REPRODUCIBILITY ASSESSMENT OF RESEARCH CODE REPOSITORIES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Eyüp Kaan AKDENİZ

July 2023
İZMİR

We approve the thesis of **Eyüp Kaan AKDENİZ**

Examining Committee Members:

Assoc. Prof. Dr. Kaya OĞUZ

Department of Computer Engineering, İzmir University of Economics

---

Asst. Prof. Dr. Emrah İNAN

Department of Computer Engineering, İzmir Institute of Technology

---

Assoc. Prof. Dr. Selma TEKİR

Department of Computer Engineering, İzmir Institute of Technology

18 July 2023

Assoc. Prof. Dr. Selma TEKİR

Supervisor, Department of Computer Engineering

İzmir Institute of Technology

Prof. Dr. Cüneyt F. BAZLAMAÇCI

Head of the Department of Computer Engineering

Prof. Dr. Mehtap EANES

Dean of the Graduate School of Engineering and Sciences

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my academic advisor, Assoc. Prof. Dr. Selma Tekir, for her invaluable guidance, support, and motivation throughout this process. I would also like to extend my sincere appreciation to Malik Hinnawi for his enthusiasm and substantial contributions.

I would like to convey my heartfelt thanks to my family, whose unwavering support I have constantly felt throughout this journey. Their encouragement has been a significant source of strength for me.

ABSTRACT

REPRODUCIBILITY ASSESSMENT OF RESEARCH CODE REPOSITORIES

The growth in machine learning research has not been accompanied by a corresponding improvement in the reproducibility of the results. This thesis presents a novel, fully-automated end-to-end system that evaluates the reproducibility of machine learning studies based on the content of the associated GitHub project's Readme file. This evaluation relies on a readme template derived from an analysis of popular repositories. The template suggests a structure that promotes reproducibility. Our system generates a reproducibility score for each Readme file assessed, and it employs two distinct models, one based on section classification and the other on hierarchical transformers. The experimental outcomes indicate that the system based on section similarity outperforms the hierarchical transformer model. Furthermore, it has a superior edge concerning explainability, as it allows for a direct correlation of the scores with the respective sections of the Readme files. The proposed framework provides an important tool for improving the quality of code sharing and ultimately helps to increase reproducibility in machine learning research.

ÖZET

ARAŞTIRMA KOD DEPOLARININ YENİDEN ÜRETİLEBİLİRLİK DEĞERLENDİRMESİ

Makine öğrenimi araştırmalarındaki büyümeye, sonuçların tekrar üretilebilirliğinde buna karşılık gelen bir gelişme eşlik etmemiştir. Bu tez, ilişkili GitHub projesinin Readme dosyasının içeriğine dayalı olarak makine öğrenmesi çalışmalarının yeniden üretilebilirliğini değerlendiren yeni, tam otomatik bir uçtan uca sistem sunmaktadır. Bu değerlendirme, popüler depoların analizinden türetilen bir readme şablonuna dayanmaktadır. Şablon, yeniden üretilebilirliği teşvik eden bir yapıyı önerir. Sistemimiz, değerlendirilen her Readme dosyası için bir yeniden üretilebilirlik puanı üretir ve biri bölüm sınıflandırmasına, diğeri hiyerarşik dönüştürücülere dayanan iki farklı model kullanır. Deneysel sonuçlar, bölüm benzerliğine dayalı sistemin hiyerarşik dönüştürücü modelinden daha iyi performans gösterdiğini göstermektedir. Ayrıca, skorların Readme dökümanlarının ilgili bölümleriyle doğrudan ilişkilendirilebilmesi açısından üstün bir açıklanabilirliğe sahiptir. Önerilen çerçeve, kod paylaşımının kalitesini artırmak için önemli bir araç sunmakta ve sonuçta makine öğrenimi araştırmalarında yeniden üretilebilirliğin arttırılmasına yardımcı olmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES ix

LIST OF TABLES x

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. REPRODUCIBILITY 4

2.1. Terminology 4

2.2. Significance 7

2.3. Crisis 8

2.4. Problems and Solutions 9

2.4.1. Code and Data Availability 9

2.4.2. Incompleteness of Materials 10

2.4.3. Missing Consensus and Assessment Methodologies 12

2.4.4. Platforms, Practises and Documentation 13

CHAPTER 3. BACKGROUND 16

3.1. Word Embeddings 16

3.2. Transformers 17

3.2.1. BERT 18

3.2.2. Sentence-BERT 18

3.3. Hierarchical Models 19

3.4. Zero-shot Learning 20

CHAPTER 4. METHODOLOGY 21

4.1. System Workflow 23

4.1.1. Readme Parsing 23

4.1.1.1. Readme Sections 24

4.1.2. Readme Processing 25

4.1.2.1. Section Classification 25
4.1.2.1..1 Reproducibility Score Generation 25
4.1.2.1..1.1 Base Formulation 26
4.1.2.1..1.2 Consecutive Formulation 27
4.1.2.1..1.3 Score Calculation Example 28
4.1.2.2. Readme Classification (Hierarchical Transformers) 29
4.2. Data 30
4.2.1. The Association of Computer Linguistics (ACL) Papers Dataset 30
4.2.2. Section Labeling 31
4.2.2.1. Text Similarity 32
4.2.2.2. Zero-shot Classification 32
4.2.2.3. Manual Annotation 33
4.2.2.3..1 Inter-annotator Agreement 33
4.2.2.4. Training Data for Hierarchical Transformers 34

CHAPTER 5. EXPERIMENTS 35
5.1. NeurIPS Papers 35
5.2. Evaluation Metrics 36
5.3. Results 36
5.3.1. Classification-Based System 36
5.3.2. Hierarchical Transformers 39

CHAPTER 6. WEB APPLICATION 40
6.1. Application Workflow 40

CHAPTER 7. CONCLUSION & FUTURE WORK 45

REFERENCES 47

APPENDICES 57

APPENDIX A. METHODOLOGY 57

1.1. ACL Anthology Events 57
1.2. Headers of Dropped Sections 57
APPENDIX B. EXPERIMENTS 58
2.1. Training Results Based on Section Contents 58
2.1.1. Runtime Information 58
2.1.1.1. Training 58
2.1.1.2. Labeling 59
2.1.1.3. System Evaluation 59
2.1.1.4. Readme Parsing 59

LIST OF FIGURES

Figure **Page**

Figure 2.1. The spectrum of reproducibility (Source: Peng (2011)) 5
Figure 2.2. Reproducible research (Source: Kirstie (2016)) 6
Figure 2.3. Levels of reproducibility (Source: Gundersen (2021)) 6
Figure 2.4. Number of all and coded articles by year in ACL. 8
Figure 2.5. The factors and variables of reproducibility (Source: Gundersen et al. (2023)) 12
Figure 4.1. End-to-end System Workflow 21
Figure 6.1. System Screen after URL Input 40
Figure 6.2. System Parameters 41
Figure 6.3. Reproducibility Score Output 42
Figure 6.4. Checklist Tab 43
Figure 6.5. Analysis Tab 44

LIST OF TABLES

Table **Page**

Table 2.1. Code repository usage statistics. 14

Table 4.1. Readme template sections and example contents. (Source: Paperswithcode (2020)) 22

Table 4.2. Readme element transformations. 24

Table 4.3. Example of Section Classification. 28

Table 4.4. Parsed section statistics. 31

Table 4.5. Automatic labeling performance of different section contents. 34

Table 5.1. Example of NeurIPS Dataset. 35

Table 5.2. Section Classification-Based System's Evaluation Results 37

Table 5.3. System Evaluation Results Based on Labeling Method of Training Data. 37

Table 5.4. System Evaluation Results Based on Labeling Content of Training Data. 38

Table 5.5. System Evaluation Results Based on Scoring Types. 38

Table 5.6. System Evaluation Results Based on Evaluation Model. 39

CHAPTER 1

INTRODUCTION

Over the last decade, research in machine learning has increased, but there has been insufficient improvement in reproducibility (Peng, 2011), a key metric for assessing claims in the scientific community. The scientific community is grappling with a reproducibility crisis where the increase in research output has not been accompanied by a commensurate increase in the reproducibility of published work (Baker, 2016). Despite initiatives to increase the sharing of source code in computer science research under the Open Science Initiative (Easterbrook, 2014), a significant proportion of papers (74%) are not reproducible (Stodden et al., 2018).

As sound scientific claims require reproducibility, the growing mismatch between the volume of research produced and its reproducibility is a cause for concern. However, some progress has been made, with studies showing that the adoption of open data policies and the inclusion of source code in research papers increases reproducibility rates (Laurinavichyute et al., 2022).

However, code and data sharing alone are insufficient unless strict standards are followed, making it difficult to verify the accuracy of shared material. Academia has proposed various standardization mechanisms such as checklists, datasheets, and reproducibility challenges (Gundersen et al., 2023). However, the lack of a universal definition, measurement, and approach to reproducibility has led to disagreements (Belz et al., 2021). The verification process for conformance to standards remains manual, which can lead to time-consuming and potentially less reliable results.

Despite the growing trend of code sharing with research articles, these offerings are often incomplete or inadequate. Vandewalle (2019)'s study found that 71% of the link addresses associated with articles were invalid, making the usability of the code even more challenging. Code quality is another major barrier to the successful replication of experiments. Common issues include missing classes or interfaces, dependencies on

specific files, and deprecated methods (Mondal and Roy, 2021). To ensure the reproducibility of code, the importance of robust documentation, including clear workflows, registered working protocols, and well-maintained, readable code, has been emphasized (Diaba-Nuhoho and Amponsah-Offeh, 2021).

Machine learning papers often share their source code through repositories such as GitHub, Bitbucket, and GitLab. An important feature of these repositories is the "Readme" markdown file, which provides instructions for researchers to reproduce the results reported in the papers. These files play an important role in the reproducibility of the research by acting as a bridge between the complex codebase and the research article (Obels et al., 2019). Thus, the quality and comprehensiveness of these Readme files are very important.

In this work, we present a novel, automated end-to-end system designed to assess the reproducibility of machine learning articles. The system uses a GitHub project link to generate a reproducibility score based on the project's Readme file. Our workflow covers two different models: one model uses chapter classification, while the other uses hierarchical transformers (Chalkidis et al., 2022). Both models aim to measure the reproducibility of the research in question.

Our evaluation is based on a readme template proposed by a well-known platform for sharing research code. This template was developed by examining popular repositories and identifying common elements associated with their success (Paperswithcode, 2020). The template proposes six sections for the readme of a reproducible project: Introduction, Requirements, Pre-trained Models, Training, Evaluation, and Results, each with a short description.

We propose a comprehensive framework for assessing whether Readme files meet the criteria stipulated by this template and provide a reproducibility score for each Readme. In order to validate our system, we utilize a separate hold-out test set comprised of reproducibility-reviewed papers from the NeurIPS 2019 conference (Paperswithcode, 2020). This test set provides a reliable foundation for gauging the effectiveness of our proposed framework given that the papers included have already been vetted for their reproducibility.

We have also made our system accessible as a public resource (Akdeniz, 2023a), to allow researchers and practitioners alike to leverage our system in assessing the repro-

ducibility of their own or others' works. Our aim is to provide a practical, user-friendly tool that can assist in the improvement of research reproducibility in the machine learning field.

Furthermore, in the spirit of transparency and reproducibility, we have openly shared our codebase, datasets, and trained models (Akdeniz, 2023b). By making our system and materials available to the public, we aim to encourage scrutiny, iteration, and extension of our work by the broader scientific community. It's our belief that openness and collaboration will drive improvements in reproducibility, contributing to the advancement of machine learning research.

