ClassifAI: Automating Issue Reports Classification using Pre-Trained BERT (Bidirectional Encoder Representations from Transformers) Language Models

Khubaib Amjad Alam*
khubaib.amjad@nu.edu.pk
National University of Computer and Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan

Ashish Jumani
i200494@nu.edu.pk
National University of Computer and Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan

Harris Aamir
i200943@nu.edu.pk
National University of Computer and Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan

Muhammad Uzair
i202341@nu.edu.pk
National University of Computer and Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan

ABSTRACT

The utilization of Issue Tracking Systems by users to systematically manage and monitor issue reports within their repositories has become indispensable. An issue report encapsulates a wealth of software-related information, encompassing problem descriptions, requests for new features and inquiries about the software product, to name a few. As the volume of these issues escalates, manual management becomes increasingly challenging, prompting the exploration of automated approaches for more effective handling. This paper introduces **ClassifAI** [1], an automated Issue Report Categorization approach built on the foundation of the Transformer-based pre-trained RoBERTa-Large model. ClassifAI proficiently classifies issue reports into three primary categories: Bug report, Enhancement/feature request, and Question. The process involves cleaning and preprocessing data sets provided for the NLBSE'24 [7] tool competition, followed by fine-tuning the pre-trained RoBERTa model on the refined data set. The experimental evaluation of ClassifAI is performed on approximately 1500 issue reports belonging to five different projects. The results indicate that RoBERTa-Large fine tuned variant demonstrates an acceptable level of performance by achieving a 83.2% F1-score (micro average).

CCS CONCEPTS

• **Software and its engineering** → *Software libraries and repositories*; **Maintaining software**; • **Applied computing** → **Document management and text processing**;

KEYWORDS

Issue Reports, Natural Language Processing, Text Analysis, Machine Learning, Language Models, BERT

ACM Reference Format:
Khubaib Amjad Alam, Ashish Jumani, Harris Aamir, and Muhammad Uzair. 2024. ClassifAI: Automating Issue Reports Classification using Pre-Trained BERT (Bidirectional Encoder Representations from Transformers) Language Models. In *2024 ACM/IEEE International Workshop on NL-based Software Engineering (NLBSE '24), April 20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3643787.3648041

1 INTRODUCTION

Issue reports in software development process, serve as a critical conduit for users to communicate encountered problems or articulate requests for new features [1]. These reports find management and tracking through specialized systems known as Issue Tracking Systems (ITSs), offering a comprehensive suite of functionalities, including issue assignment to developers, progress monitoring, and issue prioritization [16]. The systematic classification of these issue reports emerges as an essential task, ensuring streamlined and effective management within the overall development process.

The classification of issue reports involves organizing them based on various criteria, such as issue type , severity, or priority level. A meticulous classification process guarantees that each issue report is directed to the appropriate individual or team; for instance, bug reports are routed to developers, while feature requests are directed to product managers [2]. Beyond its immediate benefits, the process of issue report classification carries deeper significance in the context of enhancing software quality. By identifying and categorizing recurring issues, developers gain valuable insights into user experiences [12]. This insight becomes pivotal in refining software design and implementation, fostering a culture of continual improvement.

In this tool paper, using data-driven approaches, we have fine-tuned a Transformer-based pre-trained RoBERTa-based model to predict the category of issue reports.

*All authors contributed equally to this research.
[1]https://github.com/HarrisAamir/Issue-Report-Classification-NLBSE-2024

2 RELATED WORK:

Kallis et al. [8],[9] initially introduced TicketTagger, a tool leveraging FastText for the classification of issues into three primary categories: Bug, Enhancement, and Question. These categories align with the default labels in the GitHub issue system. The model was trained on the textual content, encompassing both the title and description, extracted from 30,000 issue reports originating from approximately 12,000 GitHub repositories. CatIss [6] stands out as an automated issue report categorization tool, constructed on the foundation of the Transformer-based pre-trained RoBERTa model. This system proficiently classifies issue reports into three principal categories: Bug, Enhancement, and Question. The training of CatIss is conducted on the designated training set for the NLBSE tool competition and authors reported F1-score of 87.2% on the provided dataset. Notably, a subsequent development in this domain is the introduction of BEE by Song and Chaparro [3]. BEE builds upon the foundation laid by TicketTagger, utilizing its pre-trained model for issue labeling. This development highlights the continued exploration and integration of pre-existing models to further enhance the efficiency and accuracy of issue classification systems in the context of software development. When it comes to few shot learning G. Colavito et al. [3] proposed a supervised approach leveraging SETFIT, a framework tailored for few-shot learning, in conjunction with Sentence-BERT (SBERT), a variant of BERT designed for robust sentence embedding. They managed to evaluate the SETFIT model on the challenge test set, achieving F1-micro score .7767.

