iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81619

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

A Machine Learning-Based Framework for Accurate Duplicate Bug Detection

V. Raja Manikanta[1], K. Prasanthi[2], Ch. Sandeep[3], J. Pawan Kumar[4], N. Suresh[5]
Department of Cyber Security, Acharya Nagarjuna University, Andhra Pradesh, India

Abstract: ***In modern software development environments, duplicate bug reports have become a significant challenge, leading to increased manual effort, delayed issue resolution, and reduced productivity. Developers often describe the same issue using different terminology, making manual identification of duplicates time-consuming and error-prone. In this paper, we present an efficient and scalable machine learning-based framework for duplicate bug detection using Natural Language Processing (NLP) techniques. The proposed system integrates text preprocessing methods such as tokenization, stopword removal, and normalization to enhance data quality. TF-IDF (Term Frequency–Inverse Document Frequency) is used for feature extraction, and cosine similarity is applied to measure the similarity between bug reports. The system assigns similarity scores and categorizes results into confidence levels to determine whether a bug is a duplicate or unique. A Streamlit-based interface is developed to enable real-time detection and user interaction with visual feedback. Experimental results demonstrate improved efficiency in identifying duplicate bug reports, reducing manual effort, and enhancing accuracy compared to traditional approaches. The proposed approach provides a reliable, efficient, and scalable solution for automated bug tracking and software maintenance in large-scale development environments.***

Keywords: Duplicate Bug Detection, Natural Language Processing, Machine Learning, TF-IDF, Cosine Similarity, Software Engineering

I. INTRODUCTION

Duplicate bug reports have become a common and significant challenge in modern software development environments, posing difficulties for development teams and project management processes. With the rapid growth of collaborative platforms and large-scale applications, there is an increasing volume of bug reports submitted by different users and developers. This creates a strong need for efficient systems that can automatically identify duplicate issues. Traditional bug tracking methods often rely on manual inspection of reports, which can be time-consuming, inconsistent, and prone to human error. These limitations reduce the efficiency of issue resolution and highlight the necessity for automated and intelligent detection systems.

Recent observations indicate that failure to identify duplicate bugs leads to redundant work, increased development time, and inefficient resource utilization. Conventional approaches often struggle with challenges such as variations in textual descriptions, lack of structured data, and difficulty in capturing semantic similarity between bug reports. Additionally, the diversity in language used by different reporters makes it difficult to identify meaningful relationships using basic keyword matching techniques. These challenges emphasize the need for advanced computational methods that can improve detection accuracy and support efficient bug management.

Machine learning and Natural Language Processing (NLP) have emerged as powerful tools for analyzing textual data and extracting meaningful patterns. By leveraging algorithms capable of identifying relationships between words and phrases, these techniques can effectively detect similarities between bug descriptions. Methods such as feature extraction and similarity measurement further enhance the performance of detection systems. These advancements enable the development of automated solutions that assist developers in identifying duplicate issues and improving overall productivity.

In this paper, we present a machine learning-based duplicate bug detection system that utilizes techniques such as text preprocessing, TF-IDF vectorization, and cosine similarity for effective comparison of bug reports. The system incorporates preprocessing steps such as tokenization, stop sword removal, and normalization to improve text quality and consistency. A user-friendly Streamlit-based interface is developed to enable real-time detection and provide clear and interpretable results for users.

To address performance and efficiency challenges, the proposed system evaluates similarity scores and classifies results based on predefined confidence thresholds such as high, medium, and low matches. The system is designed to be scalable and efficient, making it suitable for real-world software development environments. Additionally, it supports faster identification of duplicate bugs, reducing manual effort and improving issue tracking processes

II. LITERATURE REVIEW

The application of machine learning in software engineering, particularly for duplicate bug detection, has gained significant attention in recent years due to the increasing volume of bug reports and the need for efficient issue management. Numerous research studies have explored automated approaches to improve accuracy and assist developers in identifying duplicate reports. These approaches utilize Natural Language Processing (NLP) and computational techniques to analyze textual bug descriptions and identify patterns associated with similar issues.

