UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

A Hybrid Model for Enhanced Bug Classification: Leveraging TF-IDF, BERT, and k-NN

Proyecto de Titulación

Diego Alejandro Marquez Coronel

Israel Pineda Ph.D.

Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito para la obtención del título de Magíster en Ciencia de Datos

Quito, 02 de diciembre de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ
COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

A Hybrid Model for Enhanced Bug Classification: Leveraging TF-IDF, BERT, and k-NN

Diego Alejandro Marquez Coronel

Nombre del Director del Programa:	Felipe Grijalva
Título académico:	Ph.D. en Ingeniería Eléctrica
Director del programa de:	Ciencia de datos
Nombre del Decano del colegio Académico:	Eduardo Alba
Título académico:	Doctor en Ciencias Matemáticas
Decano del Colegio:	Ciencias e Ingenierías
Nombre del Decano del Colegio de Posgrados:	Dario Niebieskikwiat
Título académico:	Doctor en Física

Quito, diciembre 2024

© DERECHOS DE AUTOR

Nombre del estudiante:	Diego Alejandro Marquez Coronel
Código de estudiante:	00338851
C.I.:	1724975386
Lugar y fecha:	Quito, 02 de diciembre de 2024.

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

Dedico este trabajo a mi familia, por ser mi guía y sostén en todo momento, y en especial a mi novia, Nathaly Villacreses.

Nathaly, tu amor, paciencia y apoyo constante han sido mi luz en los días más oscuros y mi motivación en cada desafío. Este logro es el reflejo de tu fe en mí y de la fuerza que me has dado para alcanzar mis metas.

A todos los que han creído en mí, este esfuerzo es también por ustedes.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que hicieron posible el desarrollo de este trabajo de titulación.

A mi director de tesis, **Israel Pineda**, por su invaluable guía, paciencia y aportes que enriquecieron este proyecto desde sus inicios hasta su conclusión. Su experiencia y disposición fueron fundamentales para el éxito de este trabajo.

A la Universidad San Francisco de Quito, por proporcionarme los recursos y el ambiente académico necesario para llevar a cabo esta investigación. Agradezco también a mis profesores y compañeros, quienes con su apoyo y conocimiento contribuyeron a mi formación académica y profesional.

Finalmente, reconozco a las comunidades y herramientas de software de código abierto que hicieron posible este trabajo, en particular al equipo detrás de *Bugs.jar*, por proporcionar un recurso tan valioso para la investigación en clasificación de bugs.

RESUMEN

La creciente complejidad de los sistemas de software modernos ha generado la necesidad de contar con métodos más eficientes para la clasificación de errores, con el fin de optimizar el mantenimiento y reducir el tiempo de inactividad. Este trabajo presenta un modelo híbrido que combina técnicas tradicionales de recuperación de información, como TF-IDF, con modelos avanzados de aprendizaje profundo, como BERT, integrados mediante un clasificador k-NN.

El enfoque propuesto se evaluó utilizando el conjunto de datos Bugs.jar, un benchmark de gran escala y diversidad. Los resultados experimentales muestran una mejora significativa en la precisión y el rendimiento del modelo híbrido, alcanzando una precisión general del 82%. Este enfoque no solo aborda las limitaciones de las técnicas individuales, sino que también ofrece una solución robusta para la clasificación de errores en proyectos de software de diversa índole. Finalmente, se analizan los desafíos asociados al desbalance de clases y la eficiencia computacional, abriendo nuevas posibilidades para investigaciones futuras en esta área.

Palabras clave: Bug Classification, TF-IDF, BERT, k-NN, Mantenimiento de Software, Modelos Híbridos, Aprendizaje Profundo.

ABSTRACT

The increasing complexity of modern software systems necessitates more efficient methods for bug classification to streamline maintenance processes and minimize downtime. This study presents a hybrid model that integrates traditional information retrieval techniques, such as TF-IDF, with advanced deep learning approaches, including BERT, combined through a k-NN classifier.

The proposed approach was evaluated using the Bugs.jar dataset, a large-scale and diverse benchmark. Experimental results demonstrate a significant improvement in accuracy and overall model performance, achieving an overall precision of 82%. This hybrid method addresses the limitations of individual techniques while offering a robust solution for bug classification in diverse software projects. Challenges related to class imbalance and computational efficiency are analyzed, providing new opportunities for future research in this field.