CHAPTER 2

REPRODUCIBILITY

In this chapter, the terminology is systematically described, and the importance of reproducibility in scientific research, particularly in machine learning (ML), is emphasized. Both the successes and challenges encountered are discussed, along with potential solutions proposed to address these limitations. Additionally, the areas still in need of improvement are identified.

2.1. Terminology

Reproducibility is one of the building blocks of scientific progress. Despite this, the definition of it is still controversial (Plesser, 2018). In particular, it is used interchangeably with repeatability and replicability, and there is no general acceptance of their differences or similarities. According to the report of Association for Computing Machinery (2016), among those who argue that they are different, these concepts are explained in terms of experimenters and experimental design variables as follows; *Repeatability* refers to the same team getting the same results in a large number of trials, given the same experimental conditions. *Replicability* refers to the ability of different teams to obtain the same results in a large number of trials given the same experimental condition. *Reproducibility* refers to the ability of different teams to obtain the same results given different experimental conditions. In Committee on Reproducibility and Replicability in Science et al. (2019)'s study, the differences between these concepts are explained using different variables. In providing consistent results, given the input data, computational steps, methods, code, and analysis conditions, it is considered reproducibility if they are all the same and replication if only the data is different. Similarly, according to Peng (2011), the same data should be used for reproducibility, while different data should be used for replicability. Apart

from this definition, he states that in order to talk about the reproducibility of work, at least its code should be shared and argues that reproducibility is the lowest condition of replication. He expressed his idea with the spectrum in Figure 2.1.

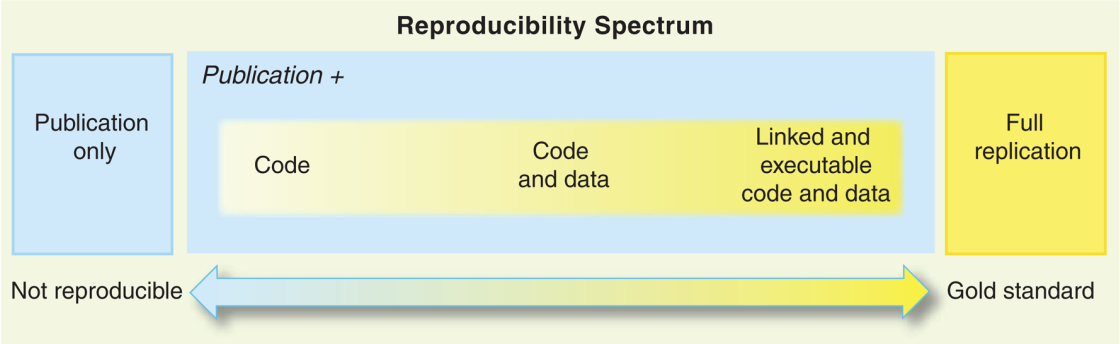

Figure 2.1. The spectrum of reproducibility (Source: Peng (2011))

Repeatability and reproducibility are not very different from each other, and they are the terms providing each other according to the Albertoni et al. (2023). But replicability and reproducibility differ certainly. Goodman et al. (2016) takes a different perspective on all these ambiguities, stating that reproducibility and replicability are not different, but different levels of each other, and divides reproducibility into three categories: a) *Methods reproducibility:* providing sufficient detail so that procedures can be replicated exactly, b) *Results reproducibility:* repeating procedures independently and obtaining results that are very close to the original, c) *Inferential reproducibility:* conducting a new study separate from the original or re-examining the original study to reach similar results. According to Goodman et al. (2016), these explanations resolve the confusion caused by the proximity of the lexical meanings of these words in their scientific meaning.

Views on reproducibility can be categorized into three main groups: those who argue that they are absolutely different, those who think that they express different levels, and those who claim that they are exactly the same (Barba, 2018). Barba (2018); Gundersen et al. (2023); Albertoni et al. (2023); Plesser (2018) have extensively covered these views in their work. However, according to Gundersen et al. (2023), these definitions are very open to interpretation, broad in scope, and imprecise.

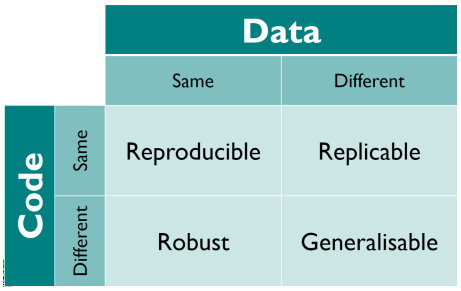

Figure 2.2. Reproducible research (Source: Kirstie (2016))

In the context of machine learning, according to some studies (Belz et al., 2021; Pineau et al., 2020), definitions vary based on the state of the code and data variables, as shown in Figure 2.2. Different from the previous terms, there are two new concepts here: *Generalizable* means that the same results are obtained with different codes and different data, while *Robust* means that the same code gives the same results with different data. On the other hand, Gundersen and Kjensmo (2018); Gundersen (2019, 2021) consistently recognizes in their work that they are the same thing, and even if they are not, these discussions do not help to resolve them. Gundersen (2021) classifies reproducibility on 4 different levels depending on the variables text, code, and data, as shown in Figure 2.3. In this figure, the text field refers to any document containing descriptions of experimental procedures and although the availability of code and data is variable, it shows that there should be documentation at all levels. Full experimental reproducibility is attained when we have access to the corresponding document, code, and data simultaneously. In cases where only code or data is available, we can discuss the reproducibility of each of these components separately.

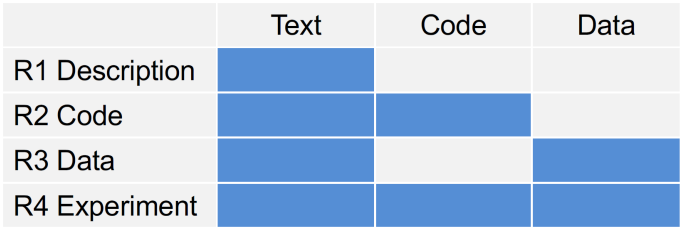

Figure 2.3. Levels of reproducibility (Source: Gundersen (2021))

In conclusion, although conceptual debates continue, the results of a study should

not be a one-off but should be suitable to be analyzed under different conditions. As Peng (2011) points out, reproducibility is an indicator of minimum standards for evaluating research results, and the most crucial requirements for reproducibility are code and data.

In the aftermath of all these debates, we position ourselves on the side where there are no clear-cut lines between these definitions. Our primary focus is less on defining what reproducibility is, and more on highlighting the importance of the criteria necessary to ensure it and these criteria need to be evaluated based on their feasibility.

2.2. Significance

Science works for our curiosity about the world, propelling new discoveries through a process of inquiry. However, these discoveries only mark the beginning. Each one needs to be validated by others before it can be universally accepted. The evolution of existing knowledge hinges directly on the sustainability of this iterative validation process. Unverified results are often built upon those accepted by others, setting the stage to be verified in the course. This ongoing chain of verification symbolizes the trust and progress inherent in the scientific process (Committee on Reproducibility and Replicability in Science et al., 2019). These verifications are made possible by ensuring the reproducibility of results and it is one of the most crucial prerequisites for the reliability of scientific results. Otherwise, The incapacity to reproduce the results of scientific research often leads to mistrust about their validity (Belz et al., 2021). Moreover, if other researchers attempt to reproduce the original experiment but fail to obtain the same results, the original hypothesis is typically considered invalid (Oates, 2006).

Scientific progress is a gradual process that can be accelerated by making reproducibility a standard practice (Peng, 2011). Following best practices for reproducibility allows for the efficient application of established procedures to new data and facilitates code reuse. Good reproducibility habits can lead to substantial time savings (Sandve et al., 2013). In essence, the more reproducibility is achieved, the greater its impact on both progress and trust in science.

As depicted in Figure 2.4, there has been a significant increase in the number

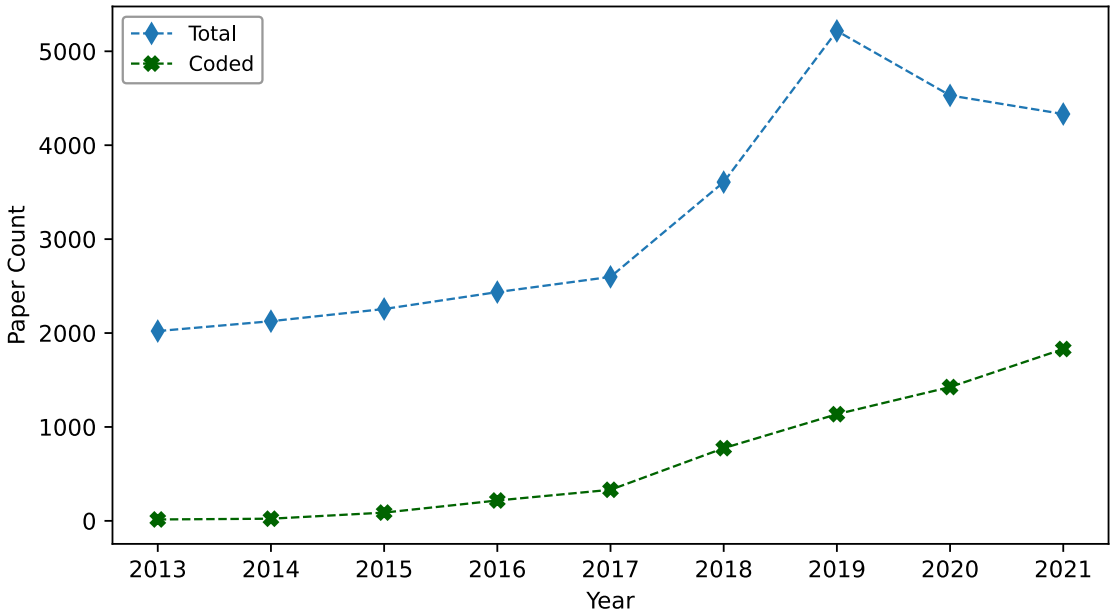

Figure 2.4. Number of all and coded articles by year in ACL.

of studies conducted at conferences and published in journals under the auspices of The Association for Computational Linguistics (ACL). Correspondingly, the number of studies that share code is also on the rise. As indicated in The Association for Computational Linguistics (2023b), materials shared within the ACL began to be licensed in 2016, and since that time, code sharing has increased. While this is an important step towards reproducibility, the proportion of shared materials does not provide clear evidence of their reproducibility. It suggests that sharing code and data alone is insufficient to ensure reproducibility. However, it can be argued that code and data sharing accelerate scientific development and demonstrates the importance of reproducibility, especially in the field of computational sciences (Stodden et al., 2018).

2.3. Crisis

Reproducibility is a foundational principle of scientific research and is under threat, with research indicating a substantial rate of failure in reproducing scientific results. Specifically, approximately 70% of researchers reported failure in reproducing others' results, and over half reported failure in replicating their own studies. This phenomenon

referred to as the '*reproducibility crisis*', questions the validity of these findings (Baker, 2016). An alarmingly high volume of published research is not reproducible, illustrating the magnitude of this crisis (Gundersen and Kjensmo, 2018). This problem is prevalent across scientific domains, including exploratory research, which forms the first step in creating new knowledge Committee on Reproducibility and Replicability in Science et al. (2019). But, it seems even more intense in the field of machine learning (ML) and natural language processing (NLP) (Pedersen, 2008; Mieskes et al., 2019; Belz, 2021).

Looking at the AI and ML fields, reproducibility issues are dominant. A troubling scarcity of shared code and data has been reported. For instance, a survey found that just 6% of algorithm presenters at top AI conferences shared their code, and only a third disclosed their testing data Hutson (2018). Reasons for these problems vary from ongoing development and proprietary ownership to researchers' competition concerns. The crisis of reproducibility is growing quickly due to the complex nature of AI and ML models Albertoni et al. (2023). This issue calls for a joint permanent effort from the scientific community to tackle this situation and improve reproducibility Peng (2011).