3 TOOL DESCRIPTION

ClassifAI aims at solving the non-Trivial problem of Issue report classification by using Bidirectional Encoder Representations from Transformers (BERT). This section elaborates the working methodology of the proposed approach. It aims to use and explore BERT based models and few shot-learning techniques. We experimented with a total of 5 distinct BERT-based model variations. This chapter is organized into three Sections denoting the Research Question, Text Preprocessing techniques used, and the selected model along with the process of training. Figure 1. highlights the flow of activities followed within ClassifAI.

3.1 Research Question

This study aims at adressing the follwoing primary Research Question: **What is the performance of the selected model on the test dataset of issue reports compared to generic Pre-Trained Language Models (PLMs)?**

This primary research question centers on determining the most capable PLM for automatic categorization of issue reports into three distinct classes: Bugs, Features, and Questions. This study considers the generic PLMs as they are trained on enormous corpora and have the potential to achieve much higher levels of accuracy. Therefore, domain specific variants of the BERT are not considered in the current research. Moreover, as the the data is already balanced, BERT variants designed to handle imbalanced data are not included in this study.

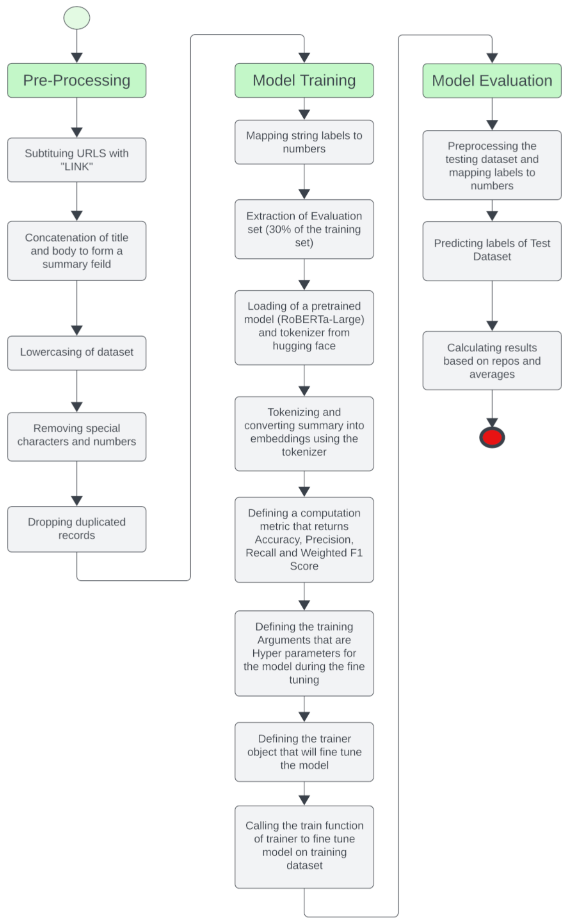
Pre-Processing
Subtituing URLS with "LINK"
Concatenation of title and body to form a summary feild
Lowercasing of dataset
Removing special characters and numbers
Dropping duplicated records
Model Training
Mapping string labels to numbers
Extraction of Evaluation set (30% of the training set)
Loading of a pretrained model (RoBERTa-Large) and tokenizer from hugging face
Tokenizing and converting summary into embeddings using the tokenizer
Defining a computation metric that returns Accuracy, Precision, Recall and Weighted F1 Score
Defining the training Arguments that are Hyper parameters for the model during the fine tuning
Defining the trainer object that will fine tune the model
Calling the train function of trainer to fine tune model on training dataset
Model Evaluation
Preprocessing the testing dataset and mapping labels to numbers
Predicting labels of Test Dataset
Calculating results based on repos and averages

Figure 1: step-wise procedure of ClassifAI

3.2 Text Pre-Processing

Text preprocessing encompasses the refinement of datasets and the transformation of unprocessed text to augment its quality and structure for machine learning models. This process involves eliminating special characters to mitigate noise, and normalizing the text by lowercasing it. Initially, the dataset was loaded into Python Pandas DataFrames [2]. A new column named "summary" was created, containing concatenated text from the "title" and "body" columns. Subsequently, all hyperlinks, special characters, blank spaces, and numerical values were eliminated from the dataset.

3.3 Model Training

In recent years, Transformer-based approaches have exerted a substantial influence, particularly, Language models like Bidirectional Encoder Representations from Transformers (BERT) offer the capability for refinement through the integration of additional output layers. Based on a thorough analysis of the existing body of knowledge, we considered to experiment with BERT, XLNet, DistilBERT, S-BERT and RoBERTa (Large and Base).

