User Opinion Mining on the Maxim Application Reviews Using BERT-Base Multilingual Uncased

Sindy Eka Safitri[1], Wenty Dwi Yuniarti [2], Maya Rini Handayani [3] , Khothibul Umam[4]*
Teknologi Informasi, Fakultas Sains dan Teknologi[1], [2], [3], [4]
UIN Walisongo Semarang
Semarang, Indonesia
2208096018@student.walisongo.ac.id[1], wenty@walisongo.ac.id[2], maya@walisongo.ac.id[3],
khothibul_umam@walisongo.ac.id[4]

***Abstract*— Online transportation applications such as Maxim are increasingly used due to the convenience they offer in ordering services. As usage increases, the number of user reviews also grows, serving as a valuable source of information for evaluating customer satisfaction and service quality. Sentiment analysis of these reviews can help companies understand user perceptions and improve service quality. This study aims to analyze the sentiment of user reviews on the Maxim application using the BERT-Base Multilingual Uncased model. BERT was chosen for its ability to understand sentence context bidirectionally, and it has proven to outperform traditional models such as MultinomialNB and SVM in previous studies, with an accuracy of 75.6%. The dataset used consists of 10,000 user reviews with an imbalanced distribution: 4,000 negative, 2,000 neutral, and 4,000 positive reviews. The data was split into 90% training data (9,000 reviews) and 10% test data (1,000 reviews). From the 9,000 training data, 15% or 1,350 reviews were allocated as validation data, resulting in a final training set of 7,650 reviews. Evaluation results show that BERT is capable of classifying sentiment into three categories positive, neutral, and negative, with an accuracy of 94.7%. The highest F1-score was achieved in the positive class (0.9621), followed by the neutral class (0.9412), and the negative class (0.9246). The confusion matrix shows that most predictions match the actual labels. These findings indicate that BERT is an effective and reliable model for performing sentiment analysis on user reviews of online transportation applications such as Maxim.**

Keywords- sentiment analysis, app reviews, BERT, Maxim, text classification.

I. INTRODUCTION

Currently, the rapid development of digital technology has increased the need for convenience and efficiency. Consumers tend to prefer services that are fast, easy, and efficient. One of the tangible proofs of technological development is *online* transportation. Maxim is a Russian app that offers a wide range of services, from online transportation, food and goods delivery, to cleaning services and more. Maxim's popularity in Indonesia is reflected in data on the Google Play Store, which shows more than 50 million downloads and 5 million reviews from users. The reviews on the Google Play Store platform reflect users' experiences of the quality of the Maxim app they have used, whether positive, neutral or negative. Sentiment analysis, also known as *opinion mining*, is the process of extracting and analyzing opinions, sentiments, and emotions expressed in text[1] . To know and understand user reviews, sentiment analysis becomes a crucial tool to understand public perception of the Maxim app. A number of previous studies have conducted sentiment analysis using traditional methods. Research analyzing 2,000 Grab Indonesia app user reviews showed that the Bi-Directional LSTM model equipped with a *Multi-Head Attention* mechanism was able to provide superior sentiment classification results compared to Stacked LSTM. This model obtained a validation accuracy of 87%, with *F1-score* of 0.90 for negative sentiment and 0.82 for positive sentiment respectively[2]. Research on the Maxim application has also been carried out using the *Naive Bayes Classifier* method, using 1,000 review data, the model produces 84% accuracy, 83% *precision*, 93% *recall*, and 88% *F1-score*[3]. Previous research with the SVM method has also been carried out for the classification of post-presidential election public sentiment with data sourced from X as much as 3,850 data. The test results through grouping popular *tweets* into three categories, the highest results were obtained in the "Peaceful Election" class with 97.3% accuracy, followed by "Inquiry Rights" recording a score of 96.5%, while "Rigged Election" achieved 94.0%. However, there is an opportunity to improve accuracy by applying more sophisticated *deep learning* models[4].

Although traditional methods such as SVM, Naive Bayes, and LSTM are promising, recent developments show that deep learning models like BERT can provide superior results in understanding complex textual context. Several studies have demonstrated the advantages of BERT in sentiment analysis tasks. Such research conducted on the JOOX application by taking 10,000 reviews from the Google Play Store was able to produce an increase in accuracy of 3.6% for positive sentiment and 1.3% for negative sentiment compared to *baseline* accuracy. This research highlights the advantages of BERT in understanding the two-way context in the review text, so that it is able to produce an *F1-score* of 86% positive sentiment, 51% for neutral sentiment, and 76% for negative sentiment[5]. In addition, research was also conducted on the Ruang Guru application using 5,437 review data by dividing training data and test data with a ratio of 70:30 able to provide excellent performance by obtaining *F1-Score* of 98.9%, accuracy reached 99%, precision of 64.13%, and recall of 60.51%[6]. A similar study on the Access by KAI application with 9,260 reviews

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

achieved an accuracy of 85%. The BERT model was able to classify negative and positive sentiments effectively, although the performance on neutral sentiment still requires improvement[7]. Furthermore, a study on user reviews of Garuda Indonesia airline on Twitter also compared the BERT model with traditional methods such as MultinomialNB and SVM. As a result, BERT showed the best performance with an accuracy of 75.6%, outperforming the other two methods and demonstrating its ability to understand context more deeply in sentiment classification tasks[8]. Another study that examined sentiment analysis on the Shopee app compared the performance of BERT, LSTM, and CNN. The BERT model achieved the highest accuracy of 83%, outperforming LSTM (78%) and CNN (75%). This confirms that BERT is more effective in understanding complex language patterns in user review texts[9].