2.4. Problems and Solutions

Reproducibility is a challenging problem that involves many factors and lacks a direct solution (Goodman et al., 2016). Despite clearer problem definitions and the proposal of different solution methods, achieving a complete resolution still requires more time and effort (Munafò et al., 2017; Arvan et al., 2022).

2.4.1. Code and Data Availability

While sharing academic papers openly was a significant challenge in the past, open access (OA) has become increasingly prevailing in recent years. However, problems have emerged and persisted regarding the sharing of code and materials associated with these papers (Piwowar et al., 2018). In a study by Wieling et al. (2018), data sharing increased

by approximately 10% and code sharing by around 25% from 2011 to 2016, resulting in a total of 59.3% of shared codes. However, as reported in a study by Vandewalle (2019), in a dataset analyzed in 2019, only 71% of the source code was available. Despite a noticeable increase in the past decade, these results indicate that code and data sharing are still obstacles to reproducibility.

Similar to the open access movement, academic institutions are also playing an important role in addressing this issue. Recently, many educational institutions and conference organizers have implemented open data policies that have been shown to clearly contribute to reproducibility (Laurinavichyute et al., 2022; Liu et al., 2022; Stodden et al., 2018). However, as mentioned in our previous definition (see section 2.1.), data sharing alone is not enough to ensure reproducibility. It has been emphasized that the code associated with the related work should also be made accessible (Laurinavichyute et al., 2022; Nature, 2021).

Following the emergence of the open source movement in the software community, its impact echoed globally, and a similar culture gradually took root in academia. As this culture continues to spread, barriers to reproducibility will decrease, and scientific progress will accelerate (Peng, 2011).

2.4.2. Incompleteness of Materials

The increasing practice of sharing code and data brings considerable benefits, but sticking to proper standards is essential. Without this, it becomes challenging to verify the accuracy of these resources. As highlighted in a study by Mondal and Roy (2021), code quality presents barriers to experiment reproducibility, often due to issues like outdated methods, file dependencies, and the lack of interfaces or classes, and these results show that the verification of these materials is compulsory. In addition, Diaba-Nuhoho and Amponsah-Offeh (2021) state that the fundamental aspect of reproducible code is substantial documentation, including open workflows, registered study protocols, methodology, and code that is readable and well-maintained.

Given the fast-paced growth of study and the need for verification of materials,

controlling the situation becomes nearly impossible. To meet this demand in academia, several standardization measures such as checklists, data sheets, and reproducibility challenges have been suggested (Gundersen et al., 2023; Albertoni et al., 2023). Organized challenges aim to encourage the replication of previous studies by different researchers, with the goal of producing consistent results. Through these efforts, the intention is to foster a culture of reproducibility within the scientific community (Liu et al., 2022).

In their study, Gundersen et al. (2023) extensively classified the research process from experimental design to documentation, highlighting the necessary steps for ensuring reproducibility. Through their findings, they presented a framework that researchers can follow to facilitate reproducibility in their work.

Guidelines play a vital role in fostering reproducibility in machine learning as they provide researchers with a set of recommended practices and standards. By adhering to these guidelines, researchers can ensure the transparency, reliability, and replicability of their work (Albertoni et al., 2023). They categorized guidelines that contribute to promoting reproducibility:

1. Survey studies or position papers that offer recommendations.
2. Reproducibility checklists.
3. Guidelines to be followed during paper submissions.
4. Academic resources that discuss the issues and solutions surrounding insufficient reproducibility of research.

Additionally, they comprehensively presented other recommendations and suggestions in their study. Their conclusion is as follows, although there has been a rise in understanding and numerous investigations into the significance of reproducibility, its practical implementation remains inadequate when it comes to advanced machine learning and deep learning techniques. Building AI models involves various factors that lead to technical dependencies which impose challenges on achieving reproducibility.

2.4.3. Missing Consensus and Assessment Methodologies

There is a lack of consistent agreement, measurement, and approach to reproducibility across academic institutions. Instead, there is an increasing variety of perspectives and opinions on this matter (Belz et al., 2021). Moreover, the process of verifying standards is done manually, which leads to longer procedures and less trustworthy outcomes. Although there have been studies on automating the review of research papers (Yuan et al., 2021), meaningful progress has not been achieved in automating the verification of shared materials.

Efforts have been made to facilitate manual processes or expand the scope of addressing these problems. In a study conducted by Gundersen (2019), three factors—method, data, and experiment—were identified, along with their respective subvariables as shown in Figure 2.5, and argues that an assessment can be made based on the fulfillment of the defined criteria for each variable.

Factor	Variable	Description
Method	Problem	Is there an explicit mention of the problem the research seeks to solve?
	Objective	Is the research objective explicitly mentioned?
	Research method	Is there an explicit mention of the research method used (empirical, theoretical)?
	Research questions	Is there an explicit mention of the research question(s) addressed?
	Pseudocode	Is the AI method described using pseudocode?
	Hypothesis	Is there an explicit mention of the hypotheses being investigated?
	Prediction	Is there an explicit mention of predictions related to the hypotheses?
	Experiment setup	Are the variable settings shared, such as hyperparameters?
Data	Training data	Is the training set shared?
	Validation data	Is the validation set shared?
	Test data	Is the test set shared?
	Results	Are the relevant intermediate and final results output by the AI program shared?
Experiment	Method source code	Is the AI system code available open source?
	Experiment source code	Is the experiment code available open source?
	Software dependencies	Are software dependencies specified?
	Hardware	Is the hardware used for conducting the experiment specified?

Figure 2.5. The factors and variables of reproducibility (Source: Gundersen et al. (2023))

In their study, Pouchard et al. (2023) proposed a framework named Uncertainty-

Aware Quantification(UQ) based on the measurement of uncertainties, recognizing the difficulty of predicting the outputs of ML models, and they argue that reproducibility can be improved by measuring uncertainties.

Belz et al. (2022) discuss the application of metrology meta-science methods to measure the reproducibility of ML/NLP research. This measurement approach provides an evaluation by considering multiple reproductions of a study and is called Quantified Reproducibility Assessment (QRA). While this study provides a valuable quantitative metric in the field, a limitation arises because studies without reproducible examples cannot be subjected to this measurement.

While we have examined some studies focusing on quantitative analysis beyond qualitative analysis, some of these studies remain at the recommendation level, while others are subject to specific conditions for their applicability. In the current state, a consensus regarding reproducibility is still not apparent (Belz et al., 2021), and the proposed solutions for reproducibility are deemed insufficient (Albertoni et al., 2023).

2.4.4. Platforms, Practises and Documentation

As mentioned earlier, code and data sharing alone is not enough to ensure reproducibility and requires the adoption of certain standards for these materials. In this context, Crane (2018) emphasizes that reproducibility cannot be achieved through code sharing alone, but rather requires supporting environments. The platforms developed for this purpose were evaluated by Isdahl and Gundersen (2019) according to criteria such as code versioning, software dependencies, and traceability of model and results. The results show that although there are platforms that meet these standards, they are not widely used in academia and the platforms that are widely used do not possess these features.

According to our analysis of the collected data, Table 2.1 shows the distribution of article codes across different providers. According to the data, GitHub (GitHub, 2023) is the most widely used platform by a large margin. Papers with Code (Papers with Code, 2023) is not a code repository, but a platform that presents articles with their associated code from authors or the community. The number in the figure represents the community

Platform	Count
GitHub	8967
Papers With Code	234
Bitbucket	41
GitLab	26
Others	32

Table 2.1. Code repository usage statistics.

code. Just as arXiv (arXiv, 2023) means open access for articles, Papers with Code have the same importance for sharing the code associated with articles (Lucic et al., 2022).

The main purpose behind these platforms is version control and open access. To ensure reproducibility, the entire code development process should be complemented by the active use of version control systems (Nguyen and Rampin, 2022). The documentation should also indicate which version of the code the provided details correspond to. In academic research, however, code is often uploaded to such providers after the research is completed, and version control practices are not consistently applied throughout the research period (Sandve et al., 2013; Wilson et al., 2016; Taschuk and Wilson, 2017).

While it is common practice to describe written code and algorithms in research papers, this practice is insufficient to ensure reproducibility in today's context (Sandve et al., 2013; Wieling et al., 2018; Gundersen et al., 2023). Regardless of how well-written the shared code is, extensive documentation is necessary for others to easily understand it. Unfortunately, the quality of code written for academic purposes often falls short of established standards (Joppa et al., 2013). Moreover, given the complexity of software in the field of machine learning (Arvan et al., 2022), the quality of documentation directly impacts reproducibility. Therefore, code in research papers requires more than simple documentation; it needs comprehensive and informative documentation (Gundersen et al., 2023; Wilson et al., 2016).

In the software community, the most commonly used form of documentation is a text file in markdown format called a *"Readme"* uploaded to code repositories. This serves as the primary document that appears when opening the project and provides a project description along with relevant details. In the context of academic research, once the code is uploaded to repositories, documentation of the code and data through this file is essential for reproducibility (Obels et al., 2019; Trisovic et al., 2022; Taschuk and

Wilson, 2017). However, according to a study by Zhang (2019) on code repositories of academic research, the documentation in these repositories is often insufficient. A checklist introduced at NeurIPS 2019 Pineau et al. (2020), which aims to standardize both code content and documentation, has been a significant step towards standardization of reproducibility and has gained acceptance among other researchers (Albertoni et al., 2023; Belz, 2021).

In conclusion, when we examine the problems and proposed solutions on reproducibility, we can see that these problems are not impossible to solve. But, in current status, some proposed solutions have a limited scope, and others may bring new challenges. However, it is clear that dedication is required by both researchers and assessors and that the process is time-consuming. Still, in any case, standards should be established and evaluated to ensure reproducibility (Belz, 2021; Belz et al., 2022). Automating the evaluation processes is of great importance for reproducibility assessment and improvement, as it can streamline the processes and reduce the time spent (Trisovic et al., 2022; Nüst and Eglen, 2021).

CHAPTER 3

BACKGROUND

In this chapter, the background information about the research is outlined. Specifically, our approach incorporates the use of word embeddings, transformers, text similarity, and classification techniques.

3.1. Word Embeddings

Word embeddings are a form of word representation common in the field of natural language processing. Basically, they are based on the distributional hypothesis, which claims that words that appear in similar contexts often contain similar semantic definitions (Harris, 1954).

The Vector Space Model (VSM) by Salton et al. (1975), is recognized as one of the best and most effective models for transforming words into vectors. The main idea behind this model is the representation of words in a vector space. These vector representations of words are used in several sub-tasks like determining text similarity, and question answering Almeida and Xexéo (2023).

One of the early versions of word embeddings is Latent Semantic Analysis (LSA) proposed by (Deerwester et al., 1990). LSA uses a method called singular value decomposition on a matrix of words and documents to represent words in a large and complex space. This helped to some extent to understand the connections between words. However, the way we see words today as tightly packed numerical vectors started with models based on neural networks.

Neural network-based word embeddings were introduced in the work of Bengio et al. (2000). In their pioneering research, they presented a neural network-based language model that learns to represent words in a distributed way and delivers top results in tasks

related to language modeling. However, this model was computationally expensive. To address this problem, Mikolov et al. (2013) introduced two different designs under the name Word2Vec: Continuous Bag of Words (CBOW) and Skip-gram. The CBOW model aims to predict a word based on its surrounding words, while the Skip-gram model does the opposite by predicting the surrounding words given a target word. Then Pennington et al. (2014) introduced the GLoVe (Global Vectors) model, which combines global matrix factorization methods (e.g. LSA) and the methods offered by Word2Vec.

