Enhanced Bug Report Classification using XGBoost and Transformer-based NLP Models

P Vamsi[1], Dr. I. Nasrulla[2]

[1]PG Student, Vemu Institute of Technology, P. Kothakota

[2]Assistant professor, Vemu Institute of Technology, P. Kothakota

Abstract: **In software development, maintaining software systems has garnered attention due to the critical task of fixing defects identified during testing through bug reports (BRs). These BRs contain key details such as description, status, priority, and severity of bugs. The challenge lies in analyzing these growing numbers of BRs, which can be time-consuming and labor-intensive when done manually. Automation offers a promising solution. While much research focuses on automating tasks like predicting bug severity or priority, little attention is given to classifying the nature of the bugs. This paper proposes a new prediction model using natural language processing (NLP) and machine learning to automate this process. Simulated on publicly available datasets, the model demonstrated improved accuracy in predicting multi-class bug categories. Bug reports are essential for identifying and resolving issues in software systems. However, manual analysis of these reports is often time-consuming, error-prone, and inefficient due to the increasing volume and complexity of data. To address these challenges, this project proposes a nature-based prediction model that leverages ensemble machine learning techniques, particularly XGBoost, in combination with transformer-based Natural Language Processing (NLP) models. The approach automates the classification of bug reports into six distinct categories: Client, General, Hyades, Releng, Xtext, and CDT-core. Utilizing publicly available datasets from Eclipse and Mozilla, the model demonstrates superior performance, with XGBoost achieving an accuracy of 92%, outperforming other traditional models like SVM, Random Forest, and Logistic Regression. This system enhances software maintenance by improving classification accuracy, reducing manual effort, and expediting the bug resolution process.**

INTRODUCTION

In today's fast-paced software development environment, bug tracking systems are essential for ensuring software quality and reliability. Bug reports serve as crucial documentation that helps developers identify, categorize, and resolve issues. However, with the growing scale and complexity of software applications, the number of bug reports has increased significantly, making manual analysis both time-consuming and error-prone. This results in delays in identifying and fixing bugs, ultimately affecting the software maintenance process. To address these challenges, automation using Natural Language Processing (NLP) and Machine Learning (ML) techniques has emerged as a promising solution. This project introduces a nature-based prediction model that leverages ensemble machine learning, particularly the XGBoost algorithm, combined with transformer-based NLP models to classify bug reports into six categories: Client, General, Hyades, Releng, Xtext, and CDT-core. The model is trained and tested on publicly available datasets from Eclipse and Mozilla. Experimental results demonstrate that XGBoost achieves a classification accuracy of 92%, outperforming traditional models like SVM (74%), Random Forest (77%), Logistic Regression (70%), and even a Voting Classifier (89%). By automating the classification process, the proposed model significantly reduces manual effort, improves accuracy, and enhances the overall efficiency of software maintenance.

MOTIVATION

The increasing complexity and scale of software development have led to a rapid surge in the volume of bug reports generated by users and developers. Traditionally, these reports are analyzed manually, which is not only time-consuming but also prone to inconsistencies and human error. This inefficiency delays the bug-fixing process and affects the quality and reliability of software products. Moreover, existing automated tools often lack the precision and adaptability required for multi-class bug classification. These limitations inspired the need for a more accurate, efficient, and intelligent system that can handle large-scale bug data with minimal human intervention. The motivation behind this project is to harness the power of ensemble machine learning and advanced NLP techniques to automate bug report classification. By introducing a model based on XGBoost and transformer-based NLP, the goal is to improve classification accuracy, reduce developer workload, and accelerate the overall bug resolution process. This not only enhances software maintenance but also ensures faster delivery of stable and high-quality software systems.

PROPOSED SYSTEM

The proposed system introduces an advanced, automated framework for classifying bug reports using a combination of ensemble machine learning and natural language processing (NLP) techniques. Specifically, the model utilizes the XGBoost algorithm, known for its high performance in classification tasks, alongside transformer-based NLP models to effectively analyze and understand the textual content of bug reports. The system is designed to classify bug reports into six predefined categories: Client, General, Hyades, Releng, Xtext, and CDT-core. It processes bug data collected from publicly available datasets (such as Eclipse and Mozilla) by performing data cleaning, preprocessing, and feature extraction using NLP techniques. Multiple machine learning models, including SVM, Random Forest, and Logistic Regression, are trained and evaluated, but XGBoost delivers the highest accuracy of 92%. The system also includes modules for performance evaluation using metrics like precision, recall, F1-score, and confusion matrix. Overall, this proposed solution aims to reduce manual effort, enhance bug triaging accuracy, and improve the speed and efficiency of software maintenance workflows. So author of this paper trying to automate bug reports using machine learning algorithms. All existing bug prediction models are based on individual machine learning algorithms whose detection accuracy is not accurate. So author of this paper employing ensemble algorithm by combining various algorithms using Voting Classifier and then this algorithm will vote out each algorithm and then select algorithm with highest accuracy. Voting classifier combining various classifier such as Logistic Regression, Multinomial Naïve Bayes, SVM and Random Forest. Each algorithms run individually and with Voting classifier and Voting Classifier giving high accuracy.

LITERATURE REVIEW

1. M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, “Software testing techniques: A literature review,” *Proc. 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M)*, Nov. 2016, pp. 177–182.
2. S. Adhikarla, “Automated bug classification: Bug report routing,” M.S. thesis, Fac. Arts Sci., Dept. Comput. Inf. Sci., Linköping Univ., Sweden, 2020.
3. K. C. Youm, J. Ahn, and E. Lee, “Improved bug localization based on code change histories and bug reports,” *Inf. Softw. Technol.*, vol. 82, pp. 177–192, Feb. 2017.
4. N. Safdari et al., “Learning to rank faulty source files for dependent bug reports,” *Proc. SPIE*, vol. 10989, 2019, Art. no. 109890B.
5. A. Kukkar et al., “A novel deep-learning-based bug severity classification technique using CNN and random forest with boosting,” *Sensors*, vol. 19, no. 13, p. 2964, Jul. 2019.
6. A. Aggarwal. (May 2020). “Types of Bugs in Software Testing: 3 Classifications With Examples.” [Online]. Available: https://www.scnsoft.com/software-testing/types-of-bugs
7. A. Kukkar and R. Mohana, “A supervised bug report classification with incorporated textual field knowledge,” *Proc. Comput. Sci.*, vol. 132, pp. 352–361, Jan. 2018.
8. A. F. Otoom, S. Al-jdaeh, and M. Hammad, “Automated classification of software bug reports,” *Proc. 9th Int. Conf. Inf. Commun. Manage.*, Aug. 2019, pp. 17–21.
9. P. J. Morrison et al., “Are vulnerabilities discovered and resolved like other defects?” *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1383–1421, Jun. 2018.
10. F. Lopes et al., “Automating orthogonal defect classification using ML algorithms,” *Future Gener. Comput. Syst.*, vol. 102, pp. 932–947, Jan. 2020.
11.T. Hirsch and B. Hofer, “Root cause prediction based on bug reports,” *IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2020, pp. 171–176.
12.Q. Umer, H. Liu, and I. Illahi, “CNN-based automatic prioritization of bug reports,” *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1341–1354, Dec. 2020.
13.H. Bani-Salameh et al., “A deep-learning-based bug priority prediction using RNN-LSTM neural networks,” *e-Inform. Softw. Eng. J.*, vol. 15, no. 1, pp. 1–17, 2021.
14.Ö. Köksal and B. Tekinerdogan, “Automated classification of unstructured bilingual bug reports,” *Appl. Sci.*, vol. 12, no. 1, p. 338, Dec. 2021.

15. B. Alkhazi et al., “Learning to rank developers for bug report assignment,” *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106667.
16.L. Jonsson et al., “Automated bug assignment: Ensemble-based ML in industrial contexts,” *Empirical Softw. Eng.*, vol. 21, no. 4, pp. 1533–1578, Aug. 2016.
17. X. Ye, R. Bunescu, and C. Liu, “Learning to rank relevant files for bug reports,” *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2014, pp. 689–699.
18.Y. Tian et al., “Learning to rank for bug report assignee recommendation,” *Proc. IEEE 24th Int. Conf. Program Comprehension (ICPC)*, May 2016, pp. 1–10.
19.D. Devaiya, *Castr: A Web-Based Tool for Creating Bug Report Assignment Recommenders*, Univ. Lethbridge, 2019.
20.M. Alenezi, S. Banitaan, and M. Zarour, “Using categorical features in mining bug tracking systems,” arXiv:1804.07803, 2018.
21.H. A. Ahmed, N. Z. Bawany, and J. A. Shamsi, “CaPBug—a framework for automatic bug categorization and prioritization using NLP and ML,” *IEEE Access*, vol. 9, pp. 50496–50512, 2021.
22. R.-M. Karampatsis and C. Sutton, “How often do single-statement bugs occur?: The ManySStuBs4J dataset,” *Proc. 17th Int. Conf. Mining Softw. Repositories*, Jun. 2020, pp. 573–577.
22. X. Han, T. Yu, and D. Lo, “PerfLearner: Learning from bug reports to generate performance test frames,” *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 17–28.
23P. E. Strandberg et al., “Intermittently failing tests in embedded systems,” *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 337–348.

MODULUS AND WORKING

1. Data Collection and Preprocessing
The first step in the system is gathering bug report data from publicly available sources like the Eclipse and Mozilla datasets. These datasets contain real-world bug reports labeled into multiple categories. Once collected, the data undergoes preprocessing to remove noise and irrelevant information. This includes converting text to lowercase, removing stopwords, punctuation, and applying techniques like tokenization and lemmatization. These steps help in cleaning the text and making it suitable for further processing by machine learning models.

2. Feature Extraction
After preprocessing, the next module involves converting the cleaned textual data into numerical form so that it can be understood by machine learning algorithms. This is done using feature extraction techniques such as TF-IDF (Term Frequency–Inverse Document Frequency), which gives importance to significant words in the text, or word embeddings like Word2Vec or BERT. These features capture the context and meaning of the words, providing a robust representation of each bug report.

3. Model Training
This module is focused on training various machine learning models using the extracted features. The models include Support Vector Machine (SVM), Random Forest, Logistic Regression, and a Voting Classifier, which is a combination of several models. Each of these models is trained on 80% of the data and tested on the remaining 20% to assess their performance. The goal of this module is to identify the best-performing model for accurate bug classification.

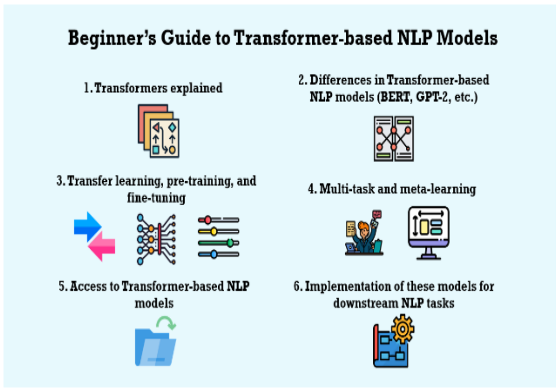
4. XGBoost Integration
To further improve classification accuracy, the XGBoost model is introduced. XGBoost is an advanced gradient boosting algorithm that builds a series of decision trees to make predictions. It is known for its high efficiency, accuracy, and speed. In this project, XGBoost outperformed all other models by achieving an impressive accuracy of 92%, making it the most effective model for classifying bug reports.

5. Model Evaluation
Once the models are trained, their performance is evaluated using standard metrics such as accuracy, precision, recall, and F1-score. Additionally, a confusion matrix is used to visualize the number of correct and incorrect predictions for each bug category. These metrics help compare the models and validate the effectiveness of the proposed approach. Among all models, XGBoost showed the highest performance in terms of accuracy and reliability.

