Journal of Information Systems Engineering and Management
2025, 10(26s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Comparative Study of Multilabel Classification Techniques for Analyzing Bug Report Dependencies

Zhengdong Hu, Jantima Polpinij*, Gamgarn Somprasertsri
Department of Computer Science, Faculty of Informatics, Mahasarakham University, Maha Sarakham Province, Thailand.
**Corresponding Author: jantima.p@msu.ac.th*

ARTICLE INFO

Received: 26 Dec 2024

Revised: 16 Feb 2025

Accepted: 01 Mar 2025

ABSTRACT

Bug report dependency analysis entails identifying and examining the interrelations among software bug reports. Dependencies may indicate that bugs are interconnected, with one bug obstructing the resolution of another. Consequently, one software defect must be rectified prior to resolving another. To our knowledge, a baseline has been developed for textual similarity-based grouping and keyword matching. Regrettably, this usually fails to comprehensively represent the complicated associations among bug reports, resulting in ineffective debugging and heightened maintenance expenses. As a results, this study presents an alternate way to enhance bug dependency analysis via extensive multilabel classification methods. The dataset, obtained from Bugzilla, comprises 4,781 bug reports pertaining to Mozilla Firefox, with each report linked to one to four dependency labels. A thorough comparison of Binary Relevance, Classifier Chains, and Label Powerset was performed with classifiers like Multinomial Naïve Bayes (MNB), K-Nearest Neighbor (KNN), Random Forest (RF), and Support Vector Machine (SVM). Furthermore, deep learning architectures including LSTM and TextCNN, as well as transformer-based models like BERT and RoBERTa, were assessed. Although multilabel classifiers developed by machine learning exhibit strong performance, they encounter issues with class imbalance, which adversely impacts F1-scores despite elevated overall accuracy. The experiment's findings indicate that BERT surpasses all other models and the baseline, attaining the greatest F1-score (0.647) and Micro-averaged Accuracy (0.9967), underscoring its efficacy in identifying semantic relationships within bug reports. The results demonstrate that transformer-based models return the most efficient approach for identifying bug report dependencies, hence enhancing problem triaging and automating software maintenance.

Keywords: Bug report, bug dependency, multilabel classification technique, machine learning, deep learning, transformer learning

INTRODUCTION

Bug dependency occurs when flaws or defects in one part of the software affect the functionality of other interconnected components. A critical defect can disrupt the operation of related functions, leading to system-wide errors and inefficiencies. In large-scale software projects, defects are typically reported through "bug reports." However, the sheer volume of bug reports complicates the process of analyzing dependencies among them. Eventually, the performance of bug fixing can be optimized by understanding the dependencies between bug reports, which can offer meaningful insights into the dependencies of software bugs [1], [2], [3]. Hence, the domain of bug report dependency analysis has observed growth in attention in both research and practice. To the best of our knowledge, existing techniques for identifying bug dependencies can be text-based similarity analysis, clustering, and keyword matching. Unfortunately, it is difficult in successfully identifying bug dependency because of complicated dependencies between bug reports. Therefore, it is not surprised that this difficulty can lead to challenges in ineffective debugging or bug fixing task, bug dependency identification, and handling of high costs for software maintenance. As a result, the performance of bug report dependency analysis needs to be should be improved through the application of more reliable and stable methods [4], [5], [6].

Despite progress in software debugging, numerous current methodologies encounter difficulties in precisely recognizing and categorizing multilabel relationships due to their intricacy. Traditional classification approaches, such as TF-IDF analysis and fundamental clustering algorithms, are inadequate for modeling the complex relationships among bug reports [7], [8], [9]. These mentioned methods often consider dependencies as isolated cases rather than related situations, inadequately representing the fundamental point of software failures. In addition, It is also well-known that datasets from a bug tracking system are imbalanced because particular dependency labels occur more frequently than others. This can inhibit the capability of machine learning models to perform successfully. [10], [11]. To overcome these problems, credible classification algorithms are required to improve performance of identification of bug dependencies in broad software systems [12], [13], [14].