These word embeddings are an important step forward in the representation of text and led to the development of complex language models based on Recurrent Neural Networks (RNNs) (Elman, 1990) and Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks.

Recently, the BERT (Bidirectional Encoder Representations from Transformers) model and its derivatives, introduced by Devlin et al. (2019) and built on the transformer architecture proposed by Vaswani et al. (2017), have been widely used in various applications.

Utilizing a vector-based representation for words allows the assessment of their similarities via methods such as cosine similarity and other vector measurement techniques (Mikolov et al., 2013). As the accuracy of these vector representations improves, it subsequently enhances the performance of associated sub-tasks (Devlin et al., 2019).

3.2. Transformers

Vaswani et al. (2017) introduced Transformers in their research paper "Attention is All You Need". Transformers introduced a new way to manage tasks involving sequence transmission while overcoming computational inefficiencies observed in RNNs and LSTM networks.

The transformer architecture is based on the attention mechanism. Unlike RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) networks that process data sequentially, the transformer model allows simultaneous attention to different parts of the input word sequence, allowing dependencies between words to be determined

independently of their position. Unlike previous models that encounter difficulties in capturing dependencies as the length of a sequence increases, transformers excel at handling long-range dependencies. Furthermore, these models can be trained in parallel, making the training process more efficient and requiring less time.

Since its introduction, transformer architecture has made a significant impact in the field of natural language processing (NLP), and models built on this architecture have achieved state-of-the-art performance on various NLP tasks.

3.2.1. BERT

The BERT (Bidirectional Encoder Representations from Transformers) model, built on transformer architecture, has led to a significant transformation in the field of natural language processing (NLP). Devlin et al. (2019) introduced a new approach for building pre-trained language models, pushing the boundaries of performance on NLP tasks.

BERT is pre-trained on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). In the MLM task, the model learns to predict masked words in a sentence, thus capturing semantic and syntactic features of the language. In the NSP task, the model learns to predict the next sentence given the previous sentence, thus capturing the relationships between words and sentences.

One of the most important and remarkable features of BERT is that it can be fine-tuned for various NLP sub-tasks such as text classification, question answering, and named entity recognition (NER) without the need for large datasets. It can be adapted and applied to many different domains according to specific needs.

3.2.2. Sentence-BERT

The Sentence-BERT (SBERT) model developed by Reimers and Gurevych (2019) is a specialized version of BERT, specifically designed to represent sentence embeddings.

Generating word embeddings using BERT for semantic similarity tasks can be computationally expensive. However, SBERT is advantageous in terms of both efficiency and performance as it is trained to generate sentence embeddings directly.

SBERT adds a pooling layer on top of the output of the transformer network to obtain fixed-size sentence embeddings. During training, Siamese or triplet network architectures are used to produce semantically meaningful embeddings. Siamese networks compute embeddings for two input sentences using two identical networks, while triplet networks use three different sentences: an anchor sentence, a semantically different sentence, and a semantically similar sentence.

It can compute sentence embeddings in a single iteration and provides faster computation. The sentence-transformers library Reimers (2019), developed as part of this work, has been widely used since its development.

3.3. Hierarchical Models

Hierarchical models are designed to create representations of long texts in vector space and can be used for sub-tasks in NLP. As the name suggests, they take into account the hierarchical structure of the text. They work based on using the attention mechanism not only between words but also between sentences and paragraphs.

The Hierarchical Attention Network (HAN) model proposed by Yang et al. (2016) uses Gated-Recurrent Units (GRU)(Cho et al., 2014) and two different attention mechanisms at the word and sentence levels to sequentially identify semantically important words and sentences to build document vector representation. With the model they developed, they achieved impressive results in document classification tasks compared to other models at that time.

On the side of the transformers, the transformer model of Vaswani et al. (2017) is not efficient for processing long documents due to its quadratic computational cost. Also, BERT and its derivatives have token limits. To address these issues, models using different attention mechanisms such as Transformer-XL (Dai et al., 2019), Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2021) have emerged. However, none of

these models take into account the hierarchical structure of documents. Chalkidis et al. (2022) used hierarchical-based attention for long document classification and achieved superior performance in terms of both resource utilization and speed compared to other models. It also showed overall better performance on downstream tasks. Similarly, the hierarchical-based transformer model developed by Nawrot et al. (2022) outperformed non-hierarchical models. The results show that hierarchical models can handle long texts more efficiently.

3.4. Zero-shot Learning

Zero-shot learning refers to a paradigm in machine learning in which a model has the capacity to deliver precise predictions for classes that were not exposed to it during the training phase (Norouzi et al., 2014). It has the strength of generalizing to unseen categories, making it effective for recognizing instances that might not exactly align with any template but carry semantic relevance (Xian et al., 2019; Rios and Kavuluru, 2018).

Zero-shot learning first gained popularity in computer vision but has also shown potential in NLP applications. It works by using embeddings represented in vector space. Different models have been developed using statistical methods such as K-NN and neural networks such as Siamese Networks (Rios and Kavuluru, 2018). With the birth of transformers, pre-trained models have been found to be effective for zero-shot learning with their ability to efficiently learn and represent natural language (Dickinson et al., 2021; Zhao et al., 2023).

Supervised learning methods rely heavily on labeled data, which is often manually annotated by human labelers. However, the reliability of human annotations is questionable, and given the substantial volume of data required to train these models, obtaining a sufficient amount of labeled data can be a time-consuming process. Zero-shot classification techniques have shown promise in efficiently labeling datasets and achieving results comparable to human labelers in a relatively short time. (Rondinelli et al., 2022). It has also been reported that fine-tuning pre-trained models with data labeled with this method can be successful (Bujel et al., 2021).

CHAPTER 4

METHODOLOGY

In this section, details of the proposed system design are delved into by elaborating on the scoring mechanism employed in the research, as well as discussing the data used and the methods of data labeling.

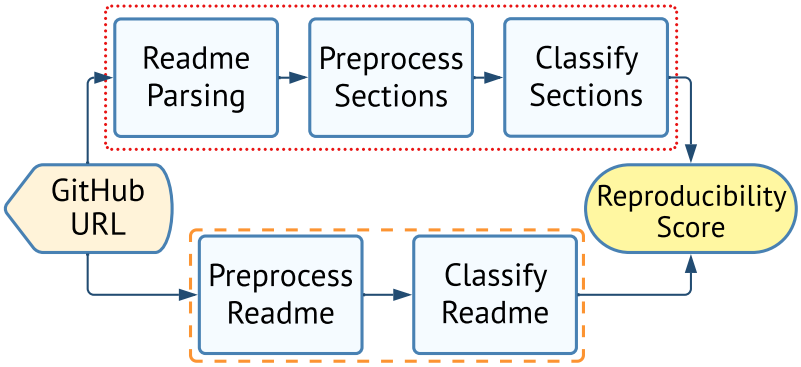

Figure 4.1. End-to-end System Workflow

The readme file is recognized as the foundational document for code documentation as discussed in Section 2.4.4.. With this premise, our objective is to evaluate the reproducibility of research efforts through an in-depth analysis of these Readme files. To establish the ground truth for our measurements, we adopt the 'readme' template (Paperswithcode, 2020). This template is recommended by a popular platform for sharing research code (Papers with Code, 2023) and has found application within academic contexts (Belz, 2021).

This template is created by examining existing repositories, identifying those that received the most positive response within the community, and then pinpointing common elements that correlate with popularity Paperswithcode (2020). The template suggests six sections for a readme of a reproducible project: Introduction, Requirements, Pre-trained

Section	Description
Introduction	This repository is the official implementation of <LINK>. Optional: includea graphic explaining your approach/main result, bibtex entry, link to demos,blog posts and tutorials
Requirements	To install requirements: <CODE> Describe how to set up the environment,e.g. pip/conda/docker commands, download datasets, etc...",
Training	To train the model(s) in the paper, run this command: <CODE> Describe how to train the models, with example commands on how to train the models in your paper, including the full training procedure and appropriate hyperparameters.
Evaluation	To evaluate my model on ImageNet, run: <CODE> Describe how to evaluate the trained models on benchmarks reported in the paper, give commands that produce the results (section below).
Pre-trained Models	You can download pretrained models here: <LINK> trained using parameters x,y,z. Give a link to where/how the pretrained models can be downloaded and how they were trained. Alternatively you can have an additional column in your results table with a link to the models.
Results	Our model achieves the following performance on <TASK>: <TABLE> Include a table of results from your paper, and link back to the leaderboard for clarity and context. If your main result is a figure, include that figure and link to the command or notebook to reproduce it.

Table 4.1. Readme template sections and example contents. (Source: Paperswithcode (2020))

Models, Training, Evaluation, and Results. It also provides an example content of each section as presented in Table 4.1. The elements in this template represent the Machine Learning reproducibility checklist.

In the Introduction section, general information about the conducted research and the shared codes is expected to be provided. The Requirements section should provide information on the libraries needed to run the codes, the installations to be made, and the data to be downloaded. In the Training and Evaluation sections, there should be guidelines for the training and evaluation of models. In the Pre-trained Models section, there should be necessary information to allow access to the models trained by the researchers. The

Results section should display the research results through tables or figures and provide the pieces of code that can produce these outputs.

Despite its highly generalized nature, this template fundamentally comprises elements that are indispensable in any machine learning project. Providing these basic components within a project is deemed adequate to ensure its reproducibility.

In this study, we propose a comprehensive framework that hinges on assessing reproducibility based on how well research readme documents adhere to this template.

4.1. System Workflow

Our system is structured around three essential components: Readme parsing, Readme processing, and reproducibility scoring. Figure 4.1 illustrates the system workflow. The flow initiates by receiving a GitHub link as input, and proceeds through the following stages:

4.1.1. Readme Parsing

Our first step towards appropriately segmenting Readme files with a parser we developed. Given the complexity associated with processing the markdown format, we initiated the process by converting these files into HTML format. Markdown's hierarchical and component structure can be equivalently depicted in HTML. It's crucial to mention that GitHub employs a unique flavored markdown format (MacFarlane, 2019). Consequently, in order to process Readme files sourced from GitHub, our strategy necessitated the use of a library distinct from standard ones (Flowers, 2018).

Our developed parser operates based on two auxiliary parameters:

1. **Transform**: This parameter is employed when there is a need to transform elements like links, code snippets, images, tables, and citation elements within readme files into tag-like components, as outlined in Table 4.2.

2. **Group by Parent**: This parameter comes into play when sections of a readme file are to be aggregated according to their parent section. In such a scenario, every parent header morphs into the header of the grouped section, and its corresponding subsections within the hierarchical structure are treated as content.

Element	**Transformation**
Link with Text	<LINK text={Text}>
Link	<LINK>
Code Snippet	<CODE>
Image with Text	<IMAGE alt={Text}>
Image	<IMAGE>
Table with Caption	<TABLE caption={Caption}>
Table	<TABLE>
Citation	<CITE>

Table 4.2. Readme element transformations.

4.1.1.1. Readme Sections

Markdown-formatted files, such as Readme files, inherently support the establishment of a hierarchical structure. This format allows headers to be arranged at various levels, emphasizing the importance of proper structuring for effective content communication to the reader.

We denote each parent header, header, and content triplet as a "**section**" in this study. We perceive sections as the smallest self-contained units of meaning. Our analyses will be carried out based on these Readme sections. We aim to investigate the impact of section components and the hierarchical structure on system performance and seek to establish their optimal usage.

4.1.2. Readme Processing

The primary component of our system is the evaluation phase for readme files. In this stage, parsed and preprocessed sections are assessed using two different methodologies. The first is a section classification-based system that involves individually classifying each section and generating respective classification scores. The second methodology utilizes a readme classification-based system, which takes into account the hierarchical structure of the readme and evaluates it as a whole.