The uniqueness of the Maxim app review data that supports the use of the bert base multilingual uncased model lies in the diversity of language and writing styles used by Indonesian users. Many reviews are written informally and often without capitalization rules. Moreover, the large volume of data creates a complex and varied review context. Therefore, using the bert base multilingual uncased model is appropriate, as it is case-insensitive and capable of understanding bidirectional context in linguistically and semantically rich texts.

Based on these studies and conditions, this research aims to adapt and optimize the capabilities of the BERT model, which has proven superior in understanding bidirectional text context. By applying this approach to the sentiment analysis of Maxim user reviews, it is expected that the sentiment predictions will be more accurate and in-depth. Furthermore, this research is expected to make a real contribution to the development of app quality improvement strategies, through a more comprehensive understanding of user opinions. The relevance of this research becomes increasingly important, especially for stakeholders such as Maxim online transportation companies, to evaluate and improve service quality based on real-time user feedback.

II. RESEARCH METHODOLOGY

Sentiment analysis of the Maxim application is carried out through a number of stages that have been prepared through the application of the BERT method. Starting from data collection, data labeling, tokenization, data sharing, BERT model implementation, and ending with evaluation to find the results of model accuracy.

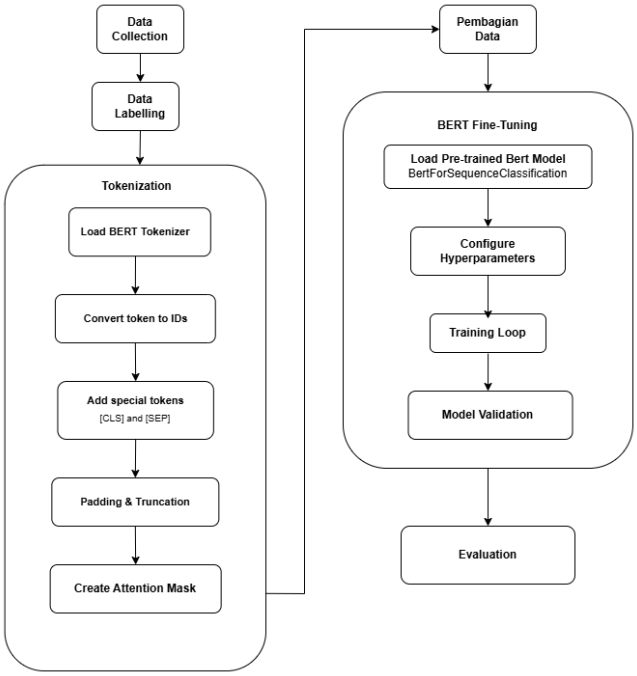


Fig 1. Research Flow Chart

A. . *Data Collection*

The data source in this study comes from reviews of Maxim application users available on the Google Play Store. This data was taken using the *google-play-scraper library* totaling 10,000 data. The data collected includes review texts and scores of 1 to 5. The reviews, written in Indonesian, were sorted by recency. The *scrapped* dataset was then saved in CSV format.

B. *Data Labeling*

The collected data in CSV format was labeled by categorizing sentiment based on user review scores. The labeling process was conducted using a rule-based function that converts numerical scores into sentiment labels. Reviews with scores from 1 to 2 were labeled as negative with a value of 0, reviews with a score of 3 were labeled as neutral with a value of 1, and reviews with scores from 4 to 5 were labeled as positive with a value of 2. This method automatically categorizes each review by applying the rules to the score column in the dataset. The process was implemented in code using a function that checks the score value and returns the corresponding sentiment label. After labeling, the data consisted of approximately 4,000 positive reviews, 2,000 neutral reviews, and 4,000 negative reviews.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