The objective of this study is to examine multilabel classification methods for analyzing dependencies in bug reports through a comparison of machine learning, deep learning, and transformer learning algorithms. Machine learning algorithms such as MNB, KNN, RF, and SVM are utilized alongside multilabel methods such as Binary Relevance [15], Classifier Chains [16], and Label Powerset [17]. Furthermore, this study also compares those traditional models to deep learning algorithms, i.e., LSTM [18] and TextCNN [19], and transformer learning algorithms, i.e., BERT [20] and RoBERTa [21], in their individual ways in order to obtain the most appropriate multilabel classification model. Finally, these models used for bug report dependency analysis are evaluated by F1-score, Micro-averaged Accuracy, and AUC.

This research is significant for debugging and software maintenance through bug dependency analysis using multilabel classification techniques. Additionally, it may facilitate bug triaging, prioritize repairs, and minimize program downtime. Beyond software debugging, multilabel classification algorithms have the potential to be applied to project management and task scheduling, offering valuable insights into dependency analysis across various domains. Enhancing the efficiency, scalability, and precision of bug report dependency management can ultimately improve software engineering practices and support intelligent software maintenance solutions [22], [23], [24].

The organization of this paper can be described as follows. Section 2 presents the dataset used in this study, and research methods can be explained in Section 3. Section 4 mentions to the experimental results and discussion. Finally, summary of this study and future work can be presented as Section 5.

DATASET

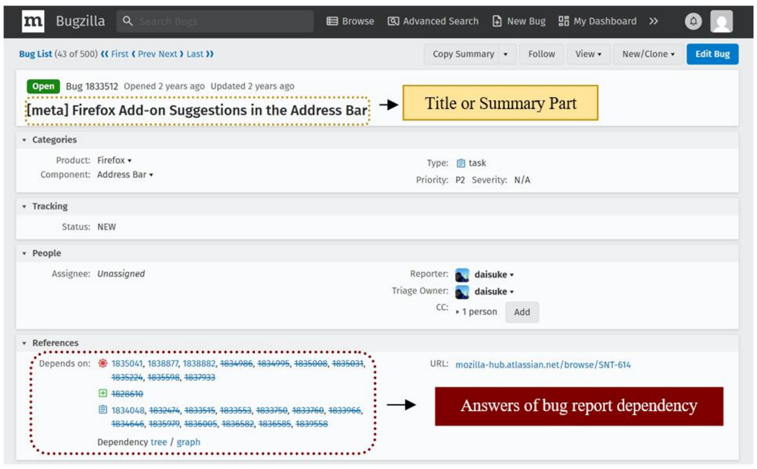
This analysis uses Mozilla Firefox bug reports from Bugzilla from September 1–November 30, 2019. Bug reports total 4,781, including 200 meta-bugs. Meta-bugs link all other bug reports, establishing a network. Each bug report has one to four class labels indicating dependence types. Bug reports may depend on meta-bug reports, according to this dataset. Meta-bug reports can establish a tree-like hierarchy. The experiment used all bug report types except UNCONFIRMED to assure data accuracy. Because bug triagers, software developers, and software testers verified the reported issues were real. Figure 1 shows a Bugzilla-style bug report. This shows dataset dependencies' arrangement and preservation. Figure 2 shows an XML bug report, revealing data representation in different formats.

METHODS

This study employs a systematic research methodology that methodically advances through bug report preparation, model selection, and evaluation. The procedure commences with preprocessing, during which raw bug reports are sanitized and standardized to guarantee uniformity in the dataset. Subsequently, model selection entails the identification and use of appropriate classification methodologies, encompassing classical, deep learning, and transformer-based strategies. Ultimately, an evaluation is performed to assess the efficacy of the chosen models in effectively finding and forecasting bug dependencies. The research framework can be illustrated in Figure 3.

1. Bug Report Pre-processing

To prepare bug reports for analysis, different pre-processing techniques are applied based on the model type. For machine learning models, the text undergoes lowercasing, text correction, tokenization, stop-word removal, special character removal, and stemming, ensuring consistency and reducing noise.



Bugzilla Browse Advanced Search New Bug My Dashboard

Bug List (43 of 500) First Prev Next Last

Copy Summary Follow View New/Clone Edit Bug

Open Bug 1833512 Opened 2 years ago Updated 2 years ago

[meta] Firefox Add-on Suggestions in the Address Bar

Categories

Product: Firefox
Component: Address Bar
Type: task
Priority: P2 Severity: N/A

Tracking

Status: NEW

People

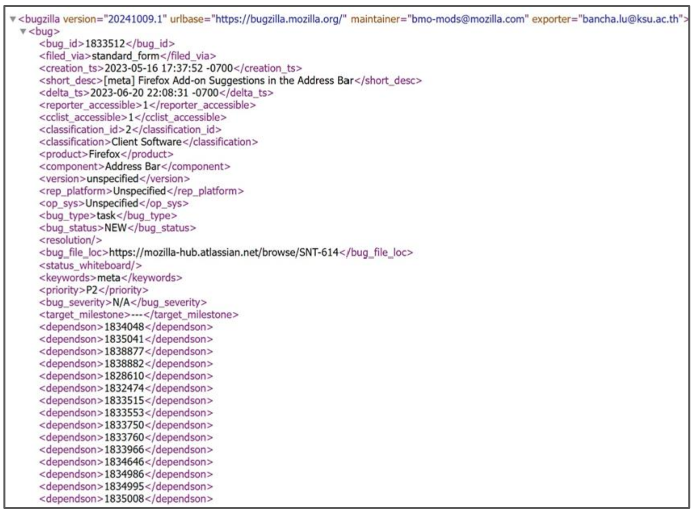
Assignee: *Unassigned*
Reporter: daisuke
Triage Owner: daisuke
CC: 1 person Add

References

Depends on: 1835041, 1838877, 1838882, ~~1834986~~, ~~1834995~~, ~~1835008~~, ~~1835031~~, ~~1835224~~, ~~1835598~~, ~~1837933~~
~~1828610~~
1834048, ~~1832474~~, ~~1833515~~, ~~1833553~~, ~~1833750~~, ~~1833760~~, ~~1833966~~, ~~1834646~~, ~~1835979~~, ~~1836005~~, ~~1836582~~, ~~1836585~~, ~~1839558~~
Dependency tree / graph

URL: mozilla-hub.atlassian.net/browse/SNT-614

Figure 1: An example of bug reports representing with the Bugzilla format



```
▼<bugzilla version="20241009.1" urlbase="https://bugzilla.mozilla.org/" maintainer="bmo-mods@mozilla.com" exporter="bancha.lu@ksu.ac.th">
  ▼<bug>
      <bug_id>1833512</bug_id>
      <filed_via>standard_form</filed_via>
      <creation_ts>2023-05-16 17:37:52 -0700</creation_ts>
      <short_desc>[meta] Firefox Add-on Suggestions in the Address Bar</short_desc>
      <delta_ts>2023-06-20 22:08:31 -0700</delta_ts>
      <reporter_accessible>1</reporter_accessible>
      <cclist_accessible>1</cclist_accessible>
      <classification_id>2</classification_id>
      <classification>Client Software</classification>
      <product>Firefox</product>
      <component>Address Bar</component>
      <version>unspecified</version>
      <rep_platform>Unspecified</rep_platform>
      <op_sys>Unspecified</op_sys>
      <bug_type>task</bug_type>
      <bug_status>NEW</bug_status>
      <resolution/>
      <bug_file_loc>https://mozilla-hub.atlassian.net/browse/SNT-614</bug_file_loc>
      <status_whiteboard/>
      <keywords>meta</keywords>
      <priority>P2</priority>
      <bug_severity>N/A</bug_severity>
      <target_milestone>---</target_milestone>
      <dependson>1834048</dependson>
      <dependson>1835041</dependson>
      <dependson>1838877</dependson>
      <dependson>1838882</dependson>
      <dependson>1828610</dependson>
      <dependson>1832474</dependson>
      <dependson>1833515</dependson>
      <dependson>1833553</dependson>
      <dependson>1833750</dependson>
      <dependson>1833760</dependson>
      <dependson>1833966</dependson>
      <dependson>1834646</dependson>
      <dependson>1834986</dependson>
      <dependson>1834995</dependson>
      <dependson>1835008</dependson>
```

Figure 2: An example of bug report representing with XML format

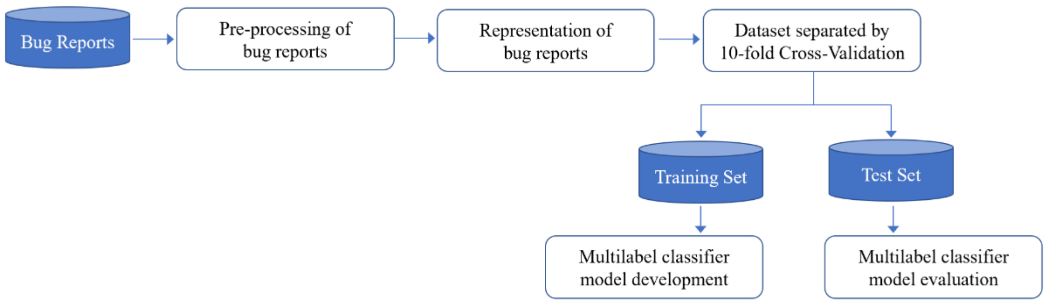

Bug Reports
Pre-processing of bug reports
Representation of bug reports
Dataset separated by 10-fold Cross-Validation
Training Set
Test Set
Multilabel classifier model development
Multilabel classifier model evaluation

Figure 3. The research framework

In deep learning models, tokenization, lowercasing, stop-word elimination, and lemmatization are employed to efficiently structure the data, followed by the application of a word embedding technique such as Word2Vec to transform text data into numerical representations. This facilitates deep learning models to determine word semantic associations.

For transformer learning application, slight text modifications are necessary for maintaining contextual meaning in transformer learning. Key processing steps text cleaning, processing CamelCase and SnakeCase words, tokenizing with BERT's WordPiece, padding, truncation, converting to input IDs, and generating attention masks are peformed respectively. These processes enable conversion of the input into a suitable structure for classification. This enhances the accuracy of the model in analyzing bug report dependencies through the application of a suitable framework.

2. Multilabel Classification Models

This study examines a number of multilabel classification methods to evaluate how well they identify the dependency of bug reports. Traditional methods, such as Binary Relevance, Classifier Chains, and Label Powerset, are utilized as multilabel text classifiers through various machine learning algorithms, including Support Vector Machine (SVM), Random Forest (RF), Multinomial Naïve Bayes (MNB), and K-Nearest Neighbor (KNN). This study further assesses deep learning methodologies, specifically Long Short-Term Memory (LSTM) and TextCNN, to determine their effectiveness in identifying complicated structures within text data. Additionally, the contextual learning abilities of transformer learning—more especially, BERT base and RoBERTa—are evaluated. Each of these approaches presents distinct benefits in dependency classification, providing valuable insights into the most effective methods for enhancing bug triaging and software debugging.

In multilabel classification, different approaches are employed to handle bug report dependencies, each offering unique strengths and limitations. Binary Relevance (BR) treats each dependency label as an independent binary classification problem. While this method is simple and computationally efficient, it does not account for relationships between labels, potentially leading to misclassification when dependencies are interconnected. Classifier Chains (CC) address this limitation by predicting labels sequentially, allowing dependencies to be captured. However, because each label prediction influences subsequent ones, errors can propagate through the chain, affecting overall accuracy. Another alternative, Label Powerset (LP), converts multilabel classification into a multiclass problem, effectively capturing label correlations. However, this approach requires a significantly larger dataset for training, as the number of possible label combinations increases.