Ahmed et al. proposed a duplicate bug detection model using Gradient Boosting combined with optimization techniques to enhance similarity prediction performance. Their approach demonstrated improved accuracy compared to traditional keyword-based methods; however, the complexity of optimization increased computational cost and limited real-time applicability in large-scale systems.

Choi et al. introduced a detection system that incorporates data balancing and preprocessing techniques to improve model performance on textual datasets. While these techniques improved detection sensitivity and recall, the approach still faced challenges related to overfitting and generalization when applied to diverse bug reporting styles and platforms.

Mortazavi et al. conducted a comparative analysis of multiple machine learning algorithms, including Decision Tree, Random Forest, and Logistic Regression, for duplicate bug detection. Their results highlighted the effectiveness of ensemble methods; however, the study lacked efficient deployment strategies and did not address real-time detection requirements in practical environments.

Beunza et al. applied optimization methods to improve the performance of text-based similarity models using bug report datasets. Although their approach enhanced detection accuracy, it required extensive parameter tuning and computational resources, making it less suitable for scalable and lightweight applications.

Alaa et al. explored the use of automated machine learning (AutoML) techniques for text classification and similarity detection. Their system achieved promising results by automating feature extraction and model selection, but it lacked interpretability and transparency, which are important for understanding similarity decisions in bug tracking systems.

Several existing approaches also utilize feature extraction and similarity analysis techniques to improve detection performance. While these methods help in identifying important textual patterns, they often fail to capture semantic meaning effectively and may overlook relationships between differently worded but similar bug descriptions.

From the above studies, it can be observed that although machine learning-based duplicate bug detection systems show significant potential, existing approaches suffer from limitations such as variability in textual data, overfitting, high computational cost, and lack of real-time deployment capabilities. These challenges reduce their effectiveness in practical software development environments.

To address these issues, this work proposes a machine learning-based duplicate bug detection system that integrates preprocessing techniques such as text cleaning, tokenization, and normalization, along with feature extraction using TF-IDF and similarity measurement using cosine similarity. The proposed approach ensures improved detection accuracy, better generalization, and efficient real-time implementation for automated bug tracking and management systems.

III. AIM OF THE STUDY

This work aims to design and develop an efficient machine learning-based system for accurate duplicate bug detection using textual bug report data. The proposed system addresses the limitations of traditional manual bug tracking methods by improving detection accuracy, handling variations in textual descriptions, and enabling real-time decision support for developers.

The study has the following specific objectives:

1) To design a duplicate bug detection framework using machine learning techniques that can accurately identify similar bug reports based on textual descriptions.

2) To implement preprocessing techniques such as text cleaning, tokenization, and normalization to improve the quality and consistency of the dataset.

3) To apply effective text processing methods to handle variations in language and improve model generalization across different bug reporting styles.

4) To develop and evaluate similarity-based techniques including TF-IDF vectorization and cosine similarity for accurate comparison of bug reports.

5) To identify optimal similarity thresholds based on evaluation metrics such as accuracy, precision, recall, and F1-score for reliable duplicate detection.

6) To perform feature extraction and text analysis to retain significant keywords and improve detection performance and system stability.
7) To implement a real-time detection system using a Streamlit-based interface for user-friendly interaction and deployment.
8) To enhance the efficiency, scalability, and reliability of the system for practical software development environments.
9) To improve early identification of duplicate bug reports, reducing manual effort and supporting faster issue resolution and better development workflow management.

IV. PROPOSED SYSTEM

The proposed system introduces a machine learning-based framework for accurate and efficient duplicate bug detection using textual bug report data. The system is designed to overcome the limitations of traditional manual bug tracking approaches by improving detection accuracy, handling variations in textual descriptions, and enabling real-time decision support for developers and software teams.

A. *System Overview*

The system operates on a structured machine learning pipeline, where bug report data is processed, analyzed, and used for similarity-based detection. It integrates preprocessing techniques, feature extraction, and text similarity algorithms such as TF-IDF and cosine similarity to provide accurate results. Unlike traditional manual inspection, the proposed system automates the detection process and assists developers in identifying duplicate bugs efficiently.