The two separate models explained in further detail below, are fine-tuned and utilized using the data outlined in our data section.

4.1.2.1. Section Classification

The evaluation process of this workflow involves individually classifying each section of a readme, followed by an assessment of the readme using classification scores that correlate with checklist coverage as shown in the dotted box in Figure 4.1. The primary objective is to assess how well-predetermined standards are adhered to, such as compliance with a chosen readme template. Additionally, the results provide insights into areas of potential improvement or deficiency.

In this workflow, we utilize the 'bert-base-uncased' model (Hugging Face, Inc., 2019) developed by Devlin et al. (2019), as our classifier. This BERT model is suitable for fine-tuning, making it adaptable for specific tasks.

4.1.2.1..1 Reproducibility Score Generation

Two slightly different formulas have been developed for computing the reproducibility score for our classification-based system, which primarily aims to evaluate the

classification efficacy of the sections.

The hierarchical system's scores are not involved in any further computations since they already directly convey the reproducibility score. As a standardization step, the values are divided by six, constraining them within the 0 to 1 range to represent reproducible or not reproducible.

4.1.2.1..1.1 Base Formulation

$$\mathrm{R}(C_S, R_c) = \frac{\sum_{S_a \in C_S} (\max (S_a))}{\mathrm{len}\,(R_c)} \tag{4.1}$$

The reproducibility score is determined by the formula (4.1). To create a list of classification scores for each checklist element ($\mathrm{C_S}$), we group the assigned classification scores from each section based on its corresponding reproducibility checklist ($\mathrm{R_c}$) elements. The process continues with extracting the maximum score from the classification scores array ($\mathrm{S_a}$) for every checklist element, followed by summing up these maximum values. The resulting summation value is then divided by the length of the entire checklist to obtain a single numerical outcome ranging between 0 and 1 as an indicator of reproducibility. The reason for taking the maximum score for the same sections here is to ensure the scoring is resilient against classification errors and to minimize the impact of content structure, which is challenging to distinguish automatically.

4.1.2.1..1.2 Consecutive Formulation

Algorithm 1 Consecutive Mean Algorithm

Require: List of pairs, L, where each pair contains a class and a score.

Ensure: A dictionary, D, where each key is a class, and each value is a list of means of consecutive scores of that class.

// Variables:

R: A dictionary to store results. Each key is a class, and each value is a list of means of sequential scores of that class.

P: The class of the previous pair in the list L.

T: A temporary list to store the scores of consecutive pairs with the same class.

(C, S): Class and its classification score.

$R \leftarrow \{\}$

$P \leftarrow L[0][0]$

$T \leftarrow []$

for each (C, S) in L **do**

  if $C = P$ **then**

    Append S to T

  else

    Append $mean(T)$ to $R[P]$

    $T \leftarrow [S]$

  end if

  $P \leftarrow C$

end for

Append $mean(T)$ to $R[P]$

return R

We devised a second formula to include the impact of the readme content's hierarchical structure in our evaluation. Following classification, the scores of consecutive sections bearing the same label are averaged as shown in algorithm 1 and subsequently incorporated into the formula (4.1). In this way, it serves as a self-penalty system against faulty document structuring.

4.1.2.1..1.3 Score Calculation Example

Table 4.3 provides the results of an example section classification for an imaginary readme composed of a total of seven sections. According to the results, five elements of the checklist are found within the readme. A section related to pre-trained models could not be found. There are also two instances in each of the Training and Requirements sections, with the Training sections given consecutively.

Section	Label	Classification Score
1	Introduction	0.75
2	Requirements	0.98
3	Training	0.95
4	Training	0.80
5	Evaluation	0.64
6	Requirements	0.87
7	Results	0.55

Table 4.3. Example of Section Classification.

Using the base scoring method and the formula in 4.1, a sample calculation would look like this; ($Introduction(0.75)$+$Pretrained\ Models(0)$+$Max\ of\ Requirements(0.98)$+ $Max\ of\ Training(0.95)$ + $Evaluation(0.64)$ + $Results(0.55)$) / 6 = **0.645**. Thus, the reproducibility score is calculated.

With consecutive scoring, only the average of the training sections will be taken,

and the calculation will be as follows; ($Introduction(0.75)$ + $Pretrained\ Models(0)$ + $Max\ of\ Requirements(0.98)$ + $Mean\ of\ Training(0.875)$ + $Evaluation(0.64)$ + $Results(0.55)$) / 6 = **0.6325**.

4.1.2.2. Readme Classification (Hierarchical Transformers)

Our goal was to develop an alternative system design that, instead of classifying each section of the readme individually and generating scores with manual formulations, takes the entire readme document as an input and outputs a reproducibility score as shown in the dashed box in Figure 4.1, while taking into account the hierarchical structure of the readme content. For this purpose, similar to Deng et al. (2020)'s work, which suggests document scoring using the relational hierarchical structure among words, sentences, and paragraphs in documents, and Ormerod et al. (2021)'s usage of transformers for essay scoring, we too developed a system that utilizes the hierarchical structure of readme content to generate scores as output.

The basic idea behind hierarchical networks is to encode documents hierarchically by providing context to word representations within each sentence and then integrating these sentence-level representations across multiple sentences (Chalkidis et al., 2022).

To achieve this, we employed the Hierarchical Attention Transformers (HAT) model (Hugging Face, Inc., 2022), as proposed within the scope of Chalkidis et al. (2022)'s work. This model is capable of classifying long documents, producing a score ranging from 0 to 6 for the readme content provided as input. We offer this model as an alternative system to our section classification-based system. However, it is crucial to note that, fundamentally, this model operates as a classification model that classifies the readme in its entirety.

The main advantage of this system over the section classification-based system is its simplicity - it's a one-step process where you input the readme and receive the score. However, this system falls short in terms of evaluative capability. For instance, while the classification-based system can analyze which sections are missing or insufficient based on classification results, the hierarchical model does not offer such insights.

4.2. Data

4.2.1. The Association of Computer Linguistics (ACL) Papers Dataset

The Association for Computational Linguistics (ACL) is known for organizing various events about natural language processing (NLP) and promoting the open-science. Within this institutional framework, 14 different events (see 1.1.) are actively organized. All articles are made freely available on their website, and for those articles that include source code, the code can also be easily accessed (The Association for Computational Linguistics, 2023a).

To facilitate data collection from this website, we developed a web crawler library in Python (Akdeniz, 2022). With this library, we can easily access the desired articles using different filters such as event, year, and paper type.

We gathered all articles published between 2013 and 2022 under the umbrella of these 14 events, amounting to a total of 47,117 articles. Out of these, we found 9,300 articles that contained links to source codes. As shown in Table 2.1, GitHub was the most frequently used platform. To streamline the process, we filtered out articles that utilized platforms other than GitHub. This resulted in a final set of 7,460 accessible articles that contained readme files. We then collected these readme files, creating a dataset that includes the Readme files of all 7,460 articles.

We parsed and segmented the 7460 research Readme files we gathered (see 4.1.1.1.) using the developed parser. A frequency analysis performed on section headers facilitated the creation of a list of perceived insignificant words (see 1.2.), and sections whose headers contained these words were subsequently cleaned. Following that, the remaining sections were further cleaned, ensuring they only comprised Latin letters, numbers, and punctuation.

The transformation operations, as indicated in Table 4.2, were executed to prevent the specified data from influencing system performance. For components like links, images, and tables that may contain distinctive text features (e.g., Caption, Alt Text), we ensured this information was preserved during the transformation process.

To evaluate the significance of the structural organization and semantic characteristics of Readme files, we saved the sections in six distinctive forms: 1) header, 2) content, 3) parent header + header, 4) header + content, 5) parent header + header + content, and 6) sections grouped by parent headers. As a result, we gathered a total of 50475 individual sections and 32233 grouped sections. The mean word length for these sections can be found in Table 4.4.

		Section
Parsing Type	**Count**	**Avg. Word Count**
Plain	50475	87
Transformed	50475	64
Grouped Plain	33233	135
Grouped Transformed	33233	98

Table 4.4. Parsed section statistics.

4.2.2. Section Labeling

The collected sections need to be labeled for training the deep learning models that will be deployed in our system. These labels correspond to the section headers detailed in the ground truth template (see Table 4.1). These headers, excluding the introduction, constitute the "ML Reproducibility Checklist" (Paperswithcode, 2020). We include the introduction, a section found in the readme template, into this checklist, thus utilizing a 6-item checklist both for section labeling and as an assessment reference.

Manually labeling these sections is both labor-intensive and can potentially introduce subjectivity. Hence, we applied two distinct automatic labeling methodologies - text similarity and zero-shot classification - to the gathered readme sections. The precision of these labels was then cross-verified with a subset that had been manually labeled by human annotators.

4.2.2.1. Text Similarity

The procedure adopted for labeling by text similarity relies on gauging the resemblances between the collected sections and those within the readme template, attributing the label of the section that exhibits the highest similarity.

This labeling approach uses a methodology similar to those in the studies of Pham et al. (2016); Chakrabarty (2022). It involves generating embeddings of the texts (refer Section 3.1.), then estimating the similarity between these embeddings, represented in vector format, using cosine similarity.

We utilized the "all-mpnet-base-v2" (Hugging Face, Inc., 2021) model, a fine-tuned variant of the MPNet model developed by Song et al. (2020), to create section embeddings. This model is integrated into the system via the sentence-transformers library (see Section 3.2.2.), outputting the embeddings of the provided input text.

Table 4.5 presents the mean similarity scores attained for each distinct content group. As per these results, the triplet of parent, header, and content yielded the highest rate of similarity.

It's expected that this approach would yield more accurate results when analyzing the compliance of the content at hand with predefined standards, such as our template. However, it may pose limitations concerning generalizability.

4.2.2.2. Zero-shot Classification

In this labeling method, in line with the approaches of Bujel et al. (2021); Rondinelli et al. (2022), we used zero-shot classification (refer to Section 3.4.) to label the collected sections.

We employed a fine-tuned version (Hugging Face, Inc., 2020) of the model initially created by Lewis et al. (2020). This process involved supplying the model with the checklist elements as labels, followed by feeding the readme sections as input, resulting in the generation of output labels.

Table 4.5 illustrates the average classification scores procured for each distinct group of content. These results indicate that the combination of parent, header, and content achieved the highest similarity rate, consistent with text similarity.

Differing from the text similarity method, this method may enable a broader scope of application and adaptability to a range of standards. For instance, a project could contain a unique section outlining specific project requirements, diverging from the standard 'requirements' section in the template. In such scenarios, text similarity could compute the semantic resemblance, but the similarity rate could be low, potentially influencing the performance. More meaningful outcomes can be achieved with zero-shot classification in these circumstances.

4.2.2.3. Manual Annotation

To verify the validity and precision of auto-labeled data, we handpicked a subset from our entire dataset and had it annotated by a team of three human coders. This subset consists of 1050 instances, with an equivalent number originating from every section. Every section in this subset was manually assigned labels drawn from our checklist.

4.2.2.3..1 Inter-annotator Agreement

We employ the agreement measure to validate automated labeling. We initially measured the agreement score of the three labelers using Weighted Cohen's Kappa (Cohen, 1968), with the score increasing from 0.45 to 0.59 after the removal of multiple labels and the selection of common ones. By incorporating data that received the same label from a minimum of two individuals, we secured full concurrence across all 731 data points. This data serves to verify the automated labels.