Beyond traditional methods, deep learning techniques, such as Long Short-Term Memory (LSTM) and TextCNN, offer advanced solutions for dependency classification. LSTM, a type of recurrent neural network, is well-suited for sequential text processing, enabling the model to capture long-term dependencies in bug reports. This makes it effective for recognizing patterns across lengthy descriptions of software issues. Meanwhile, TextCNN applies convolutional layers to extract key textual features, making it particularly efficient in identifying dependencies through local context patterns. Both deep learning models improve classification performance compared to traditional approaches, as they learn from contextual and sequential information within the text.

Further enhancing classification accuracy, BERT-based models leverage transformer-based contextual embeddings to understand semantic relationships in bug reports. Unlike LSTM and TextCNN, which rely on sequential processing and local feature extraction, BERT captures bidirectional context, enabling a deeper understanding of dependencies within the text. By considering the broader context, transformer-based models significantly outperform conventional approaches, providing superior accuracy in bug dependency classification and improving software debugging efficiency.

3. Model Evaluation

F1-score, Micro-averaged Accuracy, and Area Under the Curve are used as evaluation metrics to measure the performance of the proposed multilabel classification models in identifying of bug report dependencies.

A comparison was conducted between machine learning, deep learning, and transformer learning approaches against a baseline utilizing the BM25-based clustering approach to determine their advantages [25]. The proposed multilabel classification methods more effectively identify bug report dependencies, as demonstrated in this comparison.

EXPERIMENTAL RESULTS AND DISCUSSION

The experimental results are presented in this section, followed by a discussion of the efficacy comparison of multilabel classification techniques and a comparison of their efficacy with the baseline. For bug report dependency analysis, Table 1 shows the results of every multilabel classification model, including the baseline's results [25]. When compared to the baseline with clustering approach (BM25), those results can provide significant insights into how well a number of multilabel classification models analyze the dependencies in bug reports. Although BM25 has a high Micro-averaged Accuracy (0.9881) and AUC (0.821), its F1-score (0.226) is markedly inferior to those of other supervised models. This inconsistency underscores the shortcomings of unsupervised clustering, which, although identifying general document similarities, fails to provide the accuracy required for accurate dependency classification.

In contrast, supervised machine learning models consistently outperform BM25 across all evaluation metrics, demonstrating the advantage of training with labeled data. Among traditional Binary Relevance models, KNN (F1 = 0.552, AUC = 0.865) and Random Forest (F1 = 0.532, AUC = 0.943) achieve higher F1-scores than SVM (F1 = 0.509, AUC = 0.957) and Multinomial Naïve Bayes (F1 = 0.355, AUC = 0.933). The relatively weak performance of Naïve Bayes suggests that its independence assumption is unsuitable for modeling interdependent bug report labels.

Table 1: Performance Evaluation of the Baseline Model and Supervised Learning Approaches

Mothods	**F1-Score**	**Micro-averaged Accuracy**	**AUC**
Baseline: Clustering with BM25	0.226	0.9881	0.821
Binary relevance using SVM	0.509	0.9963	0.957
Binary relevance using Multinomial Naive Bayes	0.355	0.9950	0.933
Binary relevance using KNN	0.552	0.9962	0.865
Binary relevance using Random Forest	0.532	0.9957	0.943
Classifier chains using SVM	0.556	0.9961	0.955
Classifier chains using Multinomial Naive Bayes	0.365	0.9954	0.934
Classifier chains using KNN	0.509	0.9947	0.901
Classifier chains using Random Forest	0.527	0.9960	0.951
Label Powerset using SVM	0.387	0.9936	0.914
Label Powerset using Multinomial Naive Bayes	0.571	0.9955	0.974
Label Powerset using KNN	0.586	0.9957	0.918
Label Powerset using Random Forest	0.599	0.9958	0.963
LSTM	0.339	0.9941	0.895
TextCNN	0.489	0.9960	0.881
BERT	0.647	0.9967	0.951
RoBERTa	0.600	0.9961	0.953