B. *System Architecture*

The architecture of the proposed system consists of the following key components:
1) Dataset: Bug report data containing attributes such as ID, title, description, and category.
2) Preprocessing Module: Handles text cleaning, tokenization, stopword removal, and normalization to improve data quality.
3) Similarity Computation Module: Implements TF-IDF vectorization and cosine similarity to measure similarity between bug reports.
4) Prediction Interface: A Streamlit-based front-end that allows users to input bug descriptions and receive duplicate detection results in real time.

C. Working Mechanism:

Fig. 1. Workflow of the proposed machine learning-based duplicate bug detection system

The proposed system operates through the following workflow:
1) Data Collection: Bug report dataset is obtained and structured for analysis.
2) Data Preprocessing: Text cleaning, normalization, and removal of irrelevant words are performed using NLP techniques.
3) Feature Extraction: Important textual features are extracted using TF-IDF to convert text into numerical vectors.
4) Similarity Computation: Cosine similarity is used to compare the input bug with existing bug reports.
5) Prediction: The system calculates similarity scores and identifies whether the bug is duplicate or unique based on threshold values.

D. *Model Functionality*

The system performs similarity-based classification of bug reports into duplicate categories. The primary functions include:
1) Train Model: Learns patterns from existing bug reports to build vector representations.
2) Detect Duplicate: Classifies input bug descriptions into High Match, Medium Match, Low Match, or No Match categories.

These automated processes improve efficiency and reduce dependency on manual analysis.

E. *Security Properties*

The proposed system incorporates several important features:
1) Accuracy: Machine learning techniques provide reliable similarity detection based on textual data.
2) Scalability: The system can handle large volumes of bug reports and multiple users efficiently.
3) Interpretability: Provides clear similarity scores and confidence levels for decision-making.
4) Reliability: Consistent detection results with improved generalization using preprocessing and feature extraction techniques.

F. Benefits of the Proposed System

The proposed system offers several advantages over traditional approaches:
Accurate and efficient detection of duplicate bug reports
Automated and intelligent bug tracking support system
Improved handling of textual variations using NLP techniques
Real-time detection using a Streamlit-based interface
Scalable and practical solution for software development environments
Overall, the system provides a reliable and efficient framework for duplicate bug detection, addressing key challenges in traditional manual bug tracking methods.

G. Diagram (You can include in your paper)

You can recreate this in Word / draw.io:

User Input Bug Description
↓
Text Preprocessing
(cleaning, tokenization, stopwords)
↓
TF-IDF Vectorization
↓
Cosine Similarity
(compare with all bugs)
↓
Similarity Score Calculation
↓
Threshold Filtering (>0.30)
↓
Duplicate Detection Result
(High / Medium / Low / No Match)
↓
Streamlit UI Display

V. RESEARCH METHODOLOGY

To address the limitations of traditional bug tracking methods, this paper proposes a machine learning-based approach for accurate and efficient duplicate bug detection. The methodology integrates text preprocessing, feature extraction, similarity computation, and evaluation techniques to ensure reliable and scalable detection. The approach focuses on improving detection accuracy while maintaining computational efficiency for real-time software development applications.

A. System Design

The proposed system follows a structured machine learning pipeline consisting of data collection, preprocessing, similarity computation, and prediction stages. Bug report data is processed and transformed into a suitable format for analysis. The system integrates text processing techniques and a user-friendly interface to provide real-time duplicate detection and decision support for developers.

B. Data Preparation and Preprocessing

The dataset undergoes several preprocessing steps to improve data quality and detection performance. Irrelevant characters, symbols, and inconsistencies are handled appropriately, and text normalization techniques such as lowercasing and tokenization are applied. Stopword removal and stemming or lemmatization are used to refine the textual data. Variations in language are addressed to ensure consistency across bug descriptions, improving model generalization.

C. Feature Engineering and Selection

Feature extraction is performed to identify and retain the most relevant textual attributes for detection. TF-IDF (Term Frequency–Inverse Document Frequency) is used to convert text into numerical vectors, capturing the importance of words across the dataset. This ensures that significant and meaningful words contribute effectively to similarity computation, improving detection accuracy and system stability.

D. Model Training and Implementation

The system implements similarity-based techniques rather than traditional classification models. TF-IDF vectorization is applied to the processed dataset to generate vector representations of bug reports. Cosine similarity is then used to compare vectors and measure similarity between bug descriptions. Threshold-based filtering is applied to classify similarity levels and improve detection accuracy.