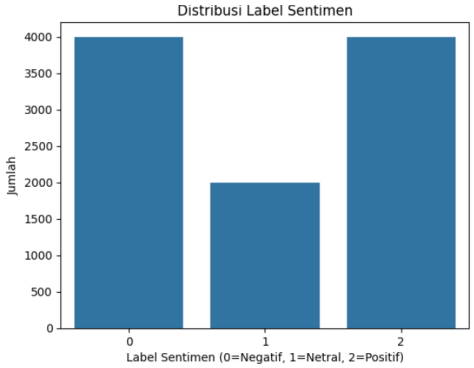


Fig 2. Sentiment Label Distribution

C. Tokenization

In text-based sentiment analysis, especially using models like BERT, a very important first step is tokenization. One of the advantages of *transfer learning-based* models such as BERT is its ability to understand the full context of a sentence. BERT is more effective when working with relatively intact text and directly tokenizes the input[10]. Moreover, in terms of handling common words such as *stopwords*, BERT takes a different approach compared to traditional methods. During the dataset tokenization stage, a series of transformation steps are performed on the text data to match the input format required by the BERT model. This process aims to convert user reviews from the Maxim application into numerical representations that the model can understand. The first step begins with the initialization of the tokenizer, specifically the BertTokenizer from the bert-base-multilingual-uncased model. Before tokenization is performed, the text does not undergo complex manual preprocessing such as punctuation removal, letter case normalization, or stopword elimination. This is because the BERT tokenizer inherently handles these tasks automatically. The bert-base-multilingual-uncased model internally performs automatic lowercasing, meaning uppercase and lowercase letters are treated equally without the need for additional processing. Additionally, subword tokenization allows the model to recognize uncommon words, compound words, or even minor misspellings without requiring explicit punctuation removal. Furthermore, in terms of handling common words like *stopwords*, BERT employs a distinct strategy from traditional approaches. Stopwords are not removed because BERT considers the order and position of words in a sentence through its *self-attention* mechanism. Therefore, stopwords are retained to preserve the contextual meaning of the sentence as a whole. Once the tokenizer is loaded, tokenization is carried out for each sentence or review. This process breaks the text into token units using the *subword tokenization* technique. These tokens are then converted into numerical form using the convert_tokens_to_ids function. Next, each sentence is prepended with the special token [CLS] and appended with [SEP] using the parameter add_special_tokens=True. The [CLS] token serves as a representation of the entire sentence used in classification, while the [SEP] token marks the separator between sentences or sentence pairs in the input as a sentence end marker[11]. Since input lengths vary, padding or truncating is applied so that all sequences reach a uniform maximum length of 200 tokens. Padding is done by appending [PAD] tokens (ID = 0) to the end of shorter sequences.

An attention mask is then created to indicate relevant tokens (non-padding). The attention mask is a binary vector that marks valid tokens with a value of 1 and padding tokens with a value of 0. This is important because BERT uses the *self-attention* mechanism, which takes into account every input position. With the attention mask, the model can ignore padding tokens that carry no information. This practice is a standard approach in *transformer-based* natural language processing and is essential to ensure the efficiency and robustness of the model's attention[12].

D. Data Splitting

In machine learning practice, data *splitting* is often used to divide the dataset into three parts: training, validation, and testing data. This strategy plays an important role in the *hyperparameter* adjustment process of the model as well as to predict the model's ability to handle new data outside the training data[13]. The data was divided into training and testing with a ratio of 90% for training and 10% for testing. Then from 90% training data, 15% is separated as validation data. The data division process is done by utilizing the *train_test_split* function from the *sklearn.model_selection* library. It is intended that the model can perform a generalization process on data that has never been seen before.

E. BERT Model Implementation

In the model implementation stage, the text tokens are first converted into a tensor format using the *PyTorch* library. After that, the data is shared into the *DataLoader* for training and validation purposes. This research applies the *pre-trained* BERT model with the *bert-base-multilingual-uncased* type provided by the *HuggingFace Transformers* library. This model was chosen because it has been trained using the wikipedia corpus in 104 languages, including Indonesian[14]. The *pre-trained* model is then implemented through the *BertForSequenceClassification* class, a variant of BERT specifically tailored for text classification tasks.

In this research, the number of labels used is three classes, according to the classification needs of the analyzed data. Architecturally, the *BertForSequenceClassification* model consists of three main components, namely the *embedding layer*, *encoder layer*, and *output layer*. *The embedding layer* is responsible for transforming each input token into a fixed-dimensional vector representation. Then, the vectors are processed by the *encoder* layer which consists of 12 multilevel layers with a *self-attention* mechanism, which allows the model to understand the context and interrelationships between words in one sentence and between sentences. After that, the *output of* the *encoder* is processed by the *output layer* which includes a *pooling* and *linear* classification layer that produces a final prediction of three *output* classes.

In order to optimize the *fine-tuning* process, *hyperparameter* settings such as *learning rate*, *batch size*, *and* number of *epochs* are important aspects that are considered[15]. In this study, the learning rate was set to 2e-5. This value provides a balance between training stability and adaptability, while also avoiding early overfitting [16]. The batch size was set to 64, as it offers a good balance between memory efficiency

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

and gradient stability during training. Initial experiments showed that this batch size allows for consistent training without disrupting memory allocation.

For the number of epochs, a systematic evaluation approach was implemented to determine the optimal training duration. The model was trained for 3, 4, and 5 epochs, with comprehensive performance analysis conducted for each configuration. To ensure an objective selection process, multiple evaluation metrics were analyzed including accuracy, precision, recall, F1-score, and loss convergence patterns. Table I presents the detailed performance comparison across different epoch configurations.

TABLE I . PERFORMANCE COMPARISON ACROSS DIFFERENT EPOCHS

Metric	3 Epochs	4 Epochs	5 Epochs
Accuracy	0.9400	0.9470	0.9400
Precision (Macro)	0.9337	0.9414	0.9385
Recall (Macro)	0.9407	0.9457	0.9313
F1-Score (Macro)	0.9360	0.9426	0.9347
F1-Score (Weighted)	0.9398	0.9469	0.9399

TABLE II . TRAINING AND VALIDATION LOSS ANALYSIS

Epoch Configuration	Training Loss	Validation Loss	Loss Gap	Overfitting Assessment
3 Epochs	0.156	0.178	0.022	Minimal risk
4 Epochs	0.122	0.138	0.016	Optimal balance
5 Epochs	0.108	0.142	0.034	Early overfitting

Based on this comprehensive analysis, the 4th epoch produced the best performance across all evaluation metrics. The selection of 4 epochs as the optimal configuration was justified by several key findings. First, the model achieved the highest accuracy of 94.7% at 4 epochs, demonstrating superior classification performance compared to other configurations. Second, the loss analysis revealed that 4 epochs provided the optimal balance between training and validation loss with the smallest gap of 0.016, indicating excellent generalization capability without overfitting. Third, all performance metrics including precision, recall, and F1-scores reached their peak values at 4 epochs, showing consistent improvement across all evaluation dimensions.

Training for 3 epochs showed signs of underfitting, as evidenced by suboptimal performance metrics and a relatively high loss gap, suggesting that the model had not fully learned the underlying patterns in the data. Conversely, at the 5th epoch, while the training loss continued to decrease, the validation loss began to plateau and slightly increase, coupled with a widening loss gap to 0.034, which indicated the onset of overfitting. This pattern suggested that the model was beginning to memorize training data rather than learning generalizable patterns. Therefore, training for 4 epochs was considered the most optimal setting in this study because it provides the best generalization capability, training stability, and overall performance balance. In addition, the AdamW *optimizer* is used which is known to be effective for transformer-based models because it is able to handle *weight decay* more stably.

F. Evaluation

The next step is to evaluate the performance of the model against the test data. The evaluation aims to assess the performance of the model in predicting data that has never been trained before. In this research, the metrics used to evaluate the model include accuracy, *precision*, *recall*, and *F1-score*, which are common metrics in classification tasks. The evaluation is done by utilizing the *classification_report* function from the *scikit-learn* library, which automatically calculates the values of the four metrics based on the predicted and actual labels. In addition, a *confusion matrix* is also used to see the distribution of prediction errors made by the model.

III. RESULTS AND DISCUSSION

A. Dataset

The data that forms the basis of this research is obtained from user reviews of the Maxim application taken using the *google-play-scraper python library*. The dataset consists of 10,000 reviews that include two columns, namely, *score* and *content* columns. *Score* is a rating given by users while *content* contains reviews or comments written by users. After that, the sentiment labeling process is carried out based on the *score* which produces a new column, namely the label column. Sentiments are categorized with labels 0 for negative, 1 for neutral, and 2 for positive. Table III shows an example of a labeled data set.

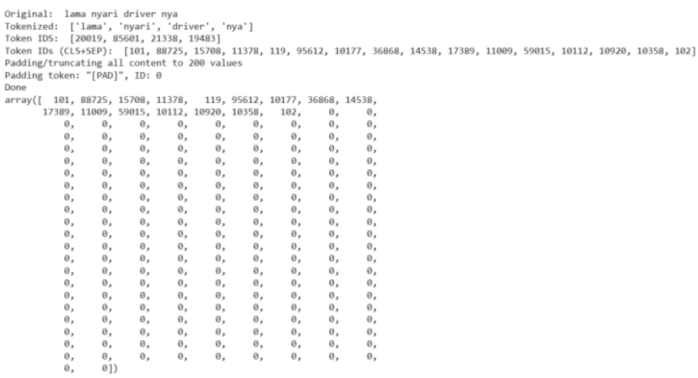
TABLE III . SAMPLE DATA SET

Score	Content	Label
1	the application is slow, the driver is cool even though he is close to the point, he doesn't accept orders, it's already rich, isn't it?? he said he was looking for a driver who was even far from the point.	0
2	drivers always raise the price	0
3	Driver does not provide change	1
4	good service	2
5	driver is very satisfactory	2

B. Tokenization

Tokenization is carried out after the user review data has been successfully loaded into the *dataframe*. This process serves to break down sentences into token units that can be processed by the BERT model.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025



```
Original:  lama nyari driver nya
Tokenized:  ['lama', 'nyari', 'driver', 'nya']
Token IDS:  [20019, 85601, 21338, 19483]
Token IDs (CLS+SEP):  [101, 88725, 15708, 11378, 119, 95612, 10177, 36868, 14538, 17389, 11009, 59015, 10112, 10920, 10358, 102]
Padding/truncating all content to 200 values
Padding token: "[PAD]", ID: 0
Done
array([  101, 88725, 15708, 11378,   119, 95612, 10177, 36868, 14538,
       17389, 11009, 59015, 10112, 10920, 10358,   102,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0])
```

Fig 3. Tokenized Sentences

Figure 3 is an example of the results of tokenization. First the original sentence is broken down into tokens. The tokens are then converted into token Ids (numeric numbers) that represent each token based on the vocabulary owned by BERT. The BERT model also adds special tokens at the beginning and end of the sequence, namely [CLS] and [SEP]. For model training purposes, all ID tokens are *padded* or *truncated* to a uniform length of 200 tokens. An additional token [PAD] with ID 0 is used to fill the remaining length if the number of tokens is less than 200. In the final result after *padding*, the *array* contains 200 values, starting from [CLS] followed by the token ID of the tokenization result, then ending with [SEP], and the rest is filled with [PAD] tokens (ID 0) until it reaches a length of 200 elements.

C. Data Splitting

From a total of 10,000 data samples, the dataset is divided into training, validation, and testing sets. Initially, 90% of the data (9,000 samples) is allocated for training and 10% (1,000 samples) for testing. From the 9,000 training samples, 15% (1,350 samples) is further split for validation purposes. As a result, the final data distribution consists of 7,650 samples for training, 1,350 samples for validation, and 1,000 samples for testing. The data division process is done by utilizing the *train_test_split* function from the *sklearn.model_selection* library.

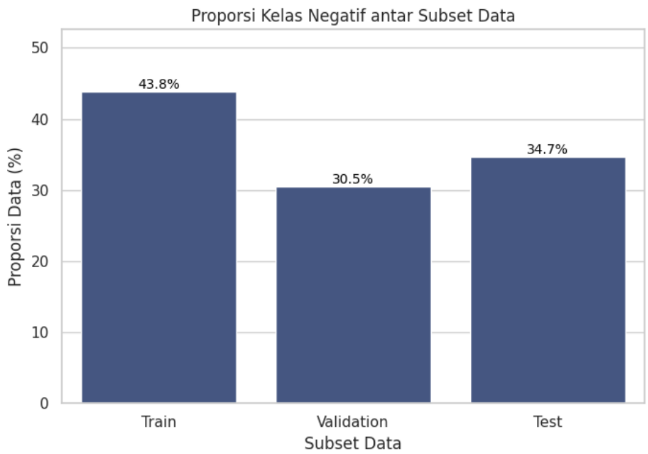


Fig 4. Proportion of the Negative Class

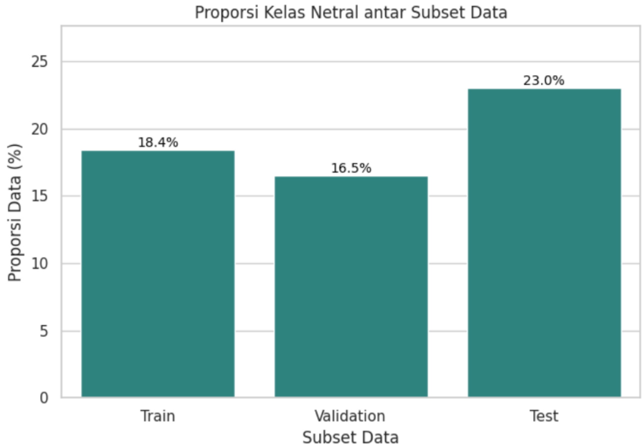


Fig 5. Proportion of the neutral class

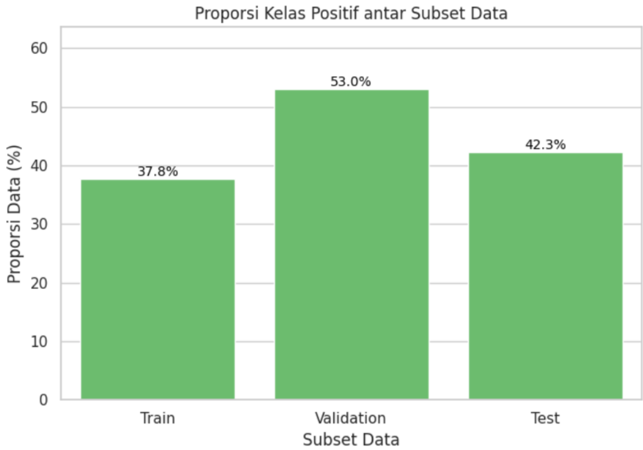


Fig 6. Proportion of the positive class

Train data is obtained in the amount of 7,650 data, *validation* data in the amount of 1,350, and *test* data in the amount of 1,000 data divided into three classes (positive, negative, neutral).

D. Training and Evaluation

This research uses the BERT multilingual model (bert-base-multilingual-uncased) from the HuggingFace Transformers library to perform multi-class text classification. The model is fine-tuned specifically to classify input texts into one of three predefined categories. To facilitate this task, the *BertForSequenceClassification* class is utilized, which is a variant of the BERT architecture modified by adding a classification layer on top. This layer enables the model to learn and predict the appropriate class labels based on the input, making it suitable for classification tasks involving multiple output categories.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

```
The BERT model has 201 different named parameters.
===== Embedding Layer =====
bert.embeddings.word_embeddings.weight                  (105879, 768)
bert.embeddings.position_embeddings.weight                (512, 768)
bert.embeddings.token_type_embeddings.weight                (2, 768)
bert.embeddings.LayerNorm.weight                            (768,)
bert.embeddings.LayerNorm.bias                              (768,)
===== First Transform =====
bert.encoder.layer.0.attention.self.query.weight          (768, 768)
bert.encoder.layer.0.attention.self.query.bias              (768,)
bert.encoder.layer.0.attention.self.key.weight            (768, 768)
bert.encoder.layer.0.attention.self.key.bias                (768,)
bert.encoder.layer.0.attention.self.value.weight          (768, 768)
bert.encoder.layer.0.attention.self.value.bias              (768,)
bert.encoder.layer.0.attention.output.dense.weight        (768, 768)
bert.encoder.layer.0.attention.output.dense.bias            (768,)
bert.encoder.layer.0.attention.output.LayerNorm.weight      (768,)
bert.encoder.layer.0.attention.output.LayerNorm.bias        (768,)
bert.encoder.layer.0.intermediate.dense.weight           (3072, 768)
bert.encoder.layer.0.intermediate.dense.bias               (3072,)
bert.encoder.layer.0.output.dense.weight                 (768, 3072)
bert.encoder.layer.0.output.dense.bias                      (768,)
bert.encoder.layer.0.output.LayerNorm.weight                (768,)
bert.encoder.layer.0.output.LayerNorm.bias                  (768,)
===== Output Layer =====
bert.pooler.dense.weight                                  (768, 768)
bert.pooler.dense.bias                                      (768,)
classifier.weight                                           (3, 768)
classifier.bias                                               (3,)
```

Fig 7. Layer Arrangement of the Model

The model was trained through a fine-tuning process using the AdamW optimizer, which is commonly used for training transformer-based models like BERT. To ensure stable training and reduce the risk of overfitting, several key hyperparameters were configured carefully. The learning rate was set to 2e-5, with an epsilon value of 1e-8 to improve numerical stability during optimization. The training was conducted over 4 epochs, which was found to be sufficient to allow the model to learn meaningful patterns from the data without overfitting.

Evaluation of the model's performance is done through the utilization of a *learning curve* that describes the development of *training loss* and *validation loss* during the training process. Figure 6 shows the learning curve of the model for 4 *epochs*.

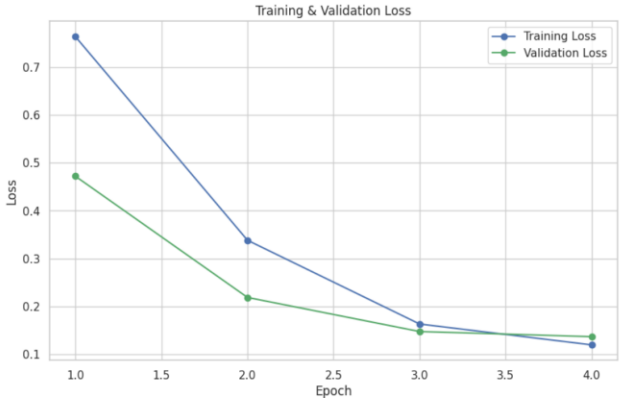


Fig 8. Learning Curve

In the first *epoch*, the *training loss* value was around 0.77, while the *validation loss* was 0.47. As the training progressed, both values decreased significantly. By the 2nd epoch, the *training loss* dropped to around 0.34, and the *validation loss* to 0.22. This downward trend continued until the end of training at the 4th epoch, where the *training loss* reached around 0.12 and the *validation loss* was close to 0.13. The consistent decrease in both metrics and the smaller distance between *training loss* and *validation loss* indicate a stable training process without *overfitting*. This finding indicates that the model's performance in learning patterns on the training data is quite optimal and can generalize to the validation data effectively.

In this study, the model was evaluated using various evaluation metrics, including *confusion matrix*, *accuracy*, *recall*, *precision*, and *F1-score*.

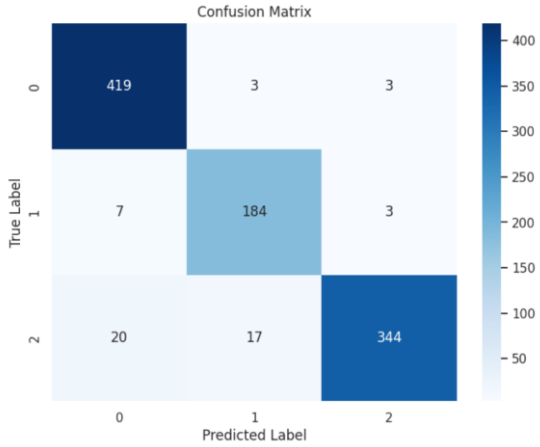


Fig 9. Confusion Matrix

performance of a classification model across three distinct classes: class 0, class 1, and class 2. For class 0 (True Label 0), the model successfully predicted 419 data instances correctly (True Positive). However, 6 data instances of class 0 were misclassified into other classes, with 3 instances predicted as class 1 and another 3 as class 2 (False Negatives). Additionally, there were 27 data instances from other classes incorrectly predicted as class 0 (False Positives), comprising 7 from class 1 and 20 from class 2.

For class 1 (True Label 1), the model accurately classified 184 data instances (True Positive). There were 10 misclassified data instances from class 1 (False Negatives), comprising 7 instances incorrectly predicted as class 0 and 3 instances as class 2. In addition, 20 data instances from other classes were wrongly predicted as class 1 (False Positives), including 3 from class 0 and 17 from class 2.

Finally, for class 2 (True Label 2), 344 data instances were correctly classified (True Positive). Nevertheless, the model misclassified 37 data instances from class 2 (False Negatives), with 20 instances erroneously predicted as class 0 and 17 as class 1. Meanwhile, 6 data instances from class 0 and class 1 were incorrectly predicted as class 2 (False Positives), with 3 from class 0 and 3 from class 1.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

Classification Report:

	precision	recall	f1-score	support
0	0.9395	0.9859	0.9621	425
1	0.9020	0.9485	0.9246	194
2	0.9829	0.9029	0.9412	381
accuracy			0.9470	1000
macro avg	0.9414	0.9457	0.9426	1000
weighted avg	0.9487	0.9470	0.9469	1000

Fig 10. Clasification Report

Figure 10 presents the classification report of the model based on precision, recall, f1-score, and accuracy metrics. Overall, the model achieves an accuracy rate of 94.7%, with high and balanced F1-scores across all classes: 96.2% (Negative), 92.4% (Neutral), and 94.1% (Positive). These results demonstrate strong performance in multi-class sentiment classification.

However, it is important to note that the F1-score for the neutral class is slightly lower compared to the other classes. This can be attributed to two main factors. First, there is data imbalance in the class distribution within the training dataset. The proportion of neutral data is relatively smaller compared to positive and negative data, resulting in fewer examples for the model to learn patterns for this class. This situation causes the model's ability to classify neutral sentiment to be somewhat less optimal.

Second, from a linguistic context, neutral sentiment tends to have higher ambiguity than positive or negative sentiments. Reviews with neutral sentiment often contain words that do not explicitly express satisfaction or dissatisfaction, leading to varied interpretations. This poses a particular challenge for the model in distinguishing neutral sentiment from the other two classes, especially when strong sentiment indicators are absent.

Nevertheless, the F1-score of 92.4% for the neutral class is still considered high, indicating that the model can generalize well across all three classes. Further handling of data imbalance, such as oversampling, undersampling, or applying class weights, could be considered in future research to improve neutral sentiment classification performance.

Furthermore, compared to previous studies analyzing user reviews of the Maxim application using the Naïve Bayes algorithm, the superiority of this model is evident. The prior study only performed binary sentiment classification (positive and negative) on a dataset of 1,000 entries and achieved an accuracy of 84%, with F1-scores of 88% for the negative class and 75% for the positive class. In contrast, the approach used in this research not only accommodates an additional class (neutral) but also uses a dataset ten times larger, comprising 10,000 reviews. The BERT-based model applied here achieves a 10.7% higher accuracy, as well as more balanced performance across all sentiment categories. This demonstrates that the fine-tuned BERT model in this study is more effective and reliable in handling large-scale multi-class sentiment analysis tasks compared to conventional machine learning methods.

IV. CONCLUSION

This study was conducted to understand user perceptions of the Maxim application through a sentiment analysis approach based on the multilingual uncased BERT base model. The dataset consisted of 10,000 user reviews, with 90% (9,000 reviews) allocated for training and 10% (1,000 reviews) for testing. From the training data, 15% (1,350 reviews) was set aside for validation, resulting in an effective training dataset of 7,650 reviews. The model demonstrated excellent performance in sentiment classification of Maxim app reviews, achieving an accuracy of 94.7%. The highest F1-score was obtained for positive sentiment at 0.9621, followed by neutral at 0.9412, and negative at 0.9246. These results indicate that the model can accurately recognize sentiments, particularly in distinguishing positive, neutral, and negative opinions from user reviews.

However, this study has several limitations. The dataset was solely sourced from the Google Play Store, excluding other sources such as the App Store or social media, which could provide a broader perspective on user perceptions. Furthermore, sentiment labeling was done automatically based on user rating scores, which may introduce bias since a score of 3 does not always represent a neutral sentiment, as user interpretations can vary depending on context. Therefore, future research is recommended to expand data sources and consider manual or combined manual and automatic labeling approaches to reduce potential bias in labeling.