Key words: Bug Classification, TF-IDF, BERT, k-NN, Software Maintenance, Hybrid Models, Deep Learning.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

A Hybrid Model for Enhanced Bug Classification: Leveraging TF-IDF, BERT, and k-NN

Diego Marquez, *Student, USFQ*

***Abstract*—The escalating complexity of modern software systems necessitates efficient bug classification to streamline maintenance and reduce downtime. Traditional Information Retrieval (IR) techniques, such as TF-IDF, often fall short in capturing the nuanced semantic relationships embedded in bug reports and code changes. This paper presents a hybrid model that integrates TF-IDF vectorization, BERT embeddings, and k-Nearest Neighbors (k-NN) classification to enhance bug classification accuracy across multiple software projects. Experimental results on the Bugs.jar dataset reveal that the proposed approach significantly outperforms individual models, achieving an overall accuracy of 82%, and addressing their limitations. This offers a robust solution for more efficient software maintenance.**

***Index Terms*—Bug Classification, TF-IDF, BERT, k-NN, Software Maintenance, Machine Learning, Hybrid Model, Deep Learning**

I. Introduction

Bug classification is a critical component of the software development lifecycle, particularly during the maintenance phase, where the primary objective is to ensure software reliability and user satisfaction. Effective bug classification enables development teams to efficiently identify, categorize, and prioritize issues, ensuring each bug is assigned to the appropriate team or individual with the necessary expertise. This process minimizes resolution time, reduces development costs, and enhances overall software quality [13]. As software systems grow in complexity and scale—with extensive codebases and distributed development teams—the volume of bug reports has surged, often overwhelming manual triage processes [11].

Bug reports are typically unstructured textual documents containing summaries, descriptions, steps to reproduce the issue, and sometimes logs or stack traces [1]. The variability in how bugs are reported, driven by differences in natural language usage, technical terminology, and reporting styles, further complicates accurate classification. Moreover, ambiguous, incomplete, or inconsistent information can hinder effective bug diagnosis [8].

Traditional Information Retrieval (IR) methods, such as Term Frequency-Inverse Document Frequency (TF-IDF), have been widely employed to extract relevant information from textual data and support tasks like document classification and clustering [16], [10]. In the context of bug classification, TF-IDF identifies significant terms within reports, which can be used as features for machine learning models. However, these methods rely on term frequency and fail to capture deeper semantic relationships and contextual nuances between words, limiting their effectiveness in complex scenarios such as software debugging [10].

The advent of deep learning and advanced Natural Language Processing (NLP) techniques has introduced new methods to address these challenges. Models such as **BERT** (Bidirectional Encoder Representations from Transformers) generate contextual embeddings that capture rich semantic information from text [6]. BERT employs a transformer architecture to understand the bidirectional context of words in a sentence, enabling precise representations of language nuances. This capability has been applied to various NLP tasks, including text classification, named entity recognition, and question answering, achieving state-of-the-art results [9]. In bug classification, BERT can help bridge lexical gaps and better capture the underlying semantics of bug reports [21].

Despite its advantages, using BERT alone for bug classification presents several challenges. BERT models are computationally intensive, requiring significant resources for training and inference [17]. While fine-tuning BERT for specific tasks can improve performance, it may not be feasible for all applications due to computational demands and the requirement for high-quality labeled data [18]. Additionally, BERT does not fully leverage term frequency information, which can be valuable in certain contexts [20].

This paper proposes a **hybrid** approach that combines TF-IDF vectorization with BERT embeddings, utilizing a **k-Nearest Neighbors (k-NN)** classifier to exploit the strengths of both traditional IR techniques and deep learning models. The TF-IDF component captures the importance of terms within the corpus, providing a lexical representation, while BERT embeddings contribute contextual and semantic understanding. By integrating these features, we aim to enhance the representation of bug reports, resulting in improved classification accuracy.

The **k-NN classifier** was chosen for its simplicity and effectiveness in handling multi-class classification problems, as well as its ability to operate well with combined feature spaces [12]. The classifier predicts the class of a bug report based on the majority class among its nearest neighbors in the feature space, making it suitable for datasets where similar bugs share common characteristics.