[2]https://pandas.pydata.org/

RoBERTa, a robust natural language processing model, is an extension of the BERT architecture, developed by Facebook AI[3]. With 125 million parameters, RoBERTa Large handles big datasets with ease, adjusts to peculiar language requirements, and is affordable for resource-constrained businesses. Thus we also experimented with this model.[10] Through previous work, we were able to see the good performance of XLNet, SBERT, DistilBERT, and Bert Large. Through XLNet's bidirectional analysis, deep context is unlocked, resulting in accurate categorization. Moreover, XLNet's scalable architecture handles massive datasets with ease thanks to its effective architecture. Because of its openness, which enables intelligent optimization.[15] Similarly, to tackle uncategorized issue reports, SBERT leverages domain adaptation and semantic understanding. Accurate and efficient categorization is ensured by precise language analysis and lightweight architecture, while optimization is facilitated and trust is built through transparency. [14] This clarifies the motivation behind our models' experimentation. Through our literature review, we were able to identify that DistilBERT is another possible option for effective issue report classification because of its small architecture and knowledge distillation features. [13] Lastly, the Google AI created potent pre-trained language model BERT Large is a great option for challenging tasks like issue report categorization because of its outstanding accuracy in reading and interpreting language [4].

We started to experiment on these models by trying to fine-tune them to the most appropriate hyperparameters. Finally, the model was trained and tested on the provided dataset and the results can be seen in Chapter 5 of this document.

4 EXPERIMENTAL SETUP

This section is divided into 3 subsections that explain the collection of datasets, the evaluation metrics used for finding and working out the results and most importantly our implementation and experimentation of the RoBERTa large model.

4.1 Data Collection

This research utilizes the Issue reports dataset provided by NLBSE'24 [7] tool competition. This dataset encompasses 3 thousand labeled issue reports (as bugs, enhancements, and questions) extracted from 5 real open-source projects. We leveraged this dataset to evaluate our proposed approach and compare the results against the provided baseline that is based on sentence transformers. Each issue report contains the following information: **Repository, Label, Title and Body**. Each issue is labeled with one class that indicates the issue type, namely, bug, feature, and question. Issues with multiple labels are excluded from the dataset. The dataset is then split into a training set (50%) and a test set (50%). To ensure accurate model output, it is essential to have a separate evaluation set, which is used during the training of the model. The initial training set was further divided in a ratio of 70:30 following the methodology mentioned by Ismail Muraina [11], resulting in a revised ratio of 35:15:50. This breakdown allocates 35% for training, 15% for evaluation, and 50% for testing. The evaluation set plays a crucial role during model training by computing the loss and adjusting weights accordingly.

[3]https://ai.meta.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/

Table 1: Experimentation using RoBERTa -Large

learning rate	num train epochs	train batch size	Accuracy	Weighted F1
5.00E-05	8	4	0.782112299	0.780638203
7.00E-06	32	4	0.785427807	0.781836024
7.00E-06	12	3	0.782085561	0.782531131
5.00E-06	16	8	0.760026738	0.758328035
5.00E-06	16	3	0.782085561	0.780923568
7.00E-06	**32**	**4**	**0.832443258**	**0.832210740**
5.00E-06	8	8	0.770053476	0.767823363
7.00E-06	8	2	0.77406417	0.77156813
7.00E-06	6	2	0.784090909	0.783340988
7.00E-06	4	2	0.780748663	0.780156645

The key terms TP, FP, TN, and FN represent the counts of True Positives, False Positives, True Negatives, and False Negatives, respectively, for each issue class. [5]

4.2 Evaluation Metrics

The evaluation of ClassifAI involves applying standard metrics commonly used for classifier assessment, including precision, recall, and F1-score. These meticulous evaluation metrics offer a detailed and comprehensive assessment of ClassifAI's performance, enabling insights into its effectiveness.

4.3 Experimentation and Implementation

We used HuggingFace [4] to derive a base RoBERTa [Large] model and then after fine-tuning the model to our pre-processed dataset, we started experimentation by tuning the model on different sets of hyperparameters. In our pursuit of identifying the optimal hyperparameters, we conducted a thorough experimentation by training the models with various hyperparameter configurations. Initially, we employed hyperparameters derived from previous studies that demonstrated superior results. Subsequently, in our quest to achieve a satisfactory score, we systematically explored different combinations of hyperparameters, closely monitoring the outcomes making sure that it is prevented from becoming either overfitted or underfitted. The table 1 shows the results of the model experimentation with different configurations.

It is worth mentioning that this process was repeated for several variations of BERT based models. However, RoBERTa-Large consistently exhibited relatively promising results.

5 RESULTS AND ANALYSIS

To address the defined research question, we conducted a thorough systematic literature review followed by rigorous experimentation, implementation and analysis to reach the following results. This study conisdered five mainstream PLMs including BERT, XLNet, DistilBERT, S-BERT and RoBERTa. The motivation behind consideration of generic category of BERT variants is based on the fact that these models trained on enormous corpora, have the potential

[4]https://huggingface.co/

to achieve much higher levels of accuracy which proved to be the solution in our case. As a result, it's critical to carefully consider the size of the dataset, the computational power at disposal, and the time restrictions for development. Both Base and Large variants of the RoBERTa were considered due to its consistently high performance reported in the extant literarure.