REFERENCES

[1] F. Greco, *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers, 2022. doi: 10.4337/9781800374263.sentiment.analysis.

[2] A. R. Gunawan and R. F. Alfa Aziza, "Sentiment Analysis Using LSTM Algorithm Regarding Grab Application Services in Indonesia," *J. Appl. Informatics Comput.*, vol. 9, no. 2, pp. 322–332, Mar. 2025, doi: 10.30871/jaic.v9i2.8696.

[3] A. N. Hasanah and B. N. Sari, "ANALISIS SENTIMEN ULASAN PENGGUNA APLIKASI JASA OJEK ONLINE MAXIM PADA GOOGLE PLAY DENGAN METODE NAÏVE BAYES CLASSIFIER," *J. Inform. dan Tek. Elektro Terap.*, vol. 12, no. 1, Jan. 2024, doi: 10.23960/jitet.v12i1.3628.

[4] K. Adib, M. R. Handayani, W. D. Yuniarti, and K. Umam, "Opini Publik Pasca-Pemilihan Presiden: Eksplorasi Analisis Sentimen Media Sosial X Menggunakan SVM," *SINTECH (Science Inf. Technol. J.*, vol. 7, no. 2, pp. 80–91, Aug. 2024, doi: 10.31598/sintechjournal.v7i2.1581.

[5] J. U. S. Lazuardi and A. Juarna, "ANALISIS SENTIMEN ULASAN PENGGUNA APLIKASI JOOX PADA ANDROID MENGGUNAKAN METODE BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER (BERT)," *J. Ilm. Inform. Komput.*, vol. 28, no. 3, pp. 251–260, Dec. 2023, doi: 10.35760/ik.2023.v28i3.10090.

[6] W. Y. Raden Mas Rizqi Wahyu Panca Kusuma Atmaja, "Analisis Sentimen Customer Review Aplikasi Ruang Guru dengan Metode BERT (Bidirectional Encoder Representations from Transformers)," *J. Emerg. Inf. Syst. Bus. Intell.*, vol. 02, pp. 55–62, 2021.

[7] T. B. B. Wicaksono and R. D. Syah, "IMPLEMENTASI METODE BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS UNTUK ANALISIS SENTIMEN TERHADAP ULASAN APLIKASI ACCESS," *J. Ilm. Inform. Komput.*, vol. 29, no. 3, pp. 254–265, Dec. 2024, doi: 10.35760/ik.2024.v29i3.12514.

[8] B. Prasetyo, Ahmad Yusuf Al-Majid, and Suharjito, "A Comparative Analysis of MultinomialNB, SVM, and BERT on Garuda Indonesia Twitter Sentiment," *PIKSEL Penelit. Ilmu Komput. Sist. Embed. Log.*, vol. 12, no. 2, pp. 445–454, Sep. 2024, doi: 10.33558/piksel.v12i2.9966.

[9] Y. Wu, Z. Jin, C. Shi, P. Liang, and T. Zhan, "Research on the application of deep learning-based BERT model in sentiment analysis," *Appl.*

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025

Comput. Eng., vol. 71, no. 1, pp. 14–20, May 2024, doi: 10.54254/2755-2721/71/2024MA.

[10] E. Alzahrani and L. Jololian, "How Different Text-Preprocessing Techniques using the Bert Model Affect the Gender Profiling of Authors," in *Advances in Machine Learning*, Academy and Industry Research Collaboration Center (AIRCC), Sep. 2021, pp. 01–08. doi: 10.5121/csit.2021.111501.

[11] J. D. M.-W. C. K. L. K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT 2019, pages 4171–4186 Minneapolis, Minnesota, June 2 - June 7, 2019. c 2019 Association for Computational Linguistics*, 2019, pp. 4171–4186.

[12] G. Letarte, F. Paradis, P. Giguère, and F. Laviolette, "Importance of Self-Attention for Sentiment Analysis," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 267–275. doi: 10.18653/v1/W18-5429.

[13] D. E. BIRBA, "A Comparative study of data splitting algorithms for machine learning model selection," KTH ROYAL INSTITUTE OF TECHNOLOGY, 2020.

[14] T. Pires, E. Schlinger, and D. Garrette, "How Multilingual is Multilingual BERT?," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 4996–5001. doi: 10.18653/v1/P19-1493.

[15] X. Liu and C. Wang, "An Empirical Study on Hyperparameter Optimization for Fine-Tuning Pre-trained Language Models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2021, pp. 2286–2300. doi: 10.18653/v1/2021.acl-long.178.

[16] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?," May 2019, [Online]. Available: http://arxiv.org/abs/1905.05583

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI: 10.32736/sisfokom.v14i3.2391, Copyright ©2025
Submitted: Mey 21, 2025, Revised: June 4, 2025, Accepted: June 11, 2025, Published: July 28, 2025