Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 1035-1042

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.1424
p-ISSN: 2723-3863
e-ISSN: 2723-3871

COMPARISON OF SUPPORT VECTOR MACHINE AND INDOBERT IN NON-FUNCTIONAL REQUIREMENT CLASSIFICATION OF APPLICATION USER REVIEWS

Andul Ghofur Rais Kumar[*1], Yudi Sukmono[2], Aji Ery Burhandenny[3]

[1,2]Industrial Engineering, Faculty of Engineering, Universitas Mulawarman, Indonesia
[3]Electrical engineering, Faculty of Engineering, Universitas Mulawarman, Indonesia
Email: [1]raisghofur07@gmail.com, [2]y.sukmono@ft.unmul.ac.id, [3]a.burhandenny@ft.unmul.ac.id

(Article received: September 28, 2023; Revision: October 24, 2023; published: July 29, 2024)

Abstract

User reviews of mobile applications have become a valuable source of information for evaluating the quality of an application. It is crucial for application developers to understand what users express in their reviews. One aspect that can be analyzed from user reviews is Non-Functional Requirement (NFR). Classifying reviews based on NFR is essential in understanding how an application can be enhanced. Although user reviews have the potential to provide valuable insights into NFR, manually processing thousands of user reviews is a laborious and inefficient task. Therefore, artificial intelligence methods are employed to automatically classify user reviews into relevant NFR categories. This research discusses the performance comparison of the SVM and IndoBERT algorithms in NFR classification. The study involves collecting application review data from 2018 to 2023, sourced from Google Playstore and Apple Appstore, followed by annotating the review data based on ISO 25010. Subsequently, the data is allocated into training and testing sets with an 80:20 ratio. Further, a data preprocessing phase is conducted, which includes steps such as lowercasing, tokenization, special character removal, text normalization, and text stemming. The next step involves training the SVM and IndoBERT algorithms on the dataset. Finally, the evaluation is carried out by calculating the F1-score. The research results indicate that the IndoBERT model outperforms the SVM model. The IndoBERT algorithm excels in recognizing NFR in reviews, achieving an F1-score of 93%, while the SVM algorithm achieves an F1-score of 91%.

Keywords: *IndoBERT, Natural Language Processing, Non-functional Requirement, Support Vector Machine, User Reviews.*

1. INTRODUCTION

At present, we are in the era of digitalization, where software applications have become an integral aspect of our daily routines. As stated in [1], Indonesia boasts a staggering 67.88% of its population, which translates to 192.15 million individuals, using smartphones. This signifies a growing reliance on smartphones and diverse software solutions to fulfill a wide array of requirements.

The quality of an application is a crucial aspect in providing an optimal user experience [2]. Therefore, it is essential for application developers to understand what is conveyed by user reviews. One aspect that can be analyzed from user reviews is Non-Functional Requirement (NFR) [3].

Although user reviews have the potential to provide valuable insights into NFR, manually processing thousands of user reviews is a laborious and inefficient task [4]. Therefore, there is a need to develop automated methods that can classify user reviews into relevant NFR categories [5].

AI can enhance collaboration, automation, productivity, and data analysis capabilities, helping industry players to remain at the forefront [6]. Many text mining techniques are employed in various tasks such as classification, clustering, document summarization, and more. Generally, text mining techniques aim to provide data in natural language processing [7].

Natural language processing focuses on developing a model that can facilitate interaction between computers and human language. Various natural language processing applications include speech recognition, spoken language understanding, dialogue systems, machine translation, chatbots, sentiment analysis, natural language summarization, and many more [8].

Research on the classification of requirement criteria based on user reviews has been conducted several times. Research by [9][5][10] analyzed non-functional requirements (NFR) based on user reviews in Android applications. Research by [11] evaluated the quality of mHealth applications based on the ISO/IEC 25010 quality model through user feedback analysis. Furthermore, research by [12] discussed multi-label classification of application quality in user reviews.

According to [13], the support vector machine (SVM) method is a classification method, and it has been used to solve problems in various fields, both in the medical and meteorological fields, to address issues related to gene expression analysis and financial problems. In the field of sentiment analysis itself, the implementation of the support vector machine (SVM) method is common. This method can provide better results compared to similar classification methods such as artificial neural networks. This is because SVM is capable of finding globally optimal solutions. This is because SVM is capable of finding globally optimal solutions. Support vector machine algorithm can be seen in Figure 1.