We evaluate our proposed model using the **Bugs.jar** dataset, a large-scale and diverse collection of real-world

Java bugs from open-source projects [4]. This dataset provides a rich testbed for evaluating bug classification models due to its variety and the inclusion of detailed bug reports and code changes.

The contributions of this paper are summarized as follows:

- We introduce a novel hybrid model that combines TF-IDF and BERT embeddings with a k-NN classifier to predict the project associated with a bug, leveraging both lexical and semantic features.
- We conduct extensive experiments on the Bugs.jar dataset, demonstrating that our hybrid approach significantly improves classification accuracy compared to models that use only TF-IDF or BERT embeddings.
- We explore ensemble techniques, such as voting and stacking, to enhance model performance, providing insights into their effectiveness in bug classification.
- We analyze challenges associated with class imbalance and computational efficiency, discussing potential solutions and directions for future research.

II. Related Work

Research on bug localization and classification has evolved significantly, with a strong focus on Information Retrieval (IR) techniques and, more recently, deep learning models. This section provides an overview of the most relevant works in the field, highlighting the strengths and limitations of various approaches.

A. Information Retrieval Techniques

Traditional IR techniques, such as **TF-IDF** (Term Frequency-Inverse Document Frequency) [16] and **LDA** (Latent Dirichlet Allocation), have been widely applied in bug classification tasks [10]. These methods extract significant terms from bug reports and source code, facilitating processes like duplicate bug detection and error localization [15]. However, they are susceptible to **lexical mismatches**, where reports describe the same issue using different terms or phrasing. This limitation can adversely affect the performance of models that rely solely on term frequency-based representations, as they struggle to capture the semantic equivalence between different wordings of the same concept.

B. Deep Learning Models

Recent advancements in **Natural Language Processing (NLP)** have led to the widespread adoption of deep learning models, particularly transformer-based architectures like **BERT** [6]. BERT has demonstrated superior performance in various NLP tasks by generating contextual embeddings that simultaneously capture text semantics from both left and right contexts. In bug localization, BERT-based models help bridge lexical gaps and enhance the understanding of complex bug reports [21]. However, these models require significant computational resources and often need fine-tuning to perform optimally on specific datasets [17].

C. Hybrid Models

Given the limitations of individual approaches, hybrid models that combine **traditional IR techniques with deep learning** have gained traction. For instance, **Saha et al.** introduced the Bugs.jar dataset to provide a benchmark for testing such hybrid models in the context of Java programs [4]. Hybrid models leverage **TF-IDF** to extract key textual features, while **BERT embeddings** enrich the semantic representation [9]. This combination has improved classification accuracy, mainly when used alongside ensemble techniques [7].

Building on these insights, our work integrates **TF-IDF vectorization** with **BERT embeddings** in an **ensemble framework**. We utilize a **k-Nearest Neighbors (k-NN)** classifier to predict the project associated with a bug and experiment with **ensemble techniques**, such as voting and stacking, to further enhance performance [14]. By applying this hybrid approach to the Bugs.jar dataset, we aim to demonstrate the potential of combining IR and deep learning methods for more effective bug classification.

D. Benchmarks and Datasets

The effectiveness of bug classification models largely depends on the availability of high-quality datasets. The **Bugs.jar dataset** provides a large-scale collection of real-world Java bugs, offering diverse test cases across multiple open-source projects [4]. Similarly, the **BigIssue benchmark** addresses the limitations of synthetic datasets by providing realistic and challenging scenarios for evaluating bug localization models [2]. These datasets play a crucial role in developing and benchmarking advanced bug classification techniques.

III. Dataset

The dataset used in this study is derived from the **Bugs.jar** repository [4]. This dataset offers a large-scale, diverse collection of real-world Java bugs spanning multiple open-source projects. Bugs.jar provides detailed bug reports and the corresponding code changes and metadata, making it a comprehensive resource for research in bug classification and localization.

A. Overview of Bugs.jar

The Bugs.jar dataset is structured around several key elements:

- **Project**: The specific project to which the bug belongs. Examples include Accumulo, Camel, and Flink.
- **Bug ID**: A unique identifier assigned to each bug report, often corresponding to an issue in a version control system (e.g., JIRA).
- **Summary and Description**: Textual information detailing the nature of the bug, its expected behavior, and steps to reproduce the issue.
- **Code Changes**: Modified source code files, including both buggy and fixed versions. These snapshots track

the evolution of the code and provide insights into the nature of the fix.

The diversity of projects within Bugs.jar makes it an ideal benchmark for evaluating machine learning models across varied software domains [2]. It offers real-world complexity and heterogeneity, crucial for developing robust classification models. Additionally, Bugs.jar allows for extracting fine-grained information, such as file diffs and developer comments, which enrich the feature space used for modeling.

IV. Methodology

Our methodology integrates traditional **Information Retrieval (IR)** techniques with modern **deep learning models**, forming a hybrid framework for bug classification. This approach combines TF-IDF vectorization and BERT embeddings, leveraging a **k-Nearest Neighbors (k-NN)** classifier to predict the project associated with each bug.

A. Data Preprocessing

The preprocessing phase is crucial to ensure the consistency and quality of the input data. The following steps were implemented:

- **Handling Missing Values**: Missing values in the summary and description fields were replaced with empty strings to prevent issues during vectorization.
- **Combining Text Fields**: The `Summary` and `Description` fields were merged to create a unified textual input. This combined input ensures that both high-level summaries and detailed descriptions are captured during vectorization.

B. Feature Extraction and Combination

We applied **TF-IDF** (Term Frequency - Inverse Document Frequency) to generate numerical representations of the merged text data, limited to the **1000 most relevant terms** [16]. Additionally, we extracted **BERT embeddings** from a pre-trained BERT model to capture the semantic context of the text [6].

These features were combined to form a comprehensive feature matrix used for classification.

C. k-NN Classifier

The **k-Nearest Neighbors (k-NN)** algorithm was selected due to its simplicity and effectiveness in multi-class classification tasks [5]. The model was trained using the combined TF-IDF vectors and BERT embeddings, allowing it to leverage both lexical and semantic information.

V. Model Architecture

This section outlines the architecture of the proposed hybrid model, which integrates TF-IDF, BERT embeddings, and a k-Nearest Neighbors (k-NN) classifier for bug classification across multiple software projects. Each component of the model plays a crucial role in enhancing the classification performance.

The architecture consists of the following components:

- **Data Input**: Bug reports, including summaries and descriptions, are extracted from the Bugs.jar dataset [4].
- **Preprocessing**: Missing values are handled, and text fields are combined to ensure the consistency of the input data.
- **TF-IDF Vectorization**: The combined text is transformed into numerical vectors using TF-IDF, capturing term importance across reports [16].
- **BERT Embeddings**: Pre-trained BERT generates semantic embeddings, enriching the feature space with contextual information [6].
- **Feature Combination**: TF-IDF vectors and BERT embeddings are concatenated to form a unified feature matrix.
- **k-NN Classification**: A k-NN classifier is trained on the combined feature matrix to predict the project associated with each bug [5].

Future enhancements include the exploration of ensemble learning techniques, such as voting and stacking, to further improve performance by leveraging the strengths of different models [7].

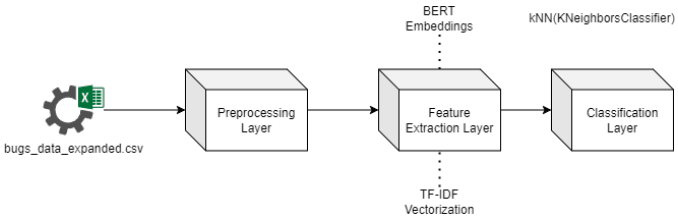

Figure 1. Proposed Architecture of the Hybrid Model (Diagram to be added).

This modular architecture leverages both lexical and semantic features, providing a robust framework for bug classification across diverse software projects.

VI. Exploratory Data Analysis (EDA)

To build a robust model, we conducted an exploratory data analysis (EDA) to better understand the distribution and characteristics of the bugs in the Bugs.jar dataset [4].

A. Bug Distribution by Project

Figure 2 illustrates the distribution of bugs across various projects in the Bugs.jar dataset. Notably, projects such as Accumulo and Camel report a larger number of bugs, while others have fewer. This disparity in bug distribution may affect the classifier's performance, especially for projects with limited data, where the model may struggle to generalize due to the smaller sample size.