The primary research question aimed at investigating the performance of the selected model on the test dataset of issue reports compared to generic PLMs. The results of the comparative evaluation with mainstream BERT variants indicate that RoBERTa-Large outperforms other PLMs with a notable margin and has the potential to be used for training the classifier for the Issue report classification challenge. Table 2 shows the comparative results of the selected model on peculair set of configurations in comparison to other BERT-based variations. The classification performance obtained by the proposed approach across multiple repositories is shown in the Table 3.

Table 2: Comparison of different BERT based variations

Model	Learning rate	Epoch	Batch size	Accuracy	F1-Score
S-BERT	7.00E-05	5	16	0.7486	0.7480
DistilBERT	5.00E-06	16	4	0.7399	0.7376
BERT-Large	5.00E-06	20	8	0.7667	0.7635
XLNet	5.00E-05	20	4	0.7787	0.7792
RoBERTa [Base]	7.00E-06	20	16	0.7740	0.7728
RoBERTa-Large	7.00E-06	32	4	0.8324	0.8322

6 CONCLUSION

Issue report classification can effectively reduce the manual work of software engineers by automating the categorization of issue reports making it a lot easier and time-saving. ClassifAI categorizes issue reports into Bug, Enhancement and Question by using the pre-trained RoBERTa Large model trained on the dataset provided by NLBSE tool competition. With ClassifAI we were able to achieve an optimal level of performance yielding the final F1-avg score of 83.2%.

Table 3: Precision, Recall, F1 per repository and per label

Repository	Label	Precision	Recall	F1
facebook/react	bug	0.8319	0.99	0.9041
	feature	0.8454	0.82	0.8325
	question	0.9048	0.76	0.8261
	average	0.8607	0.8567	0.8542
tensorflow/tensorflow	bug	0.9091	0.9	0.9045
	feature	0.9091	0.9	0.9045
	question	0.8333	0.85	0.8416
	average	0.8838	0.8833	0.8835
microsoft/vscode	bug	0.7810	0.82	0.8
	feature	0.8384	0.83	0.8342
	question	0.8438	0.81	0.8265
	average	0.8210	0.82	0.8202
bitcoin/bitcoin	bug	0.8105	0.77	0.7897
	feature	0.8302	0.8889	0.8585
	question	0.7629	0.7475	0.7551
	average	0.8012	0.8021	0.8011
opencv/opencv	bug	0.7455	0.82	0.7810
	feature	0.8438	0.81	0.8265
	question	0.8191	0.77	0.7938
	average	0.8028	0.8	0.8004
overall	bug	0.8144	0.86	0.8366
	feature	0.8531	0.8497	0.8514
	question	0.8309	0.7876	0.8086
	average	0.8328	0.8324	0.8322

REFERENCES

[1] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. 2010. Communication, collaboration, and bugs: The social nature of issue tracking in software engineering. In *Proc. ACM Conf. Comput. Support. Coop. Work.*

[2] Heetae Cho, Seonah Lee, and Sungwon Kang. 2022. Classifying issue reports according to feature descriptions in a user manual based on a deep learning model. *Information and Software Technology* 142 (2022), 106743.

[3] Giuseppe Colavito, Filippo Lanubile, and Nicole Novielli. 2023. Few-Shot Learning for Issue Report Classification. In *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE).* IEEE, 16–19.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Mohammad Hossin and Md Nasir Sulaiman. 2015. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process* 5, 2 (2015), 1.

[6] Maliheh Izadi. 2022. Catiss: An intelligent tool for categorizing issues reports using transformers. In *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering.* 44–47.

[7] Rafael Kallis, Giuseppe Colavito, Ali Al-Kaswan, Luca Pascarella, Oscar Chaparro, and Pooja Rani. 2024. The NLBSE'24 Tool Competition. In *Proceedings of The 3rd International Workshop on Natural Language-based Software Engineering (NLBSE'24).*

[8] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2019. Ticket tagger: Machine learning driven issue classification. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME).* IEEE, 406–409.

[9] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting issue types on GitHub. *Science of Computer Programming* 205 (2021), 102598.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[11] Ismail Muraina. 2022. Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts. In *7th International Mardin Artuklu Scientific Research Conference.* 496–504.

[12] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. 2015. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE international conference on software maintenance and evolution (ICSME).* IEEE, 281–290.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[14] Bin Wang and C-C Jay Kuo. 2020. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 2146–2157.

[15] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[16] Yuxiang Zhu, Minxue Pan, Yu Pei, and Tian Zhang. 2019. A bug or a suggestion? an automatic way to label issues. *arXiv preprint arXiv:1909.00934* (2019).