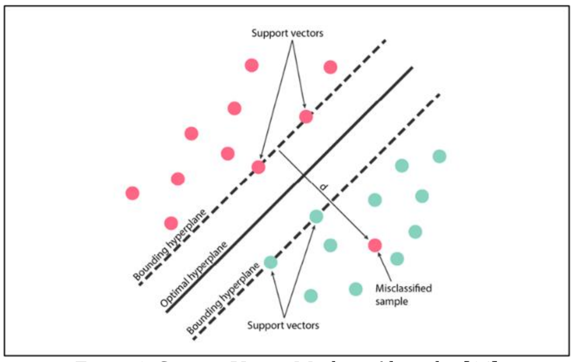


Figure 1. Support Vector Machine Algorithm[14]

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained neural network-based technique for Natural Language Processing (NLP). BERT is designed to perform bidirectional representation of text data, allowing it to capture context from both directions. Therefore, the BERT model only requires a single output layer to create various NLP models such as question-answering, without substantial architectural modifications [15].

IndoBERT is an evolution of the BERT architecture specifically trained for the use of Indonesian language corpus data. The Indonesian language dataset used for training comprises 4 billion words in both formal and everyday conversational language, resulting from the amalgamation of 12 Indonesian language corpus datasets. This dataset is then trained on the standard BERT architecture, which includes 12 transformer layers, 768 hidden layers, and 12 attention heads, resulting in 124.5 million parameters [16].

As stated in [17], requirement engineering entails a methodical and structured process for defining and overseeing requirements, with the following key aims:

a. Gaining insight into pertinent requirements, facilitating consensus among stakeholders regarding suggested requirements, adhering to prescribed standards for documentation, and implementing systematic management.
b. Understanding and recording the preferences and needs of stakeholders, as well as defining and handling these requirements to reduce the potential risk of a system not aligning with the stakeholders' preferences and needs.

2. RESEARCH METHODS

2.1. Data Collection

The data source used in the context of this research includes user reviews related to applications available on the Google Play Store and Apple App Store platforms. The data utilized in this study is the result of reviews contributed by users for the top five most downloaded applications in five different categories. The data retrieval process involves the use of the Google Play Scrapper and Appstore Scrapper libraries.

The data focused on in this research comprises historical data spanning from 2018 to 2023. Data collection during this period was designed with the aim of obtaining a comprehensive overview of user reviews over several relevant periods, namely, before, during, and post-pandemic. The pre-pandemic period, covering from 2018 to 2019, provides insights into user trends and preferences before the COVID-19 pandemic struck. Subsequently, the pandemic period, which lasted from 2020 to 2021, reflects changes in user behavior and needs during the health crisis. Finally, the post-pandemic period, from 2022 to 2023, reflects the adaptation and evolution that occurred in the app ecosystem after the pandemic.

The research primarily targets five application categories: ojek online, e-commerce, digital wallets, e-ticketing platforms, and streaming applications. These selections were made due to their widespread popularity and substantial user engagement levels on prominent app distribution platforms like Google Play Store and Apple App Store. Consequently, the extensive data gathering within these chosen categories aims to furnish comprehensive insights into the role of user reviews across diverse application domains in evaluating non-functional requirements, as outlined within the scope of this study.

2.2. Anotation Dataset

The next step in this research is the dataset annotation phase. Dataset annotation refers to the process of adding labels or tags to existing data sets, with the aim of classifying or enriching the information contained in the data. This process is carried out using manual classification methods, involving the researchers in assigning labels or tags to 10,000 user reviews of applications. The classification covers various aspects, such as performance, usability, security, reliability, and other, in accordance with the ISO 25010 framework.

2.3. Preprocessing

The next step is to preprocess the dataset, as mentioned by [18], This process includes

Lowercasing, Punctuation removal, Removing Stop Words, Standardizing Text, Tokenizing Text, and Stemming.

Lowercasing is employed to transform all text letters into lowercase. The objective of this method is to enhance text processing efficiency, eliminating any concerns related to the differentiation between uppercase and lowercase letters..

Punctuation Removal is employed to eliminate punctuation marks from the text. This is done because punctuation marks do not significantly contribute to text analysis and can be disruptive during processing. Removing punctuation marks streamlines the text for more efficient analysis.

Removing Stop Words is applied to eliminate words that do not contribute significantly to text analysis, known as stop words. Stop words typically encompass frequently used words such as "and," "or," "I," "you," and so on. This method aims to expedite the text processing procedure and enhance the quality of analytical outcomes by prioritizing meaningful content.

Standardizing Text is used to standardize text by replacing abbreviations or non-standard words with their standard or canonical forms. This technique is crucial for ensuring consistency in the text and facilitating the processing process, particularly in cases where different variations of words are used.

Stemming is the process of removing prefixes and suffixes from words to obtain their base or root form. For example, "running," "runner," and "ran" would all be reduced to the base form "run." The goal is to simplify word searching within documents and reduce the dimensionality of word representations, making the analysis more efficient and effective.

Tokenization refers to breaking the text into meaningful small units or tokens. During the tokenization process, text in the form of sentences is divided into individual words. Tokenization is an essential step in text preprocessing for various types of analyses. In the case of application review data, tokenization is performed as a fundamental step to prepare the text for subsequent processing and analysis.

2.4. Design and Model Testing

The next step involves splitting the dataset into two parts: Training set and test set. The division of the dataset helps in minimizing both underfitting and evaluating the model's performance. The training data makes up 80% of the dataset while the test data comprises 20%.

Subsequently, the next step entails designing a machine learning model using Support Vector Machine (SVM). On the other hand, the design of deep learning models is carried out using the IndoBERT algorithm with a transfer learning approach.

This data partitioning strategy ensures that the machine learning and deep learning models can be trained on a substantial portion of the data while also having a separate dataset reserved for evaluation. It is essential for assessing the models' generalization capabilities and their ability to perform well on new, unseen data, thus contributing to the robustness and reliability of the research findings.

The subsequent phase encompasses the model evaluation, encompassing both training and testing phases. The testing phase involves the computation of the Confusion matrix and evaluation metrics. The Confusion matrix is a matrix comprised of elements employed to assess the effectiveness of a machine learning model. Some values within the confusion matrix include True Positive (TP), True Negative(TN), False Positive(FP), and False Negative(FN) [19]. The values from the confusion matrix are used to calculate precision, recall, accuracy, and F1 score in the evaluation matrix.

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

In equation (1), precision measures the proportion of true positive (TP) predictions out of all positive predictions (TP + FP). Precision quantifies the accuracy of positive predictions made by a model, indicating how often the model correctly identifies true positive cases while minimizing false positives.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

In equation (2), recall calculates the proportion of true positive predictions (TP) out of all actual positive cases (TP + FN). Recall assesses a model's ability to capture all relevant positive cases, minimizing false negatives.

$$F1\,Score = \frac{2{*}TP}{2{*}TP + FN + FP} \tag{3}$$

In equation (3), F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both aspects. F1 Score particularly useful when there is an uneven class distribution or a need to optimize both precision and recall simultaneously.

$$Accuracy = \frac{TP+ TN}{TP + FN + TN+ FP} \tag{4}$$

In equation (4), accuracy measures the proportion of correctly predicted cases (TP + TN) out of the total number of cases (TP + FN + TN + FP). Accuracy provides an overall assessment of a model's correctness, but it may not be the best metric for imbalanced datasets.

3. RESULTS

3.1. Data Collection

The data collection step yielded user reviews data for various types of applications, including online motorcycle ojek online, e-commerce apps, e-wallet apps, e-ticket apps, and streaming apps. The total number of reviews collected for each category was as follows: 1,734,991 user reviews for ojek apps, 1,250,152 user reviews for e-wallet apps, 1,655,962 user reviews for e-commerce apps, 110,864 user reviews for e-ticket apps, and 23,343 user reviews for streaming apps.

3.2. Annotation Dataset

The researcher manually classified 10,000 random samples of user reviews from each application. This classification encompassed various aspects, such as performance, usability, security, and reliability, in accordance with the ISO 25010 framework. Results of annotation dataset can be seen in Table 1.

Table 1. Annotation Dataset

Platform	Category					Total
	Performance	*Reliability*	*Usability*	*Security*	Other	
E-wallet	400	400	400	400	400	2000
Ojek *Online*	400	400	400	400	400	2000
E-Ticket	400	400	400	400	400	2000
E-Commerce	400	400	400	400	400	2000
Streaming	400	400	400	400	400	2000
Total	2000	2000	2000	2000	2000	10000

3.3. Preprocessing

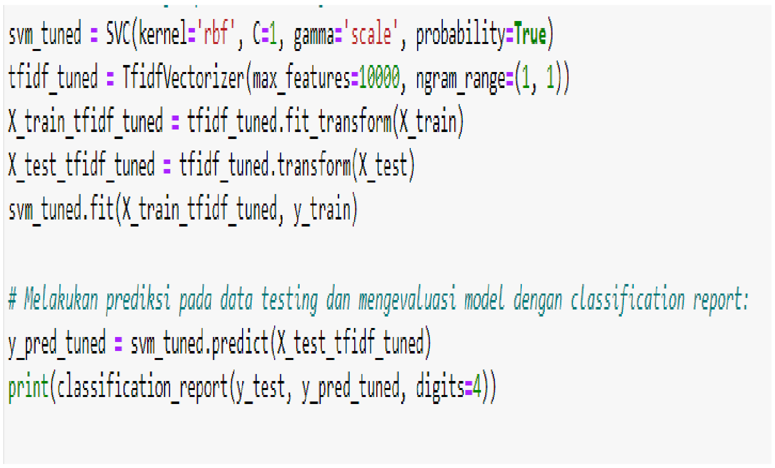
The first step after obtaining the dataset in the form of scraped data is to preprocess the dataset. This process includes Lowercasing, Punctuation removal, Removing Stop Words, Standardizing Text, Tokenizing Text, and Stemming. Sample step of preprocessing dataset can be seen in table 2.

Table 2. Preprocessing Dataset

Step	Before	After
Lowercasing	User interface nya plissss dong jgn jauh melenceng dr sebelum nya ...Pusing tau nyari2.. Apa lagi model ibu2 punya anak.. Nenek2 ngaji.. Ribet bgt jempol mesti ngapalin lagi.	user interface nya plissss dong jgn jauh melenceng dr sebelum nya ...pusing tau nyari2.. apa lagi model ibu2 punya anak..nenek2 ngaji.. ribet bgt jempol mesti ngapalin lagi.
Punctuation Removal	user interface nya plissss dong jgn jauh melenceng dr sebelum nya ...pusing tau nyari2.. apa lagi model ibu2 punya anak..nenek2 ngaji.. ribet bgt jempol mesti ngapalin lagi	user interface nya plissss dong jgn jauh melenceng dr sebelum nya pusing tau nyari apa lagi model ibu punya anak nenek ngaji ribet bgt jempol mesti ngapalin lagi
Removing Stopwords	user interface nya plissss dong jgn jauh melenceng dr sebelum nya pusing tau nyari apa lagi model ibu punya anak nenek ngaji ribet bgt jempol mesti ngapalin lagi	user interface nya melenceng sebelum nya pusing nyari model ibu punya anak nenek ngaji ribet jempol
standardization	user interface nya jauh melenceng sebelum nya pusing nyari apa lagi model ibu punya anak nenek ngaji ribet jempol mesti ngapalin	user interfacenya melenceng sebelumnya pusing nyari model ibu memiliki anak nenek ngaji susah jempol
Stemming	user interfacenya jauh melenceng sebelumnya pusing nyari apa lagi model ibu memiliki anak nenek ngaji susah jempol harus apa	user interface melenceng belum pusing cari model ibu milik anak nenek ngaji susah jempol
Tokenization	user interface jauh melenceng belum pusing cari apa lagi model ibu milik anak nenek ngaji susah jempol harus apa	[user, interface, melenceng, belum, pusing, cari, model, ibu, milik, anak, nenek, ngaji, susah, jempol]

3.4. Training and Evaluation

Data processing after going through the SVM model creation stage involves training and testing the SVM model, which is conducted using the Python programming language with the assistance of the Keras library. The codes of SVM model and evaluation can be seen in Figure 2.



```
svm_tuned = SVC(kernel='rbf', C=1, gamma='scale', probability=True)
tfidf_tuned = TfidfVectorizer(max_features=10000, ngram_range=(1, 1))
X_train_tfidf_tuned = tfidf_tuned.fit_transform(X_train)
X_test_tfidf_tuned = tfidf_tuned.transform(X_test)
svm_tuned.fit(X_train_tfidf_tuned, y_train)

# Melakukan prediksi pada data testing dan mengevaluasi model dengan classification report:
y_pred_tuned = svm_tuned.predict(X_test_tfidf_tuned)
print(classification_report(y_test, y_pred_tuned, digits=4))
```

Figure 2. Line of Code Support Vector Machine

The following are the results of the confusion matrix in the training process using the Support Vector Machine algorithm, showing precision, recall, f1-score, support, accuracy, macro average, and weighted average. Results of evaluation matrix of SVM model can be seen in Table 3.

Table 3. Evaluation Matrix Support Vector Machine

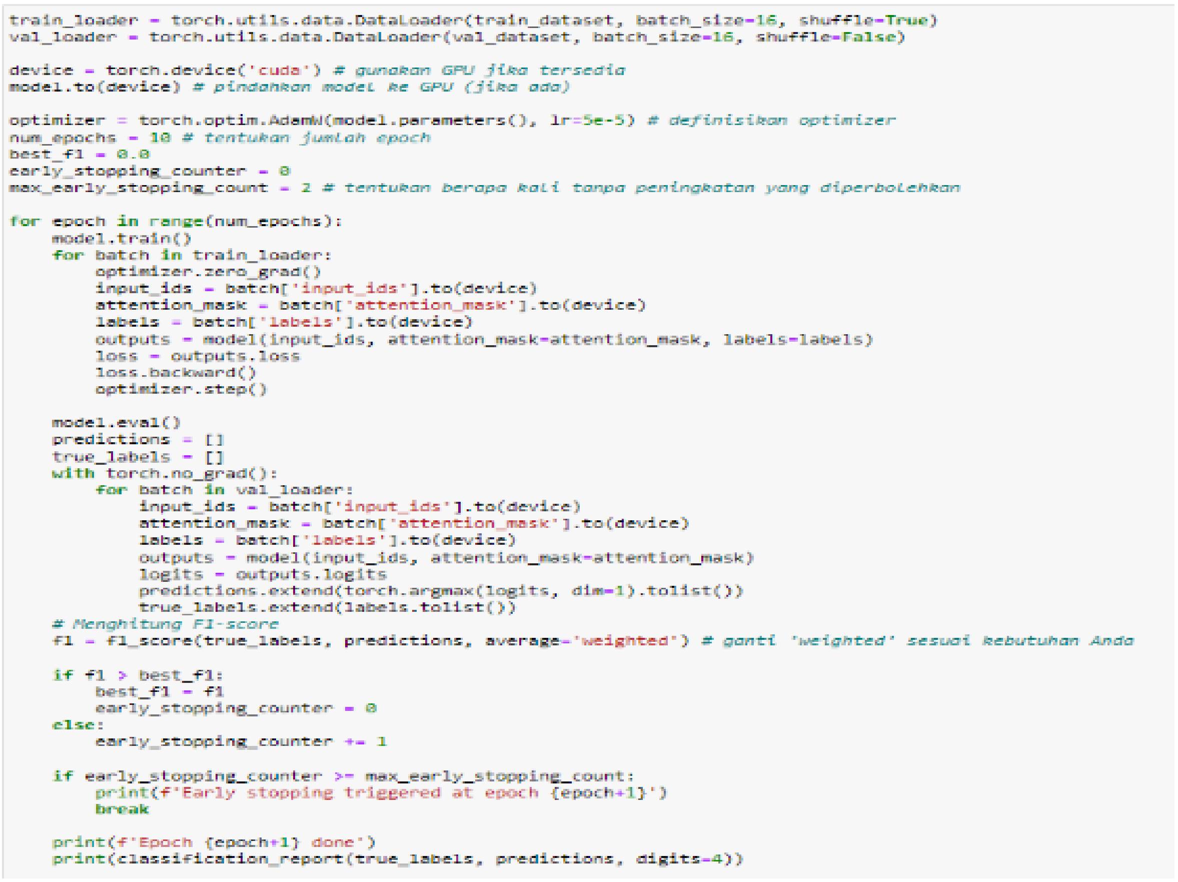
Category	Precision	Recall	F1-Score
Other	0.9570	0.9356	0.9462
Performance	0.8843	0.9348	0.9089
Reliability	0.9272	0.8536	0.8889
Security	0.9390	0.9436	0.9413
Usability	0.8920	0.9305	0.9108
Accuracy			0.9195
Macro Avg	0.9199	0.9196	0.9192
Weighted Avg	0.9204	0.9195	0.9194

The model demonstrates the highe-st level of precision in predicting the "other" class, with a rate of 95%. This is followe-d by the "security" class with a precision rate- of 93%, the "reliability" class with 92%, and the "usability" class with 89%. The- lowest precision rate is observed in predicting the "performance" class, which stands at 88%.

The class with the highest recall rate is "security" at 94%. It is closely followed by the "other," "performance," and "usability" classes, all with a re-call rate of 93%. The "reliability" class has a slightly lower recall rate at 85%. This indicates that the- model is particularly effective- in detecting data in the "other" class but struggles relatively more- with detecting data in the "pe-rformance" class.

Among the different classes, the "other" class achieved the highe-st F1-score at 94%. It was closely followed by the- "security" class and the "usability" class, which scored 94% and 91%, re-spectively. The mode-l performed well in predicting these categorie-s. However, it had a slightly lower performance in predicting the "performance" class at 90% and the "reliability" class at 88%.

The resulting accuracy is 0.91 or 91%. This indicates that the model can correctly predict approximately 91% of the overall test data. The codes of IndoBERT model and evaluation can be seen in Figure 3.



```
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=16, shuffle=False)

device = torch.device('cuda') # gunakan GPU jika tersedia
model.to(device) # pindahkan model ke GPU (jika ada)

optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5) # definisikan optimizer
num_epochs = 10 # tentukan jumlah epoch
best_f1 = 0.0
early_stopping_counter = 0
max_early_stopping_count = 2 # tentukan berapa kali tanpa peningkatan yang diperbolehkan

for epoch in range(num_epochs):
    model.train()
    for batch in train_loader:
        optimizer.zero_grad()
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

    model.eval()
    predictions = []
    true_labels = []
    with torch.no_grad():
        for batch in val_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels'].to(device)
            outputs = model(input_ids, attention_mask=attention_mask)
            logits = outputs.logits
            predictions.extend(torch.argmax(logits, dim=1).tolist())
            true_labels.extend(labels.tolist())
    # Menghitung F1-score
    f1 = f1_score(true_labels, predictions, average='weighted') # ganti 'weighted' sesuai kebutuhan Anda

    if f1 > best_f1:
        best_f1 = f1
        early_stopping_counter = 0
    else:
        early_stopping_counter += 1

    if early_stopping_counter >= max_early_stopping_count:
        print(f'Early stopping triggered at epoch {epoch+1}')
        break

    print(f'Epoch {epoch+1} done')
    print(classification_report(true_labels, predictions, digits=4))
```

Figure 3. Line of Code IndoBERT

The following are the results of the confusion matrix in the training process using the IndoBERT algorithm, showing precision, recall, F1-score, support, accuracy, macro average, and weighted average. Results of evaluation matrix of IndoBERT model can be seen in Table 4.

The model demonstrates the highest precision in the "others" class, with an accuracy rate of 97%. The reliability and security classes also have high precision rates of 97%. The usability class follows closely behind with a precision rate of 91%, while the performance class has a slightly lower precision leve-l of 86%.

Table 4. Evaluation Matrix IndoBERT

Category	Precision	Recall	F1-Score
Other	0.9797	0.9579	0.9687
Performance	0.8606	0.9565	0.9060
Reliability	0.9745	0.8536	0.9101
Security	0.9700	0.9510	0.9604
Usability	0.9142	0.9712	0.9419
Accuracy			0.9380
Macro Avg	0.9398	0.9380	0.9374
Weighted Avg	0.9411	0.9380	0.9381

The “usability” class has the highest recall at 97%, closely followed by the "others" class, “performance” class, “security” class, and “reliability” class with recall rates of 95% each. This indicates that the model is generally more proficient at identifying data in the usability category; however, it exhibits a lower detection rate in the performance class.

The "others" class has the F1 score at 96% followed closely by the security class with 96% well. The usability class achieves a score of 94% while the reliability class follows with 91%. Lastly the performance class shows the score at 90%. These results suggest that the model excels, in predicting the "others" category but struggles comparatively in predicting performance.

The resulting accuracy is 0.93 or 93%. This shows that the model can correctly predict approximately 93% of the overall test data.

4. DISCUSSION

The research conducted by [8] titled "A Practical User Feedback Classifier for Software Quality Characteristics" developed an approach by testing various Machine Learning (ML) algorithms, features, and class balancing techniques for classifying user feedback in a dataset consisting of 1500 reviews. The maximum F1 and F2 scores obtained were 60% and 73%, with a recall rate reaching 94%. This approach does not replace human experts but reduces the effort required for requirements elicitation.

The research by [5] titled "Non-Functional Requirements Analysis Based on Application Reviews in the Android App Market" aims to analyze non-functional requirements (NFR) based on user reviews of Android applications in the Android app market, which hosts over 3 million applications. In this article, user comments are automatically classified into non-functional requirements (NFR) and other types. The research proposes the loop matching classification (LMC) technique. Three classification techniques, namely LMC, BOW, and TF-IDF, are employed to classify user comments, and the accuracy, recall rate, and F-measure of the results of these three classification techniques are compared. The research results indicate that the Precision value of the LMC classification technique is 74.2%, the Recall rate is 82.5%, and the F-measure is 78.1%.

The research conducted by [9] with the title "Automatic Classification of Apps Reviews for Requirement Engineering: Exploring The Customers Need from Healthcare Applications." This research utilizes the Multinomial Naïve Bayes method to classify requirement engineering in user reviews based on the ISO25010 matrix and achieves the highest F1 Score of 84%.

The research titled "Machine learning for mHealth apps quality evaluation: An approach based on user feedback analysis" by [10] employed the Stochastic Gradient Descent (SGD) machine learning algorithm to assess the quality of mHealth applications based on the ISO/IEC 25010 quality model through user feedback analysis. For this purpose, a total of 1682 user reviews were collected from 86 mHealth applications available on the Google Play Store. The achieved accuracy rate was 82%.

The research conducted by [11] with the title "Automated Classification of Mobile App Reviews Considering User’s Quality Concerns" discusses the multi-label classification of application quality in user reviews. This study trained a multi-label review classification model using a convolutional neural network (MLRC-CNN) on pre-trained word embeddings (known as word2vec embeddings) for the classification of reviews from ten thousand user reviews extracted from fifty Android applications. The research resulted in an average accuracy of 85%.

The impressive levels of accuracy and F1 Scores achieved in the four preceding studies can be attributed to factors, including the quality of the dataset utilized and the careful selection of algorithms and parameters. However it is important to note that these studies are not, without limitations. For instance they were conducted using a amount of data and focused solely on reviews written in English. As a result it becomes imperative to conduct research with a dataset and develop models based on Indonesian language corpus in order to ensure more dependable outcomes. This study distinguishes itself significantly from previous research by utilizing the Support Vector Machine and IndoBERT methods as a comparative tool, employing a larger dataset, and incorporating the use of the Indonesian language corpus. Additionally, this research classifies reviews into five main categories: performance, reliability, security, usability, and others, making it a comprehensive multiclass classification.

5. CONCLUSION

Based on the comparison between the two algorithms, it is evident that IndoBERT consistently exhibits higher precision, recall, and F1-score values compared to SVM for nearly all classes. This indicates that IndoBERT excels in classifying user reviews more effectively in the context of NFR classification. Despite SVM achieving relatively high accuracy, IndoBERT outperforms it in terms of measuring precision, recall, and F1-score, which can be considered more relevant indicators in this context. The performance of the Support Vector Machine in classifying application review data resulted in an F1 Score of 0.9194, precision of 0.9204, recall of 0.9195, and an accuracy of 0.9195. Meanwhile, the performance of IndoBERT in classifying application review data yielded an F1 Score of 0.9381, precision of 0.9411, recall of 0.9380, and an accuracy of 0.9380. Based on the generated output, it can be observed that overall, the IndoBERT model demonstrates superior performance compared to SVM.

REFERENCES

[1] Badan Pusat Statistik, *Statistik Telekomunikasi Indonesia 2022*. Jakarta: Badan Pusat Statistik, 2023.

[2] M. Nixon, M. Jose, and A. K. Asst Professor, “App store bugs-review classification using BERT- DNN model,” *Turkish J. Comput. Math. Educ.*, vol. 12, no. 13, pp. 5300–5306, 2021, [Online]. Available: https://www.turcomat.org/index.php/turkbilmat/article/view/9721.

[3] D. Dave and V. Anu, “Identifying Functional and Non-functional Software Requirements From User App Reviews,” in *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2022, pp. 1–6, doi: 10.1109/IEMTRONICS55184.2022.9795770.

[4] J. Y. Lee, “User Review Mining: An Approach for Software Requirements Evolution,” *Int. J. Adv. Smart Converg.*, vol. 9, no. 4, pp. 124–131, 2020, [Online]. Available:http://dx.doi.org/10.7236/IJASC.2020.9.4.124.

[5] Y. Yao, W. Jiang, Y. Wang, P. Song, and B. Wang, “Non-Functional Requirements Analysis Based on Application Reviews in the Android App Market,” *Inf. Resour. Manag. J.*, vol. 35, no. 2, pp. 1–17, 2022, [Online]. Available: https://ideas.repec.org/a/igg/rmj000/v35y2022i2p1-17.html.

[6] M. Zarlis, Elviwani, A. Dilham, Relita, and V. Yasin, *Kecerdasan Buatan Era Revolusi Industri 4.0*, 1st ed. Bogor: Mitra Wacana Media, 2022.

[7] Preeti, “Review on Text Mining: Techniques, Applications and Issues,” *Proc. 2021 10th Int. Conf. Syst. Model. Adv. Res. Trends, SMART 2021*, vol. 7, no. 11, pp. 474–478, 2021, doi: 10.1109/SMART52563.2021.9676285.

[8] Deng, L., Liu, Y. Deep Learning in Natural Language Processing, Springer, Singapore. 2022. https://doi.org/10.1007/978-981-10-5209-5_11

[9] R. dos Santos, K. Villela, D. T. Avila, and L. H. Thom, “A practical user feedback classifier for software quality characteristics,” *Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE*, vol. 2021-July, no. May, pp. 340–345, 2021, doi: 10.18293/SEKE2021-055.

[10] N. Al Kilani, R. Tailakh, and A. Hanani, “Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications,” in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019, pp. 541–548, doi: 10.1109/SNAMS.2019.8931820.

[11] M. Haoues, R. Mokni, and A. Sellami, “Machine learning for mHealth apps quality evaluation,” Software Quality Journal, May 2023, doi: https://doi.org/10.1007/s11219-023-09630-8.

[12] B. Rehman, K.A. Alam, K.M., Ko." Automated Classification of Mobile App Reviews Considering User’s Quality Concerns," Advances in Intelligent Systems and Computing, vol. 1357, June 2021, doi: https://doi.org/10.1007/978-981-16-1045-5_3

[13] S. A. Amira and M. I. Irawan, “Opinion Analysis of Traveler Based on Tourism Site Review Using Sentiment Analysis,” IPTEK The Journal for Technology and Science, vol. 31, no. 2, p. 223, May 2020, doi: https://doi.org/10.12962/j20882033.v31i2.6338.

[14] J. Cardoso-Fernandes, A. C. Teodoro, A. Lima, and E. Roda-Robles, “Semi-Automatization of Support Vector Machines to Map Lithium (Li) Bearing Pegmatites,” Remote Sensing, vol. 12, no. 14, p. 2319, Jul. 2020, doi: https://doi.org/10.3390/rs12142319.

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, pp. 4171–4186, 2019.

[16] B. Wilie *et al.*, “IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural

Language Understanding," 2020, [Online]. Available: http://arxiv.org/abs/2009.05387.

[17] K. Pohl and C. Rupp, Requirements engineering fundamentals : a study guide for the certified professional for requirements engineering exam ; foundation level - IREB compliant. Santa Barbara, Calif.: Rocky Nook, 2015.

[18] A. Kulkarni, NATURAL LANGUAGE PROCESSING RECIPES : unlocking text data with machine learning and deep ... learning using python. 2019.

[19] S. Mokhtari, K. K. Yen, and J. Liu, "Effectiveness of Artificial Intelligence in Stock Market Prediction based on Machine Learning," International Journal of Computer Applications, vol. 183, no. 7, pp. 1–8, Jun. 2021, doi: https://doi.org/10.5120/ijca2021921347.