When modeling label dependencies more explicitly, Classifier Chains and Label Powerset approaches improve classification performance in different ways. Classifier Chains with SVM (F1 = 0.556, AUC = 0.955) performs slightly better than its Binary Relevance counterpart, indicating that sequential label prediction helps capture interdependencies. However, the risk of error propagation in Classifier Chains means that models relying on this approach can suffer from compounding classification mistakes. Label Powerset with Random Forest (F1 = 0.599, AUC = 0.963) achieves the highest F1-score among traditional machine learning methods, suggesting that treating multilabel classification as a single multiclass problem provides a meaningful advantage in capturing label relationships.

Deep learning models show varied performance, with LSTM (F1 = 0.339, AUC = 0.895) performing worse than most machine learning models. Its sequential nature likely limits its effectiveness in handling complex dependencies within bug reports, as long-range relationships may not be fully captured. TextCNN (F1 = 0.489, AUC = 0.881) improves over LSTM, benefiting from convolutional layers that extract key textual features efficiently. However, both models still lag behind top-performing traditional classifiers, suggesting that further tuning or additional training data may be required to maximize their potential.

The best overall performance is achieved by transformer-based models, with BERT (F1 = 0.647, AUC = 0.951) surpassing all other methods, followed closely by RoBERTa (F1 = 0.600, AUC = 0.953). These models leverage contextual embeddings, allowing them to capture semantic relationships within bug reports more effectively than previous methods. Their superior ability to process long-range dependencies and learn from large-scale textual data makes them well-suited for bug dependency classification.

Despite significant variations in F1-score, Micro-averaged Accuracy remains consistently high (≥ 0.99) across all supervised models. This is because some dependency labels show up a lot more often than others, causing an unbalance in label distribution. The general accuracy is high because models tend to focus on common labels. However, they have trouble correctly classifying rarer labels, which lowers the F1-score. This effect is stronger in datasets that aren't fair, where even small mistakes in classifying fewer common classes have a big effect on the F1-

score. However, AUC stays high (≥0.85) across all models. This means that even when the F1-score is low, the models are still good at telling the difference between relationships that are important and those that are not.

In summary, traditional machine learning models are good at setting strong foundations, but transformer-based models are the best at accurately classifying things. The fact that Micro-averaged Accuracy and AUC are always high across all models shows that classification correctness is stable at a high level. However, class imbalance is still a big problem that affects F1-score. In the future, researchers might look into ways to improve performance even more and deal with problems like imbalanced classification, such as adding more data, using class-weighted loss functions, or oversampling of minority groups.

CONCLUSION AND FUTURE WORK

This work offers a comparative examination of multilabel classification algorithms for bug report dependency analysis, assessing classic machine learning methods, deep learning approaches, and transformer-based models. The results reveal considerable disparities in performance among various strategies, illustrating the advantages and drawbacks of each method in precisely recognizing and forecasting relationships among problem reports. The experimental findings indicate that supervised classification models substantially surpass the baseline BM25 clustering approach, which, although attaining a high Micro-averaged Accuracy, is hindered by a poor F1-score due to its ineffectiveness in modeling intricate dependency interactions. Among conventional machine learning methods, Label Powerset with Random Forest achieves the greatest F1-score, leveraging its superior capacity to extract label correlations compared to Binary Relevance and Classifier Chains. Nevertheless, these conventional methods continue to encounter difficulties in addressing class imbalance and in capturing contextual linkages within textual descriptions.

Deep learning models like LSTM and TextCNN exhibit enhancements compared to conventional methods, however they remain inferior to transformer-based models. LSTM, although proficient in handling sequential data, encounters difficulties in capturing long-range dependencies, whereas TextCNN excels in extracting significant local text features. Nonetheless, both models necessitate additional optimization to rival the most effective categorization methods.