We measured agreement scores using sections agreed upon by human annotators, taken from twelve distinct datasets labeled in six different ways by two models. As depicted

	Zero Shot		Text Similarity	
Section Content	**Agreement**	**Classification Score (Avg.)**	**Agreement**	**Similarity Score (Avg.)**
Header	0.314	0.638	**0.345**	0.275
Parent + Header	0.332	0.653	0.335	0.291
Content	0.236	0.661	0.227	0.368
Header + Content	0.338	0.686	0.295	0.382
Parent + Header + Content	**0.341**	0.**700**	0.300	**0.387**
Grouped	-	0.678	-	0.384

Table 4.5. Automatic labeling performance of different section contents.

in Table 4.5, the zero-shot labeling method achieved the highest agreement with the parent, header, and content trio. On the other hand, the text similarity labeling method recorded the highest agreement when only the header was present.

We excluded the agreement rates for the sections grouped by parent header, as they encompass extended text segments and information related to multiple labels, making their inclusion unnecessary and irrelevant.

In conclusion, zero-shot labeling outperforms with broader content, while text similarity performs better with less content. On average, text similarity exhibited an agreement score of 0.30. However, zero-shot labeling slightly outperformed it with an average agreement score of 0.312, thus demonstrating higher overall success.

4.2.2.4. Training Data for Hierarchical Transformers

To train the hierarchical model, we utilized datasets that were automatically labeled via text similarity labeling, which had the highest agreement rate with human annotators. The labeled sections were grouped based on each readme, and training data was prepared by assigning a class between 0 and 6 to represent the reproducibility score, after determining checklist coverages for each readme.

CHAPTER 5

EXPERIMENTS

In this chapter, the details concerning the experimental results of the developed system, the measurement metrics utilized, and the employed data are presented.

5.1. NeurIPS Papers

This dataset encompasses the top 100 papers with the maximum star ratings among those published at NeurIPS in 2019 and evaluated for reproducibility in the Paperswithcode (2020) report. These readme files associated with these studies underwent manual analysis, and the checklist coverages were determined(Table 5.1) to use in the evaluation of our proposed system.

Repository	Stars	Introduction	Requirements	Pre-trained Models	Evaluation	Training	Results
deepmind/lab	6082	True	True	False	True	True	False
zihangdai/xlnet	5114	True	False	True	True	True	True
tensorflow/tpu	3120	True	False	False	False	False	False
tensorflow/lingvo	1914	True	True	False	False	True	False
facebookresearch/XLM	1913	True	True	True	True	True	True

Table 5.1. Example of NeurIPS Dataset.

5.2. Evaluation Metrics

To gauge the effectiveness of our system, we utilized the manually evaluated NeurIPS dataset (Section 5.1.). The results were then evaluated using three different metrics: 1) correlation 2) agreement, and 3) accuracy. All reported values derive from the validation of system-generated scores against those assigned by human evaluators. It is important to note that since both agreement and accuracy rates are calculated based only on classifications made by the system, their consecutive calculations do not impact these figures. Moreover, the section classification-based model's results and the readme classification model's outcomes are represented by these two metrics. Overall system's scoring performance can be most effectively represented through correlation values while classification performance information may be gleaned from either of the two other measures mentioned above.

5.3. Results

The system was evaluated using two different workflows we propose. A detailed analysis was conducted for the section classification-based model. As for the readme classification-based model, only its comparative results with the other model are provided.

5.3.1. Classification-Based System

We evaluated a factorial design of modeling choices: Labeling method, data content, and scoring type, ending in 24 in different ways. As depicted in Table 5.2, the results show that the system with the consecutive scoring using the classification model trained on grouped sections that are labeled with the zero-shot method gives the highest correlation (*0.661*) and agreement (*0.648*) value. This combination is still good performing for accuracy, where the best-performing choice becomes parent+header+content input

Labeling Method	Labeling Content	Scoring Type	Correlation	Agreement	Accuracy
Text Sim.	Content	Base	0.549	0.521	0.665
		Consecutive	0.554	0.521	0.665
	Grouped	Base	0.579	0.542	0.697
		Consecutive	0.581	0.542	0.697
	Header + Content	Base	0.578	0.523	0.685
		Consecutive	0.571	0.523	0.685
	Parent + Header + Content	Base	0.568	0.528	0.692
		Consecutive	0.569	0.528	0.692
	Parent + Header	Base	0.602	0.613	0.668
		Consecutive	0.597	0.613	0.668
	Header	Base	0.497	0.479	0.637
		Consecutive	0.473	0.479	0.637
Zero-Shot	Content	Base	0.582	0.563	0.662
		Consecutive	0.586	0.563	0.662
	Grouped	Base	0.651	**0.648**	0.697
		Consecutive	**0.661**	**0.648**	0.697
	Header + Content	Base	0.631	0.631	0.665
		Consecutive	0.626	0.631	0.665
	Parent + Header + Content	Base	0.617	0.556	**0.698**
		Consecutive	0.624	0.556	**0.698**
	Parent + Header	Base	0.594	0.540	0.608
		Consecutive	0.587	0.540	0.608
	Header	Base	0.399	0.419	0.587
		Consecutive	0.383	0.419	0.587

Table 5.2. Section Classification-Based System's Evaluation Results

instead of grouped sections. On the other hand, the least successful system in terms of correlation and accuracy rates was the one with consecutive scoring using the classification model trained on the section headers labeled by zero-shot methods. The combination of text similarity and header+content gives the lowest agreement score.

Method	Corr.	Agr.	Acc.
Text Sim.	0.478	0.443	**0.643**
Zero-shot	**0.486**	**0.469**	0.632

Table 5.3. System Evaluation Results Based on Labeling Method of Training Data.

In terms of labeling methods as presented in Table 5.3, models trained with data labeled via the zero-shot method emerged as the most successful on average, in terms of

Labeling Section Content	**Corr.**	**Agr.**	**Acc.**
Content	0.485	0.448	0.635
Grouped	**0.508**	**0.493**	0.664
Header	0.404	0.401	0.600
Header + Content	0.498	0.475	0.645
Parent + Header	0.503	0.471	0.614
Parent + Header + Content	0.494	0.450	**0.667**

Table 5.4. System Evaluation Results Based on Labeling Content of Training Data.

Type	**Corr.**	**Agr.**	**Acc.**
Base	**0.571**	**0.547**	**0.663**
Consecutive	0.568	"	"

Table 5.5. System Evaluation Results Based on Scoring Types.

both correlation and accuracy. However, the text similarity method surpassed zero-shot when evaluated solely on accuracy.

Upon assessing the average system performance relative to the content of the training data as depicted in Table 5.4, we find that the grouped data produced the most favorable results in terms of both correlations (*0.508*) and agreement *(0.493*). In the domain of accuracy, the combination of parent+header+content yielded the peak score (*0.667*), with the grouped data trailing just slightly behind. Conversely, the system trained on header-labeled data delivered the lowest scores across all three metrics, rendering it the least successful.

Examining the results from the perspectives of scoring and readme parsing methods, as presented in Table 5.5, the most successful average results were achieved when sections were parsed individually, and base scoring was applied. Conversely, on average, the least successful strategy involves grouped parsing and consecutive scoring.

When we evaluate the general impact of labeled data content on success rates, we observe that texts encompassing more content yield higher performance. This finding empowers the insufficiency of evaluating reproducibility solely based on headers. Additionally, both individually and on average, models trained with grouped data have proven to be more successful. Considering that grouped data contains more than one section together, we can conclude that classification models can learn better with more content.

Interestingly, when examining readme parsing methods, cases, where sections were

System	Corr.	Agr.	Acc.
Classification	**0.568**	**0.528**	**0.692**
Hierarchical	0.495	0.404	0.300

Table 5.6. System Evaluation Results Based on Evaluation Model.

treated individually were more successful than when they were grouped. This suggests that a primary section that contains multiple subsections pertaining to different labels can result in a decrease in the reproducibility score. So, this outcome underlines the significance of the hierarchical structure of readme files. In short, the most successful combination comprised models trained with comprehensive data classifying less extensive data.

Furthermore, even though the differences are slight, consecutive calculations have reduced the reproducibility scores. Given the minuscule difference, we can infer that there are not many consecutive sections on the same topic and that readme structures are well-formed. As this calculation method reduces the score by averaging consecutive sections, it also operates as an inherent penalizing mechanism, which is beneficial.

5.3.2. Hierarchical Transformers

In hierarchical transformer training, we utilized the data automatically labeled by text-similarity with the highest human agreement rate (Table 4.5). The ground-truth labels range from 0 to 6, indicating the number of checklist sections in a readme.

Table 5.6 delineates a comparative analysis of performance between the classification model and the hierarchical model, both trained on identical datasets. In all assessments, the section classification-based system demonstrated superior performance. However, the difference in correlation, which is the primary metric used to evaluate performance, was only 15%. This difference, particularly when contrasting a holistic approach like the hierarchical method, which assesses the entire readme at once, with an approach that separately evaluates each section, holds substantial promise for future research.

See Appendix 2.1.1. for runtime details of processes.

CHAPTER 6

WEB APPLICATION

The necessary software developments to make the designed system ready for use have been implemented, and it has been made available as an accessible application[1]. In this section, information about this application and software details is provided.

The software components are as follows:

- Streamlit: It is a library that enables the creation of web interfaces using Python. Additionally, it provides a free hosting service on its servers. System developments have been made using this library.
- HuggingFace: It offers a free model-sharing service. The trained models have been uploaded to this platform.

6.1. Application Workflow

Figure 6.1. System Screen after URL Input

[1]https://repro-der.streamlit.app/

Upon entering the website, the user inputs the GitHub URL of the code repository they want to assess for reproducibility. The system queries the provided URL and searches for an accessible Readme file. If it finds one, the system displays the Repository name and the number of stars, as shown in Figure 6.1. On the other hand, if an accessible Readme file is not found, or if an inappropriate GitHub URL is entered, a warning message will be displayed.

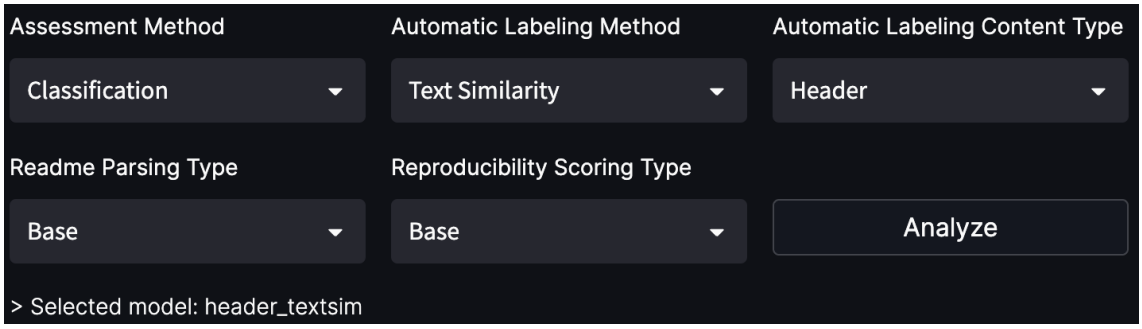

Figure 6.2. System Parameters

If an appropriate Readme is found, the system will display the system parameters as shown in Figure 6.2. The selected parameters will determine the model to be used. The selectable parameters are as follows:

- Assessment Method: Represents the selection of models from the two different flows indicated in Figure 4.1: Classification-based or Hierarchical-based. The Hierarchical-based model does not take any other parameters. The other parameters are available when the Classification-based system is selected.
- Automatic Labeling Method: Represents the methods used for automatic data labeling. It allows the user to choose which method will be used to label the data for training the model.
- Automatic Labeling Content Type: Represents the content type used during automatic labeling, determining which content-labeled data will be used to train the model.

- Readme Parsing Type: Represents the parsing method for the Readme. It allows the user to select whether sections will be taken individually or grouped under their parent header.
- Reproducibility Scoring Type: Represents two different scoring methods for reproducibility. The user can choose which method will be used to calculate the reproducibility score.