E. Model Evaluation Metrics

The performance of the system is evaluated using similarity scores and detection effectiveness. Metrics such as accuracy, precision, recall, and F1-score are considered to assess the quality of duplicate detection. These metrics provide a comprehensive evaluation of system performance, particularly in handling variations in textual data. Comparative analysis is conducted to determine optimal similarity thresholds for reliable detection.

F. Prediction and Deployment

Fig. 2. User interface for real-time duplicate bug detection.

The final system is integrated into a Streamlit-based web application that allows users to input bug descriptions and obtain real-time detection results. The system classifies bug reports into different similarity levels and provides interpretable outputs. This enables developers to identify duplicate issues quickly and efficiently.

G. System Reliability

The proposed system ensures reliability through robust preprocessing, effective text representation using TF-IDF, and consistent similarity measurement using cosine similarity. These techniques improve detection consistency and reduce errors. The system is designed to generalize well across different types of bug reports, making it suitable for practical software development environments.

H. Performance Evaluation

The performance of the system is analyzed based on detection accuracy, response time, and overall efficiency. Experimental results demonstrate that the proposed approach achieves improved detection performance and better generalization compared to traditional manual methods. The system also provides fast response times, making it suitable for real-time bug tracking applications.

VI. ANALYSIS OF RESULTS

The performance of the proposed machine learning-based duplicate bug detection system is evaluated using multiple evaluation metrics such as accuracy, precision, recall, F1-score, and overall detection efficiency. The analysis demonstrates how preprocessing techniques, feature extraction, and similarity computation contribute to improved detection performance and reliability of the system.

A. Accuracy Analysis

Fig. 3. Accuracy comparison of similarity-based detection for duplicate bug reports.

Accuracy is a primary metric used to evaluate the overall correctness of the detection system. Various similarity-based approaches, including TF-IDF with cosine similarity, are analyzed for effectiveness. Experimental results indicate that the proposed approach achieves high accuracy in identifying duplicate bug reports. This improvement is attributed to its ability to capture meaningful relationships between words in textual bug descriptions.

B. Precision and Recall Analysis

Precision and recall are critical metrics for evaluating detection performance, especially in duplicate identification where incorrect matches can affect development workflows. Precision measures the correctness of detected duplicates, while recall evaluates the system's ability to identify all actual duplicate reports. The results show that effective preprocessing and similarity threshold tuning achieve balanced precision and recall, ensuring improved detection without significantly increasing false matches.

C. Efficiency Improvement

The efficiency of the system is enhanced through preprocessing techniques such as text cleaning and normalization, which improve similarity computation and reduce processing time. Additionally, lightweight approaches such as TF-IDF and cosine similarity provide faster detection responses compared to complex machine learning models. The system demonstrates improved computational efficiency and scalability, making it suitable for real-time duplicate bug detection in software development environments.

D. Reliability and Performance Analysis

The proposed system achieves high reliability through consistent detection performance across different evaluation metrics. The integration of preprocessing, feature extraction, and similarity-based methods ensures stable and accurate results. Compared to traditional manual approaches, the system provides better generalization and reduced detection errors. The results confirm that the proposed approach is effective for identifying duplicate bug reports and supports efficient bug tracking processes.

E. Model Interpretability and Decision Support

The proposed system provides interpretable results that assist developers in understanding the similarity between bug reports. Similarity scores and confidence levels are used to indicate the strength of matches between descriptions. These insights enable developers to make informed decisions and improve issue management. The interpretability of the system ensures that results are transparent and easily understandable, increasing trust and usability in real-world applications.

F. Data Privacy and Security

To ensure data privacy, the system processes bug report data without exposing sensitive or confidential information. Only textual content relevant to detection is used, and no personal or sensitive identifiers are stored within the system. This approach minimizes the risk of data leakage and maintains confidentiality. Additionally, secure data handling practices are followed to ensure that data remains protected during processing and analysis.

G. System Robustness and Reliability

The system is designed to minimize errors and improve robustness through effective preprocessing techniques and optimized similarity computation. Techniques such as text normalization and feature extraction enhance system stability and reduce bias. Threshold tuning further improves performance by refining similarity classification. These measures ensure that the system delivers consistent and reliable detection results across different datasets and scenarios.