The most significant enhancement arises from transformer-based models, with BERT attaining the highest overall performance, closely succeeded by RoBERTa. Their capacity to utilize contextual embeddings and bidirectional attention processes allows for the capturing of intricate relationships inside bug reports, rendering them the most efficacious ways for dependency categorization. These models markedly enhance classification accuracy, especially in forecasting infrequent dependence labels.

Considering the robust performance of supervised models, class imbalance persists as a significant difficulty, adversely impacting F1-scores despite constantly elevated Micro-averaged Accuracy and AUC. Subsequent study ought to investigate methodologies such data augmentation, class-weighted loss functions, or oversampling of minority classes to further improve classification efficacy. This study conclusively shows that transformer-based methodologies, especially BERT, offer the most effective option for multilabel bug report dependency classification, enhancing software debugging and maintenance efficiency.

Future study should examine methodologies to mitigate class imbalance, including data augmentation, class-weighted loss functions, and oversampling techniques. Moreover, integrating hybrid models that amalgamate classical, deep learning, and transformer-based methodologies could further augment performance. Augmenting the dataset and assessing multilingual bug reports may enhance model generalization and applicability.

ACKNOWLEDGEMENT

This work was financially supported by Mahasarakham University.

REFERENCES

[1] G. Barreto, P. D. Battaglin, and S. Varga, “Ensuring Efficient IT Service Management to Increase Information Systems Availability,” *Journal of Information Systems Engineering & Management*, vol. 4, no. 4, Dec. 2019, doi: 10.29333/jisem/6352.

[2] I. Pashchenko, D.-L. Vu, and F. Massacci, "A Qualitative Study of Dependency Management and Its Security Implications," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA: ACM, Oct. 2020, pp. 1513–1531. doi: 10.1145/3372297.3417232.

[3] B. Luaphol, J. Polpinij, M. Kaenampornpan, "Automatic dependent bug reports assembly for bug tracking systems by threshold-based sim*ilarity," Indonesian Journal of Electrical Engi-neering and Computer Science, vol. 23, no. 3, p. 1620-1633, Sep. 2021, doi: 10.11591/ijeecs.v23.i3.pp1620-1633*.

[4] M. M. Morovati, A. Nikanjam, F. Tambon, F. Khomh, and Z. M. Jiang, "Bug characterization in machine learning-based systems," *Empir Softw Eng*, vol. 29, no. 1, p. 14, Jan. 2024, doi: 10.1007/s10664-023-10400-0.

[5] I. Saidani, A. Ouni, M. Ahasanuzzaman, S. Hassan, M. W. Mkaouer, and A. E. Hassan, "Tracking bad updates in mobile apps: a search-based approach," *Empir Softw Eng*, vol. 27, no. 4, p. 81, Jul. 2022, doi: 10.1007/s10664-022-10125-6.

[6] N. Sharma and P. Yalla, "Keyphrase Extraction And Source Code Similarity Detection-A Survey," *IOP Conf Ser Mater Sci Eng*, vol. 1074, no. 1, p. 012027, Feb. 2021, doi: 10.1088/1757-899X/1074/1/012027.

[7] A. L. Alem, K. K. Gebretsadik, S. A. Mengistie, and M. F. Admas, "Multi-label software requirement smells classification using deep learning," *Sci Rep*, vol. 15, no. 1, p. 5761, Feb. 2025, doi: 10.1038/s41598-025-86673-w.

[8] W. N. I. Al-Obaydy, H. A. Hashim, Y. A. Najm, and A. A. Jalal, "Document classification using term frequency-inverse document frequency and K-means clustering," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 3, p. 1517, Sep. 2022, doi: 10.11591/ijeecs.v27.i3.pp1517-1524.

[9] X. Wang, J. Huang, C. Ye, and H. Zhou, "GlobalTagNet: A Graph-Based Framework for Multi-Label Classification in GitHub Issues," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, IEEE, Jun. 2024, pp. 67–78. doi: 10.1109/RE59067.2024.00017.

[10] R. H. Pereira, M. J. Gonçalves, and M. A. G. Magalhães, "Reputation Systems: A framework for attacks and frauds classification," *Journal of Information Systems Engineering and Management*, vol. 8, no. 1, p. 19218, Jan. 2023, doi: 10.55267/iadt.07.12830.

[11] H. Tu, Z. Yu, and T. Menzies, "Better Data Labelling With EMBLEM (and how that Impacts Defect Prediction)," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 278–294, Jan. 2022, doi: 10.1109/TSE.2020.2986415.

[12] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools," *Eng Appl Artif Intell*, vol. 111, p. 104773, May 2022, doi: 10.1016/j.engappai.2022.104773.

[13] M. Pasha, G. Qaiser, and U. Pasha, "A Critical Analysis of Software Risk Management Techniques in Large Scale Systems," *IEEE Access*, vol. 6, pp. 12412–12424, 2018, doi: 10.1109/ACCESS.2018.2805862.

[14] Z. Shen and S. Chen, "A Survey of Automatic Software Vulnerability Detection, Program Repair, and Defect Prediction Techniques," *Security and Communication Networks*, vol. 2020, pp. 1–16, Sep. 2020, doi: 10.1155/2020/8858010.

[15] M. L. Zhang, Y. K. Li, X. Y. Liu, and X. Geng. "Binary relevance for multi-label learning: an overview," Frontiers of Computer Science, vol. 12, no. 2, pp. 191-202, March 2018, doi: 10.1007/s11704-017-7031-7.

[16] J. Read, B. Pfahringer, G. Holmes, and E. Frank, (2021). "Classifier chains: a review and perspectives," Journal of Artificial Intelligence Research, vol. 70, pp. 683-718, Feb. 2021.

[17] S. Al-Maadeed, (2013, November). "Kernel collaborative label power set system for multi-label classification," In Qatar Foundation Annual Research Forum, vol. 2013, no. 1, pp. ICTP-028, Nov. 2012, doi: 10.5339/qfarf.2013.ICTP-028.

[18] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," IEEE transactions on neural networks and learning systems, vol. 28, no. 10, pp. 2222-2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[19] T. Zhang, F. You, "Research on short text classification based on textcnn," In Journal of Physics: Conference Series, vol. 1757, no. 1, pp. 012092, 2021, doi: 10.1088/1742-6596/1757/1/012092.

[20] V. Tong, C. Dao, H.-A. Tran, T. X. Tran, and S. Souihi, "Enhancing BERT-Based Language Model for Multi-label Vulnerability Detection of Smart Contract in Blockchain," *Journal of Network and Systems Management*, vol. 32, no. 3, p. 63, Jul. 2024, doi: 10.1007/s10922-024-09832-w.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” arXiv preprint arXiv:1907.11692.

[22] D. Groen *et al.*, “FabSim3: An automation toolkit for verified simulations using high performance computing,” *Comput Phys Commun*, vol. 283, p. 108596, Feb. 2023, doi: 10.1016/j.cpc.2022.108596.

[23] T. Mårtensson, A. Martini, D. Ståhl, and J. Bosch, “Excellence in Exploratory Testing: Success Factors in Large-Scale Industry Projects,” 2019, pp. 299–314. doi: 10.1007/978-3-030-35333-9_21.

[24] H. Zhang and M. O. Shafiq, “Survey of transformers and towards ensemble learning using transformers for natural language processing,” *J Big Data*, vol. 11, no. 1, p. 25, Feb. 2024, doi: 10.1186/s40537-023-00842-0

[25] B. Luaphol, J. Polpinij, M. Kaenampornpan, “Text mining approaches for dependent bug report assembly and severity prediction”. Int. Arab J. Inf. Technol., vol. 19, no. 6, pp. 915-924.