6. Bug Prediction and Classification
The final module deals with the real-world application of the trained model. When a new or unseen bug report is input into the system, the model

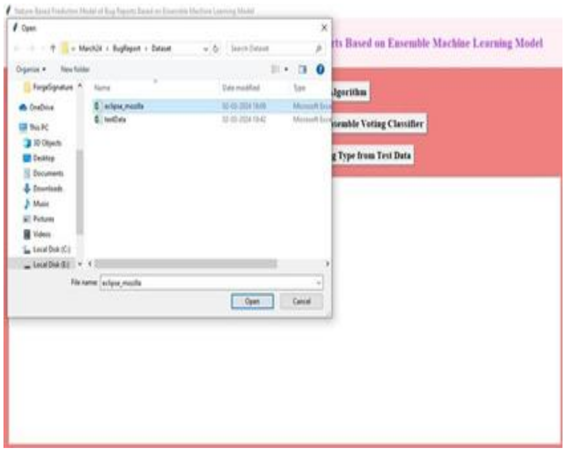
nalyses its content and automatically classifies it into one of six categories: Client, General, Hyades, Releng, Xtext, or CDT-core. This automation reduces manual work, increases classification speed, and helps software teams prioritize and address bugs more efficiently.

SYSTEM ARCHITECTURE


Beginner's Guide to Transformer-based NLP Models
1. Transformers explained
2. Differences in Transformer-based NLP models (BERT, GPT-2, etc.)
3. Transfer learning, pre-training, and fine-tuning
4. Multi-task and meta-learning
5. Access to Transformer-based NLP models
6. Implementation of these models for downstream NLP tasks

RESULT

Selecting and uploading dataset and then click on 'Open' button to select dataset.



CONCLUSION

In conclusion, this project presents a robust and efficient approach to automating bug report classification using ensemble machine learning and NLP techniques. By integrating the XGBoost algorithm with advanced textual feature extraction methods, the system achieves high accuracy and significantly outperforms traditional classifiers like SVM, Random Forest, and Logistic Regression. The model effectively classifies bug reports into six meaningful categories, thereby reducing manual effort, minimizing classification errors, and accelerating the bug triaging process. With a peak accuracy of 92%, the proposed system demonstrates its potential to greatly enhance software maintenance and support faster issue resolution in real-world development environments. This work lays the foundation for future improvements such as real-time classification, multilingual support, and integration with modern development tools.

REFERENCE

[1] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," *Proc. 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M)*, Nov. 2016, pp. 177–182.

[2] S. Adhikarla, "Automated bug classification: Bug report routing," M.S. thesis, Fac. Arts Sci., Dept. Comput. Inf. Sci., Linköping Univ., Sweden, 2020.

[3] K. C. Youm, J. Ahn, and E. Lee, "Improved bug localization based on code change histories and bug reports," *Inf. Softw. Technol.*, vol. 82, pp. 177–192, Feb. 2017.

[4] N. Safdari et al., "Learning to rank faulty source files for dependent bug reports," *Proc. SPIE*, vol. 10989, 2019, Art. no. 109890B.

[5] A. Kukkar et al., "A novel deep-learning-based bug severity classification technique using CNN and random forest with boosting," *Sensors*, vol. 19, no. 13, p. 2964, Jul. 2019.

[6] A. Aggarwal. (May 2020). "Types of Bugs in Software Testing: 3 Classifications With Examples." [Online]. Available: https://www.scnsoft.com/software-testing/types-of-bugs

[7] A. Kukkar and R. Mohana, "A supervised bug report classification with incorporated textual field knowledge," *Proc. Comput. Sci.*, vol. 132, pp. 352–361, Jan. 2018.

[8] A. F. Otoom, S. Al-jdaeh, and M. Hammad, "Automated classification of software bug reports," *Proc. 9th Int. Conf. Inf. Commun. Manage.*, Aug. 2019, pp. 17–21.

[9] P. J. Morrison et al., "Are vulnerabilities discovered and resolved like other defects?" *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1383–1421, Jun. 2018.

[10] F. Lopes et al., "Automating orthogonal defect classification using ML algorithms," *Future Gener. Comput. Syst.*, vol. 102, pp. 932–947, Jan. 2020.