H. Overall Performance Assessment

The integration of preprocessing, feature extraction, and similarity-based techniques results in a highly efficient and accurate duplicate bug detection system. The proposed approach demonstrates improved performance compared to traditional manual methods in terms of accuracy, detection speed, and reliability. The system effectively supports identification of duplicate bug reports and provides a scalable solution for software development environments with high confidence.

VII. CONCLUSION

In this paper, we proposed a machine learning-based framework for accurate duplicate bug detection using textual bug report data, addressing the limitations of traditional manual bug tracking approaches. The system integrates preprocessing techniques, feature extraction, and similarity-based methods to improve detection accuracy and reliability. By automating the analysis of bug descriptions, the proposed method supports faster identification of duplicate issues and reduces dependency on manual inspection, thereby enhancing software maintenance processes and development efficiency.

One of the key contributions of this work is the effective use of text preprocessing techniques along with TF-IDF vectorization and cosine similarity to improve detection performance. These techniques enhance the system's ability to identify similar bug reports while maintaining balanced precision and recall. In addition, the deployment of the system through a Streamlit-based interface enables real-time detection and provides interpretable similarity results, making it practical and accessible for developers in real-world environments.

The experimental results demonstrate that the proposed system achieves improved accuracy, efficiency, and reliability compared to traditional methods. The system shows strong generalization capability and consistent performance across evaluation metrics such as accuracy, precision, recall, and F1-score. Overall, the proposed approach offers a scalable, efficient, and reliable solution for duplicate bug detection, contributing to improved bug tracking processes and supporting faster issue resolution in software development workflows.

VIII. FUTURE WORKS

Although the proposed machine learning-based duplicate bug detection system demonstrates high accuracy, efficiency, and reliability, there are still several areas for improvement to enhance its performance and applicability in real-world software development environments.

To begin with, scalability can be improved by incorporating larger and more diverse bug report datasets from multiple platforms, enabling the system to generalize better across different software projects and reporting styles. Integration with real-time bug tracking systems such as issue management tools and version control platforms can further enhance data availability and improve detection accuracy in dynamic development environments.

Interoperability can be enhanced by developing standardized APIs and middleware solutions that allow seamless integration with software development tools, project management systems, and collaborative platforms. This would enable continuous monitoring of bug reports and support decision-making across different development environments.