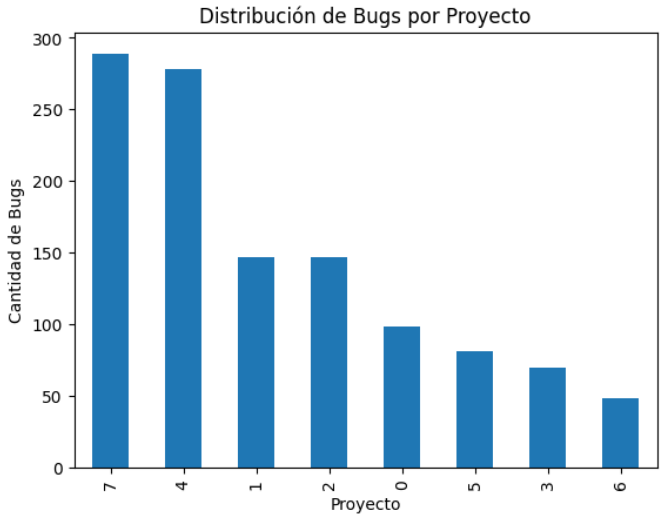

Figure 2. Bug distribution across projects in the Bugs.jar dataset.

B. Frequent Words in Bug Descriptions

An analysis of the most frequently occurring terms in bug summaries and descriptions sheds light on common issues reported in the projects. Figure 3 highlights terms such as "error," "exception," and "failure," which frequently appear in bug reports. These terms underscore the typical problems developers encounter, providing valuable context for feature extraction and model training.

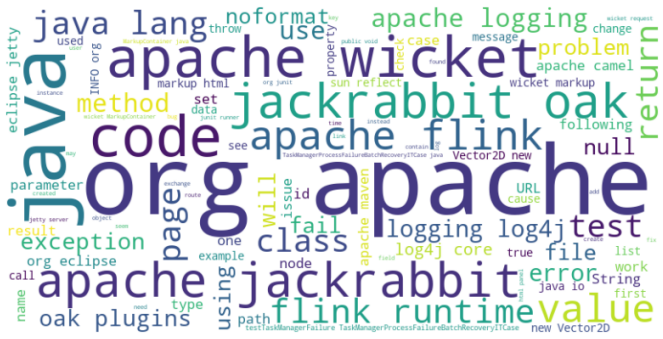

Figure 3. Word cloud of the most frequent terms in bug reports.

VII. Experimental Results

This section presents the experiments conducted to evaluate the performance of different models, focusing on k-NN and BERT, followed by their combined use through TF-IDF and BERT embeddings.

A. Initial k-NN Model with TF-IDF

The first experiment utilized a k-NN classifier trained exclusively on TF-IDF features. The results in Table I highlight the model's ability to classify bugs based on term frequency information alone.

Table I
Classification Report for k-NN with TF-IDF

Class	Precision	Recall	F1-score	Support
0	0.52	0.48	0.50	33
1	0.80	0.87	0.84	38
2	0.76	0.79	0.77	52
3	0.86	0.52	0.65	23
4	0.83	0.84	0.83	81
5	0.61	0.74	0.67	27
6	1.00	0.57	0.73	14
7	0.78	0.82	0.80	80
Accuracy			0.76	
Macro Avg	0.77	0.71	0.72	348
Weighted Avg	0.77	0.76	0.76	348

B. BERT Fine-tuning Performance

The second experiment focused on fine-tuning a pre-trained BERT model over three epochs. As shown in Table II, BERT exhibited improved understanding of the bug reports, especially in capturing semantic nuances, although its recall for smaller classes like class 6 remained low.

Table II
Classification Report for Fine-tuned BERT

Class	Precision	Recall	F1-score	Support
0	0.72	0.55	0.62	33
1	0.80	0.84	0.82	38
2	0.69	1.00	0.82	52
3	0.31	0.43	0.36	23
4	0.89	0.83	0.86	81
5	0.77	0.85	0.81	27
6	0.50	0.07	0.12	14
7	1.00	0.86	0.93	80
Accuracy			0.78	
Macro Avg	0.71	0.68	0.67	348
Weighted Avg	0.80	0.78	0.78	348

C. k-NN with Combined TF-IDF and BERT Embeddings

The final experiment involved retraining the k-NN classifier using a feature matrix that combined TF-IDF and BERT embeddings. The results in Table III reveal significant improvements in classification performance, with an overall accuracy of 82%.

Table III
Classification Report for k-NN with Combined TF-IDF and BERT Embeddings

Class	Precision	Recall	F1-score	Support
0	0.72	0.77	0.74	98
1	0.78	0.79	0.78	147
2	0.93	0.85	0.89	147
3	0.83	0.61	0.70	70
4	0.80	0.90	0.84	278
5	0.81	0.67	0.73	81
6	0.92	0.75	0.83	48
7	0.84	0.87	0.86	289
Accuracy			0.82	
Macro Avg	0.83	0.78	0.80	1158
Weighted Avg	0.82	0.82	0.82	1158

VIII. Discussion

The experimental results underscore the efficacy of the hybrid model that combines TF-IDF, BERT embeddings,

and a k-NN classifier. Achieving an overall accuracy of 82%, the model successfully leverages both lexical and semantic features. TF-IDF captures term frequency and importance, providing a solid foundation of lexical information, while BERT embeddings contribute nuanced contextual insights, enhancing the model's understanding of complex bug reports.

Despite these promising outcomes, performance disparities among different classes were observed. Classes with ample data, such as class 4 and class 7, displayed high precision and recall, suggesting the model's strong generalization capability when sufficient training samples are available. Conversely, class 6, characterized by fewer instances, exhibited notably lower recall (0.43) and precision (0.67), highlighting the challenge of class imbalance—a prevalent issue in many real-world datasets.

To mitigate class imbalance, future research could explore **data augmentation** or **oversampling techniques**, such as **SMOTE (Synthetic Minority Over-sampling Technique)**, to improve the representation of underrepresented classes. These methods could enhance the model's recall and overall balance. Additionally, further **transfer learning** or extending the fine-tuning of BERT could refine the model's adaptability to smaller classes, potentially boosting performance in these areas.

While **k-NN** was effective for this dataset, its computational demands increase with dataset size, posing scalability challenges. Future work might explore alternative classifiers such as **Random Forests** or **XGBoost**, known for their scalability and efficiency, to improve performance on larger datasets.

The preliminary investigation into **ensemble learning techniques**, including voting and stacking, revealed their potential to enhance classification performance. However, these techniques warrant further exploration. Voting ensembles, though straightforward, were sensitive to the weaknesses of individual models, whereas stacking demonstrated more robust outcomes by learning from base models' predictions. Deepening the exploration of these strategies could unlock further performance gains.

In summary, the hybrid model effectively addresses the multifaceted challenges of bug classification by combining traditional and modern techniques. While the current findings are promising, they also pave the way for several avenues of future research. Enhancing class balance, refining model architecture, and scaling for larger datasets remain critical areas for development. Additionally, more sophisticated ensemble methods could be leveraged to further improve the model's robustness and scalability, ultimately advancing the field of automated bug classification.

IX. Future Work

The proposed hybrid model has yielded promising results; however, there remain several avenues for further enhancement. Future research can focus on the following aspects to improve both the effectiveness and efficiency of bug classification:

- **Fine-tuning BERT for Bug-specific Data**: Adapting BERT specifically for bug report data could substantially enhance its ability to capture the nuanced language of software maintenance, including technical jargon and domain-specific terms. Experimenting with extended training epochs and layer-wise learning rates may further refine the semantic embeddings, leading to more accurate and contextually aware representations.
- **Addressing Class Imbalance through Data Augmentation and Oversampling**: The challenge of class imbalance, particularly evident in underrepresented classes like class 6, remains a critical area for improvement. Applying techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or generating synthetic bug reports could bolster the representation of these classes. These methods would likely enhance recall and overall generalization, leading to more balanced and reliable classification outcomes.
- **Exploration of Advanced Classifiers**: While k-NN served as a strong baseline, its scalability is limited as dataset size increases. Future research could explore more advanced classifiers like Random Forest or XGBoost, which are well-regarded for their efficiency and scalability with large datasets. Additionally, experimenting with transformer-based models for sequence classification might offer further accuracy improvements by capturing intricate dependencies within textual data.
- **Enhancing Scalability and Deployment**: For real-world applicability, optimizing the computational efficiency of the hybrid model is crucial. Techniques such as dimensionality reduction or model quantization could significantly reduce inference time without compromising performance. Integrating the hybrid model into automated software maintenance pipelines could streamline bug classification and triage, reducing manual effort and enhancing operational efficiency in practical scenarios.
- **Comprehensive Exploration of Ensemble Learning**: Although initial experiments with ensemble techniques like voting and stacking showed promise, a deeper exploration is warranted. Future studies could investigate the optimal combination of TF-IDF, BERT embeddings, and k-NN through varied ensemble strategies. Tuning hyperparameters and experimenting with different ensemble frameworks could balance accuracy and computational efficiency, pushing the model's performance further.

Addressing these aspects could significantly enhance the robustness, scalability, and practical applicability of the hybrid model, setting the stage for more effective and precise bug classification in diverse software development projects.

X. Conclusion

This paper introduced a hybrid approach to bug classification by integrating TF-IDF vectorization, BERT embeddings, and the k-Nearest Neighbors (k-NN) classifier. Experimental results on the Bugs.jar dataset demonstrated the model's effectiveness, achieving a notable accuracy of 82%. This highlights the potential of combining traditional Information Retrieval (IR) techniques with modern deep learning methods to tackle complex bug classification tasks.

The model's strength lies in its ability to harness the simplicity and efficiency of TF-IDF for identifying critical terms within bug reports, alongside BERT's capability to capture nuanced semantic relationships often overlooked by traditional methods. This combination proved effective across diverse and heterogeneous software projects, addressing the challenges posed by varying reporting styles and complex codebases.

Despite the promising results, several challenges remain. The model's performance on underrepresented classes, such as class 6, revealed limitations in handling class imbalance. This underscores the need for refined techniques like data augmentation and oversampling to ensure balanced performance across all classes. Additionally, while k-NN served as an effective baseline, its computational demands could hinder scalability as dataset sizes increase. Future exploration of advanced classifiers, such as Random Forests or XGBoost, could offer significant gains in both performance and efficiency.

The initial exploration of ensemble learning methods showed potential, but a comprehensive evaluation of these strategies was beyond the scope of this study. Future research should delve deeper into these techniques, leveraging the strengths of various models to further enhance classification accuracy and robustness.

In conclusion, this study affirms the efficacy of hybrid models in advancing bug classification for software maintenance. By effectively capturing both lexical and semantic intricacies, these models can improve the accuracy and reliability of bug triage, ultimately reducing system downtime and enhancing software quality. Addressing the identified challenges and exploring more advanced methodologies will be crucial in pushing the boundaries of automated bug classification.

Acknowledgments

The author thanks the Universidad San Francisco de Quito for their support and resources. Appreciation is also extended to the faculty, peers, and the academic community for their valuable guidance and contributions throughout this research.

References

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

[2] P. Kassianik, E. Nijkamp, B. Pang, Y. Zhou, and C. Xiong, "BigIssue: A Realistic Bug Localization Benchmark," *36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] R. K. Saha, Y. Lyu, W. Lam, H. Yoshida, and M. R. Prasad, "Bugs.jar: A Large-scale, Diverse Dataset of Real-world Java Bugs," in *Proceedings of the 15th International Conference on Mining Software Repositories (MSR)*, 2018, pp. 10–13.

[5] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[6] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, Springer, 2000, pp. 1–15.

[8] A. J. Ko, R. DeLine, and G. Venolia, "Information Needs in Collocated Software Development Teams," in *Proceedings of the 29th International Conference on Software Engineering (ICSE)*, 2007, pp. 344–353.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019.

[10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[11] A. Mockus, D. M. Weiss, and P. Zhang, "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.

[12] L. Peterson, "k-Nearest Neighbor," in *Encyclopedia of Machine Learning*, Springer, 2009.

[13] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed., McGraw-Hill, 2005.

[14] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, 2010.

[15] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing," in *Proceedings of the 29th International Conference on Software Engineering (ICSE)*, 2007, pp. 499–510.

[16] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[17] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[18] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?," in *Proceedings of the China National Conference on Chinese Computational Linguistics*, Springer, 2019, pp. 194–206.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," *Advances in Neural Information Processing Systems*, 2017.

[20] S. Ye, F. Li, J. Ding, and B. Yang, "Word2vec: A Study of All the Model Variants," *arXiv preprint arXiv:1607.04606*, 2016.

[21] J. Zhou, W. Wu, C. Xiong, and J. L. Zhao, "Transformers in Text Classification: A Detailed Survey," *IEEE Access*, vol. 8, pp. 188578–188593, 2020.