After selecting the parameters, the user presses the "Analyze" button. The system processes the Readme using the chosen model and calculates the reproducibility score. The result is then shown to the user, as illustrated in Figure 6.3.

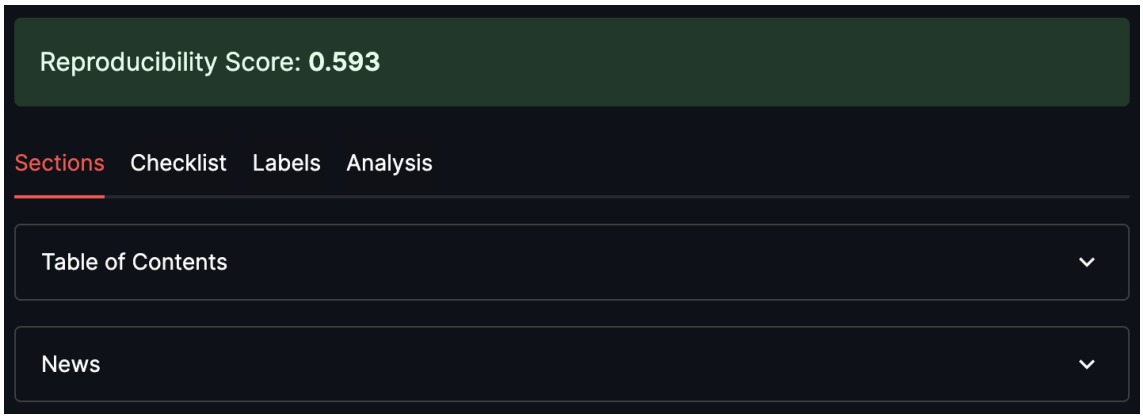

Figure 6.3. Reproducibility Score Output

If a classification-based model is selected, in addition to the reproducibility score, the user will be presented with detailed analyses in four separate panels:

1. Sections: In this panel, the table of contents for the Readme is generated, and each parsed section is displayed. By clicking on the sections, the user can access the processed content along with classification details.

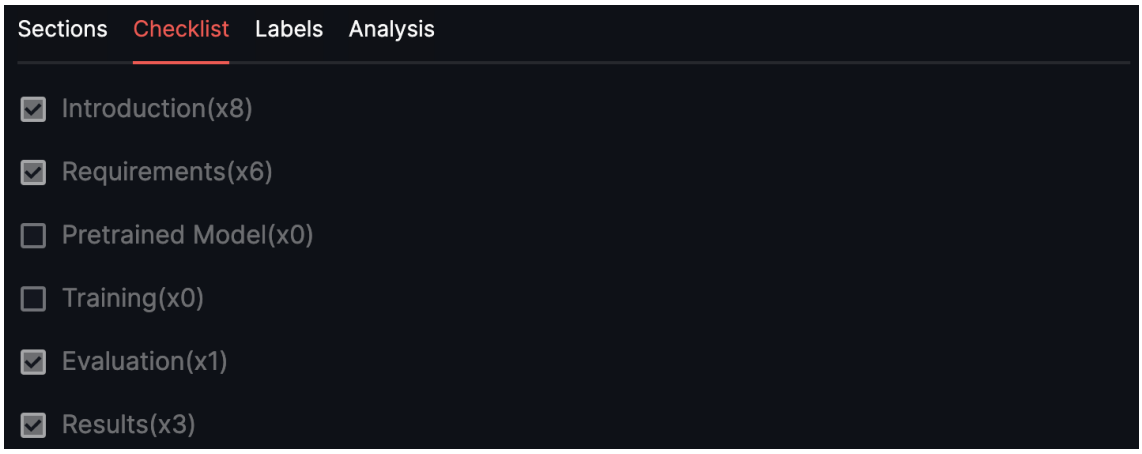

Figure 6.4. Checklist Tab

2. Checklist: This panel displays the reproducibility checklist results for the Readme (see Figure 6.4). It also provides information on how many sections correspond to each checklist item.

3. Labels: In this panel, sections related to each checklist item are shown in grouped form.

4. Analysis: In this panel, details related to the Readme are provided, as shown in Figure 6.5.

When the hierarchical model is selected, the entire Readme is processed as a whole, and no specific operations are conducted on individual sections. Consequently, only the reproducibility score is generated, which limits the application's ability to provide detailed analyses compared to the classification-based system.

Overall, this application serves as a valuable tool for researchers to assess the reproducibility of code repositories and facilitates the identification of any areas that need improvement.

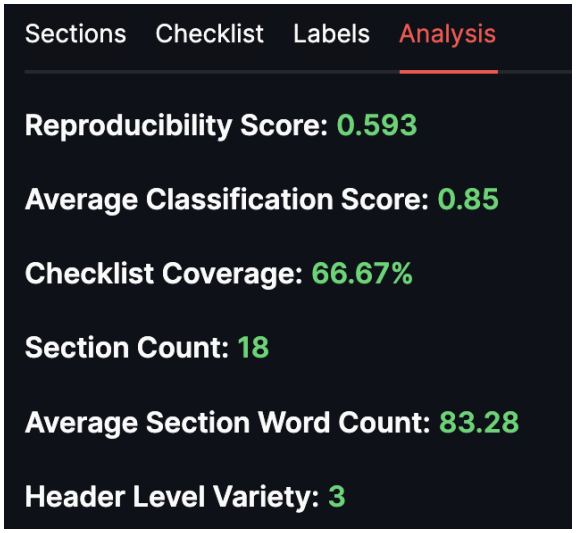

Figure 6.5. Analysis Tab

CHAPTER 7

CONCLUSION & FUTURE WORK

This study introduces an automated end-to-end system designed for evaluating readme files within source code repositories, utilizing a machine learning reproducibility checklist template as the guiding framework.

The models used in the system took into account the structural features of the readme by employing different data combinations during their training phase. Consequently, insights concerning the importance of document content were derived. Additionally, the data were labeled using two distinct automated labeling methods, demonstrating that compliance with specific standards can be assured and that more general models also yield successful results.

Two distinct workflows were designed in the system. One generates scores through a manual function depending on classification performance and presents both quantitative and qualitative analysis results. On the other hand, by employing hierarchical models and considering the entire document's hierarchical structure, automatic score generation was achieved in a single step. According to the results, scores provided through the classification performance of readme sections proved to be more successful, although hierarchical models also show promising outcomes.

Our results suggest that the presented framework has the potential to contribute positively to reproducibility efforts, where a common academic consensus remains elusive (Belz et al., 2021). Reviewers and researchers may utilize our developed tool to obtain valuable feedback regarding the reproducibility of their projects through readme files.

In future studies, research could be conducted to improve the performance of hierarchical models and to qualitatively evaluate their outputs. Simultaneously, the efficacy of classification models can be augmented by filtering automatically labeled data based on their labeling scores.

We acknowledge that the current methods of reviewing research papers are largely

manual, which can be both time-consuming and potentially subjective. Implementing automation or creating supportive tools could help to make these evaluations more objective and efficient and enhance research and its reproducibility.(Yuan et al., 2021).

Additionally, an automated approach to evaluating shared codes could facilitate a more holistic examination of studies, from inception to completion. Streamlining the process of code evaluation would likely enhance reproducibility, providing a boost to the pace of scientific advancements (Trisovic et al., 2022).

In conclusion, the development or use of tools capable of automating reproducibility assessments as supportive layers will directly affect the assessment and achievement of reproducibility, and it can streamline the processes and reduce the time spent.

REFERENCES

Akdeniz, K. (2022, November). ACL Anthology Scraper. https://github.com/kaanakdeniz/acl_anthology_scraper. Accessed on 2023-06-17.

Akdeniz, K. (2023a). Reproder - reproducibility assessment tool. https://repro-der.streamlit.app/. Accessed on 2023-06-28.

Akdeniz, K. (2023b). Source codes of reproducibility assessment of papers with source code. https://anonymous.4open.science/r/reproducibility_assessment. Accessed on 2023-06-28.

Albertoni, R., S. Colantonio, P. Skrzypczyński, and J. Stefanowski (2023, February). Reproducibility of Machine Learning: Terminology, Recommendations and Open Issues.

Almeida, F. and G. Xexéo (2023, May). Word Embeddings: A Survey.

Arvan, M., L. Pina, and N. Parde (2022, December). Reproducibility in computational linguistics: Is source code enough? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 2350–2361. Association for Computational Linguistics.

arXiv (2023). arXiv. https://arxiv.org/. Accessed on 2023-06-12.

Association for Computing Machinery (2016). Artifact Review and Badging.

Baker, M. (2016, May). 1,500 scientists lift the lid on reproducibility. *Nature 533*(7604), 452–454.

Barba, L. A. (2018, February). Terminologies for reproducible research.

Beltagy, I., M. E. Peters, and A. Cohan (2020, December). Longformer: The Long-Document Transformer.

Belz, A. (2021, September). Quantifying Reproducibility in NLP and ML.

Belz, A., S. Agarwal, A. Shimorina, and E. Reiter (2021). A Systematic Review of Reproducibility Research in Natural Language Processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online, pp. 381–393. Association for Computational Linguistics.

Belz, A., M. Popović, and S. Mille (2022, April). Quantified Reproducibility Assessment of NLP Results.

Bengio, Y., R. Ducharme, and P. Vincent (2000). A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems*, Volume 13. MIT Press.

Bujel, K., H. Yannakoudakis, and M. Rei (2021). Zero-shot Sequence Labeling for Transformer-based Sentence Classifiers. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, Online, pp. 195–205. Association for Computational Linguistics.

Chakrabarty, A. A. (2022, April). Text Data Labelling using Transformer based Sentence Embeddings and Text Similarity for Text Classification. *International Journal on Natural Language Computing 11*(2), 1–8.

Chalkidis, I., X. Dai, M. Fergadiotis, P. Malakasiotis, and D. Elliott (2022, October). An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014, September). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin 70*(4), 213–220.

Committee on Reproducibility and Replicability in Science, Board on Behavioral, Cognitive, and Sensory Sciences, Committee on National Statistics, Division of Behavioral and Social Sciences and Education, Nuclear and Radiation Studies Board, Division on Earth and Life Studies, Board on Mathematical Sciences and Analytics, Committee on Applied and Theoretical Statistics, Division on Engineering and Physical Sciences, Board on Research Data and Information, Committee on Science, Engineering,

Medicine, and Public Policy, Policy and Global Affairs, and National Academies of Sciences, Engineering, and Medicine (2019, September). *Reproducibility and Replicability in Science*. Washington, D.C.: National Academies Press.

Crane, M. (2018, April). Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics 6*, 241–252.

Dai, Z., Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov (2019, June). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990, September). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Deng, Z., H. Peng, C. Xia, J. Li, L. He, and P. Yu (2020). Hierarchical Bi-Directional Self-Attention Networks for Paper Review Rating Recommendation. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online), pp. 6302–6314. International Committee on Computational Linguistics.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics.

Diaba-Nuhoho, P. and M. Amponsah-Offeh (2021, December). Reproducibility and research integrity: The role of scientists and institutions. *BMC Research Notes 14*(1), 451.

Dickinson, D., A. Raj GV, and G. Fung (2021, September). A Model for Zero-shot Text Multi-labeling Using Semantics-based Labels. In *2021 Third International Conference on Transdisciplinary AI (TransAI)*, pp. 147–154.

Easterbrook, S. M. (2014, November). Open code for open science? *Nature Geoscience 7*(11), 779–781.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science 14*(2), 179–211.

Flowers, S. (2018, March). Cmarkgfm - Python Bindings to GitHub's Cmark. https://github.com/theacodes/cmarkgfm. Accessed on 2023-06-17.

GitHub (2023). GitHub. https://github.com/. Accessed on 2023-06-12.