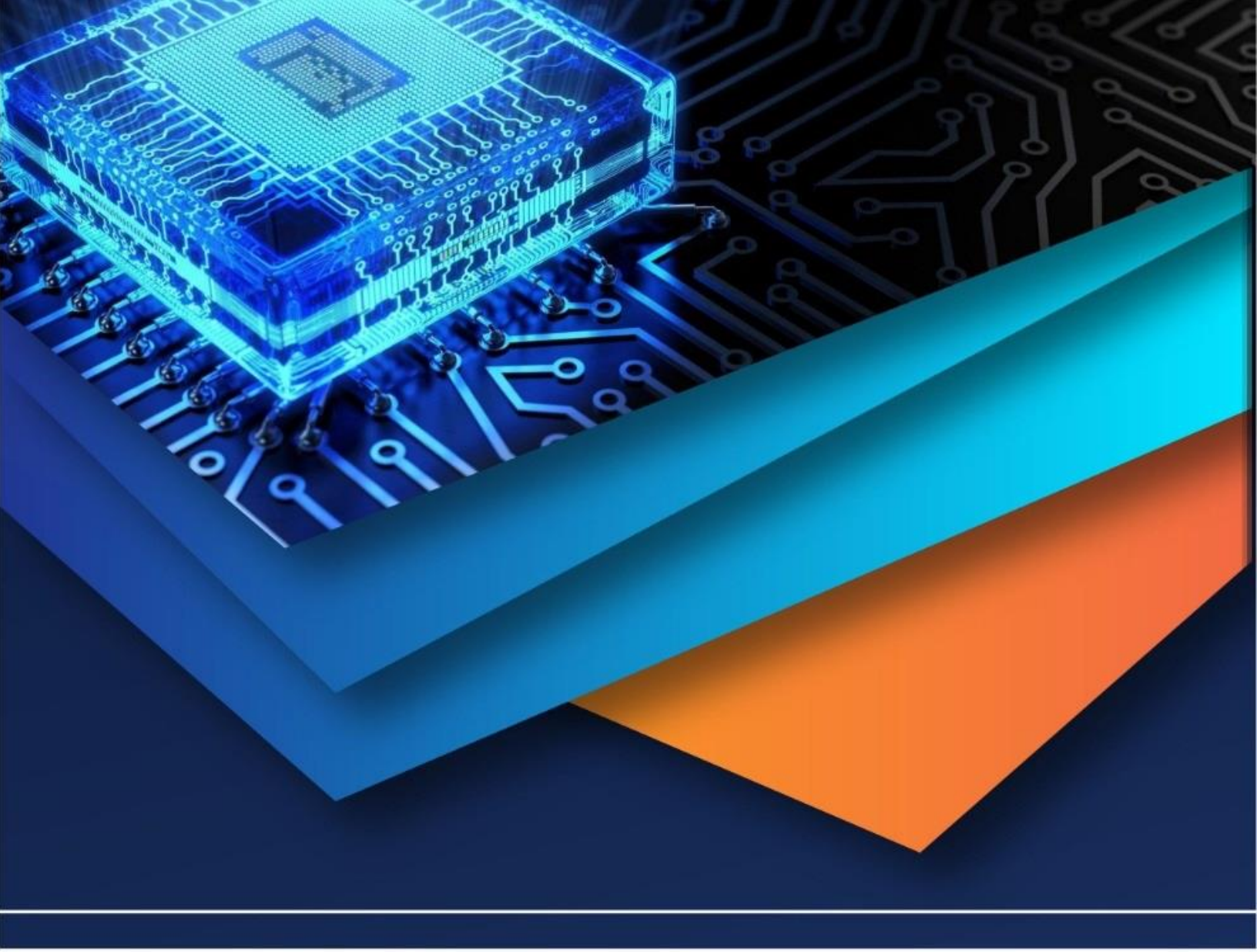
The system can also be extended to support multi-language bug detection by incorporating multilingual text processing techniques, allowing it to handle bug reports written in different languages. This expansion would increase the applicability of the system across global development teams and provide a more comprehensive bug tracking solution.

Future work may include the use of advanced techniques such as deep learning models and transformer-based architectures to further improve detection accuracy and semantic understanding of bug descriptions. Additionally, optimization of similarity computation and deployment strategies can reduce computational cost and improve response time.

Integration of mobile and web-based interfaces with enhanced visualization features can improve user experience and accessibility for developers. Finally, real-world testing and deployment in collaboration with software organizations can provide valuable feedback on system performance, usability, and reliability, enabling continuous refinement and improvement of the proposed solution.

REFERENCES

[1] T. Mikolov et al., "Efficient estimation of word representations in vector space," in Proc. International Conference on Learning Representations (ICLR), 2013.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
[3] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.
[4] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," Journal of Documentation, vol. 28, no. 1, pp. 11–21, 1972.
[5] F. Sebastiani, "Machine learning in automated text categorization," ACM Computing Surveys, vol. 34, no. 1, pp. 1–47, 2002.
[6] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.
[7] S. Banerjee, K. C. K. Li, and R. K. Saha, "Duplicate bug report detection using natural language processing," in Proc. IEEE International Conference on Software Maintenance, 2012.
[8] N. Jalbert and W. Weimer, "Automated duplicate detection for bug tracking systems," in Proc. IEEE International Conference on Dependable Systems and Networks, 2008.
[9] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language processing," in Proc. ICSE Workshop, 2008.
[10] A. Hindle, D. M. German, and R. Holt, "What do large commits tell us? A taxonomical study of large commits," in Proc. MSR, 2008.
[11] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. ICSE, 2006.
[12] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in Stack Overflow," in Proc. ASE, 2013.
[13] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," 2023. [Online]. Available**:** **https://scikit-learn.org**
[14] NLTK Project, "Natural Language Toolkit," 2023. [Online]. Available: https://www.nltk.org
[15] Streamlit Inc., "Streamlit: Data App Framework," 2024. [Online]. Available: https://streamlit.io



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY