydmartinx/generative-models	9	0	0
https://github.com/text-machine-lab/transformerpy	2	0	0
https://github.com/6-dl/darknet_wpb	4	0	0
https://github.com/MLH-Fellowship/Social-BERTerfly	92	10	9
https://github.com/djwhsdj/vw-sdk	117	0	0
https://github.com/keillernogueira/exploit-cnn-rs	2	0	0
https://github.com/cybertronai/megatron-lm	16	1	2
https://github.com/Bhavnicksm/vanilla-transformer-jax	4	0	0
https://github.com/lovedavidsilva/bert_old_version	3	0	0
https://github.com/hiun/learning-transformers	6	0	0
https://github.com/lenacabrera/gb_mnmt	2	0	0
https://github.com/david8862/keras-YOLOv3-model-set	552	94	92
https://github.com/mirzaevinom/data_science_bowl_2018	12	2	2
https://github.com/mindspore-courses/Deep-Tutorials-for-MindSpore	22	0	0
https://github.com/2023-MindSpore-1/ms-code-150	1	0	0
https://github.com/NehaTamore/TextSimilarity	6	0	0
https://github.com/enhuiz/transformer-pytorch	57	7	7
https://github.com/Mind23-2/MindCode-31	1	0	0
https://github.com/Holldean/BERT-Pruning	42	4	4

Table 1: Overview of code repositories that were used for the verification of classified commit messages, including total commits, bug fixes according to spaCy, and actual bug fixes

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Repository	False Positive	False Negative
https://github.com/nicolashernandez/readi-lrec22	1	1
https://github.com/ayoungkang/pytorch_Yolov4_lane-detection	2	1
https://github.com/hhk7734/tensorflow-yolov4	4	5
https://github.com/harsh2011/Yolov3-Detector	3	5
https://github.com/mim-solutions/bert_for_longer_texts	1	0
https://github.com/palmagro/gg2vec	1	2
https://github.com/Sage-Bionetworks/PythonEmbedInR	5	0
https://github.com/MLH-Fellowship/Social-BERTerfly	2	1
https://github.com/cybertronai/megatron-lm	0	1
https://github.com/david8862/keras-YOLOv3-model-set	2	0

Table 2: Overview of code repositories showing number of false positive and false negative bug fixes

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.