Goodman, S. N., D. Fanelli, and J. P. A. Ioannidis (2016, June). What does research reproducibility mean? *Science Translational Medicine 8*(341).

Gundersen, O. E. (2019, December). Standing on the Feet of Giants — Reproducibility in AI. *AI Magazine 40*(4), 9–23.

Gundersen, O. E. (2021, May). The fundamental principles of reproducibility. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 379*(2197), 20200210.

Gundersen, O. E., K. Coakley, C. Kirkpatrick, and Y. Gil (2023, April). Sources of Irreproducibility in Machine Learning: A Review.

Gundersen, O. E. and S. Kjensmo (2018, April). State of the Art: Reproducibility in Artificial Intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence 32*(1).

Harris, Z. S. (1954, August). Distributional Structure. *WORD 10*(2-3), 146–162.

Hochreiter, S. and J. Schmidhuber (1997, November). Long Short-Term Memory. *Neural Computation 9*(8), 1735–1780.

Hugging Face, Inc. (2019). Bert-base-uncased · Hugging Face.

Hugging Face, Inc. (2020). Facebook/bart-large-mnli · Hugging Face.

Hugging Face, Inc. (2021). Sentence-transformers/all-mpnet-base-v2 · Hugging Face.

Hugging Face, Inc. (2022, November). Kiddothe2b/hierarchical-transformer-base-4096 · Hugging Face.

Hutson, M. (2018, February). Artificial intelligence faces reproducibility crisis. *Science 359*(6377), 725–726.

Isdahl, R. and O. E. Gundersen (2019, September). Out-of-the-Box Reproducibility: A Survey of Machine Learning Platforms. In *2019 15th International Conference on eScience (eScience)*, San Diego, CA, USA, pp. 86–95. IEEE.

Joppa, L. N., G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'Hara, D. Gavaghan, and S. Emmott (2013, May). Troubling Trends in Scientific Software Use. *Science 340*(6134), 814–815.

Kirstie, J. (2016, May). Reproducible Research. https://github.com/WhitakerLab/ReproducibleResearch. Accessed on 2023-06-12.

Laurinavichyute, A., H. Yadav, and S. Vasishth (2022, August). Share the code, not just the data: A case study of the reproducibility of articles published in the Journal of Memory and Language under the open data policy. *Journal of Memory and Language 125*, 104332.

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 7871–7880. Association for Computational Linguistics.

Liu, J., J. Carlson, J. Pasek, B. Puchala, A. Rao, and H. V. Jagadish (2022, July). Promoting and Enabling Reproducible Data Science Through a Reproducibility Challenge. *Harvard Data Science Review*.

Lucic, A., M. Bleeker, S. Bhargav, J. Forde, K. Sinha, J. Dodge, S. Luccioni, and R. Stojnic (2022, May). Towards Reproducible Machine Learning Research in Natural Language Processing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, Dublin, Ireland, pp. 7–11. Association for Computational Linguistics.

MacFarlane, J. (2019, April). GitHub Flavored Markdown. https://github.github.com/gfm/. Accessed on 2023-06-17.

Mieskes, M., K. Fort, Sorbonne Université, EA STIH Paris, France, A. Névéol, Université Paris-Saclay, France, C. Grouin, Université Paris-Saclay, France, K. Cohen, and

Computational Bioscience Program, University of Colorado, USA (2019, October). NLP Community Perspectives on Replicability. In *Proceedings - Natural Language Processing in a Deep Learning World*, pp. 768–775. Incoma Ltd., Shoumen, Bulgaria.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013, September). Efficient Estimation of Word Representations in Vector Space.

Mondal, S. and B. Roy (2021, December). Reproducibility Challenges and Their Impacts on Technical Q&A Websites: The Practitioners' Perspectives.

Munafò, M. R., B. A. Nosek, D. V. M. Bishop, K. S. Button, C. D. Chambers, N. Percie Du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. P. A. Ioannidis (2017, January). A manifesto for reproducible science. *Nature Human Behaviour 1*(1), 0021.

Nature, C. S. (2021, October). Moving towards reproducible machine learning. *Nature Computational Science 1*(10), 629–630.

Nawrot, P., S. Tworkowski, M. Tyrolski, Ł. Kaiser, Y. Wu, C. Szegedy, and H. Michalewski (2022, April). Hierarchical Transformers Are More Efficient Language Models.

Nguyen, S. and V. Rampin (2022, November). Who Writes Scholarly Code? *International Journal of Digital Curation 17*(1), 18.

Norouzi, M., T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean (2014, March). Zero-Shot Learning by Convex Combination of Semantic Embeddings.

Nüst, D. and S. J. Eglen (2021, July). CODECHECK: An Open Science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility. *F1000Research 10*, 253.

Oates, B. J. (2006). *Researching Information Systems and Computing*. London ; Thousand Oaks, Calif: SAGE Publications.

Obels, P., D. Lakens, N. A. Coles, J. Gottfried, and S. A. Green (2019, May). Analysis of Open Data and Computational Reproducibility in Registered Reports in Psychology.

Ormerod, C. M., A. Malhotra, and A. Jafari (2021, February). Automated essay scoring using efficient transformer-based language models.

Papers with Code (2023). Papers with code. https://paperswithcode.com/. Accessed on 2023-06-12.

Paperswithcode (2020, March). Tips for Publishing Research Code. https://github.com/paperswithcode/releasing-research-code. Accessed on 2023-06-01.

Pedersen, T. (2008). Last Words: Empiricism Is Not a Matter of Faith. *Computational Linguistics 34*(3), 465–470.

Peng, R. D. (2011, December). Reproducible Research in Computational Science. *Science 334*(6060), 1226–1227.

Pennington, J., R. Socher, and C. Manning (2014, October). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1532–1543. Association for Computational Linguistics.

Pham, M., S. Alse, C. A. Knoblock, and P. Szekely (2016). Semantic Labeling: A Domain-Independent Approach. In P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil (Eds.), *The Semantic Web – ISWC 2016*, Lecture Notes in Computer Science, Cham, pp. 446–462. Springer International Publishing.

Pineau, J., P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and H. Larochelle (2020, December). Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program).

Piwowar, H., J. Priem, V. Larivière, J. P. Alperin, L. Matthias, B. Norlander, A. Farley, J. West, and S. Haustein (2018, February). The state of OA: A large-scale analysis of the prevalence and impact of Open Access articles. *PeerJ 6*, e4375.

Plesser, H. E. (2018, January). Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Frontiers in Neuroinformatics 11*, 76.

Pouchard, L., K. G. Reyes, and F. J. A. B.-J. Yoon (2023, May). A Rigorous Uncertainty-Aware Quantification Framework Is Essential for Reproducible and Replicable Machine Learning Workflows.

Reimers, N. (2019). SentenceTransformers. https://www.sbert.net/. Accessed on 2023-06-15.

Reimers, N. and I. Gurevych (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, pp. 3980–3990. Association for Computational Linguistics.

Rios, A. and R. Kavuluru (2018, October). Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 3132–3142. Association for Computational Linguistics.

Rondinelli, A., L. Bongiovanni, and V. Basile (2022, October). Zero-Shot Topic Labeling for Hazard Classification. *Information 13*(10), 444.

Salton, G., A. Wong, and C. S. Yang (1975, November). A vector space model for automatic indexing. *Communications of the ACM 18*(11), 613–620.

Sandve, G. K., A. Nekrutenko, J. Taylor, and E. Hovig (2013, October). Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology 9*(10), e1003285.

Song, K., X. Tan, T. Qin, J. Lu, and T.-Y. Liu (2020, November). MPNet: Masked and Permuted Pre-training for Language Understanding.

Stodden, V., J. Seiler, and Z. Ma (2018, March). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences 115*(11), 2584–2589.

Taschuk, M. and G. Wilson (2017, April). Ten simple rules for making research software more robust. *PLOS Computational Biology 13*(4), e1005412.

The Association for Computational Linguistics (2023a). ACL Anthology. https://aclanthology.org/. Accessed on 2023-06-17.

The Association for Computational Linguistics (2023b). Frequently Asked Questions - ACL Anthology. https://aclanthology.org/faq/. Accessed on 2023-06-13.

Trisovic, A., M. K. Lau, T. Pasquier, and M. Crosas (2022, February). A large-scale study on research code quality and execution. *Scientific Data 9*(1), 60.

Vandewalle, P. (2019, May). Code availability for image processing papers: A status update. *WIC IEEE SP Symposium on Information Theory and signal Processing in the Benelux*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017, December). Attention Is All You Need.

Wieling, M., J. Rawee, and G. Van Noord (2018, December). Reproducibility in Computational Linguistics: Are We Willing to Share? *Computational Linguistics 44*(4), 641–649.

Wilson, G., J. Bryan, K. Cranston, J. Kitzes, L. Nederbragt, and T. K. Teal (2016, October). Good Enough Practices in Scientific Computing.

Xian, Y., C. H. Lampert, B. Schiele, and Z. Akata (2019, September). Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence 41*(9), 2251–2265.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy (2016, June). Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, pp. 1480–1489. Association for Computational Linguistics.

Yuan, W., P. Liu, and G. Neubig (2021, January). Can We Automate Scientific Reviewing?

Zaheer, M., G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed (2021, January). Big Bird: Transformers for Longer Sequences.

Zhang, B. (2019, February). An Explorative Study of GitHub Repositories of AI Papers.

Zhao, X., S. Ouyang, Z. Yu, M. Wu, and L. Li (2023, May). Pre-trained Language Models Can be Fully Zero-Shot Learners.

APPENDIX A

METHODOLOGY

1.1. ACL Anthology Events

AACL, ACL, ANLP, CL, CoNLL, EACL, EMNLP, Findings, IWSLT, NAACL, SemEval, *SEM, TACL, WMT, WS

1.2. Headers of Dropped Sections

get involved, problems, question, disclaimer, issues, miscellaneous, misc, troubleshoot, reference, references, thoughts, abusive corpus, acknolwedgement, inquiries, changes, ethical guidelines, change logs, citation, cite, credit, contact, licence, acknowledgement, license, referense, contribution, contribute, contributing, author, changelog, faq, citing, news, table of contents, note, links, updates, contributor, todo, acknowledgment, leaderboard, structure, copyright, motivation, acknowledge, what new, bibtex

APPENDIX B

EXPERIMENTS

2.1. Training Results Based on Section Contents

	Zero Shot			Text Similarity		
Section Content	Training Loss	Validation Loss	Accuracy	Training Loss	Validation Loss	Accuracy
Header	0.60	0.83	0.71	0.54	0.74	0.72
Parent + Header	**0.52**	**0.78**	**0.74**	0.48	**0.65**	**0.77**
Content	0.64	0.83	0.71	0.51	0.73	0.73
Header + Content	0.61	0.85	0.70	0.50	0.75	0.72
Parent + Header + Content	0.56	0.80	0.72	**0.47**	0.71	0.74
Grouped	0.62	**0.78**	0.73	0.55	0.73	0.74

2.1.1. Runtime Information

Model training was done on Kaggle with T4x2 GPU. Other operations were done on a computer with GTX1650, 16GB, and i7-10750H specifications.

2.1.1.1. Training

Section Classification Model: $\sim$ 1 hour for 3 epochs. Hierarchical Transformers Model: $\sim$ 1.40 hours for 5 epochs.

2.1.1.2. Labeling

Zero-shot: $\sim$ 4 hours. Text similarity: $\sim$ 12 min.

2.1.1.3. System Evaluation

Section Classification Model: $\sim$ 30 sec. Hierarchical Transformers Model: $\sim$ 45 sec.

2.1.1.4. Readme Parsing

Base: $\sim$ 1 sec for 100 readme files. Grouped: $\sim$3 sec for 100 readme files.