Jurnal Kecerdasan Buatan dan Teknologi Informasi
Vol. 5, No. 1, January 2026, pp. 11-17
e-ISSN: 2964-2922, p-ISSN: 2963-6191
DOI : https://doi.org/10.69916/jkbti.v5i1.390

A RECURRENT NEURAL NETWORK–BASED SENTIMENT ANALYSIS OF MOBILE LEGENDS APP REVIEWS

Naufal Ilmi Rangkuti*1, Imran Lubis2

1,2Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan, Medan, Indonesia
Email: 1naufal.ilmi.rangkuti@gmail.com, 2imran.loebis.medan@gmail.com

(Received: December 8, 2025; Revised: January 13, 2026; Accepted: January 16, 2026)

Abstract

With the rapid growth of mobile applications, user reviews have become a valuable source of feedback for developers. This study investigates the use of a Recurrent Neural Network (RNN) for sentiment analysis of Mobile Legends user reviews. The textual data were preprocessed through cleaning, tokenization, and padding, while sentiment scores were converted into categorical labels. A Sequential RNN model, consisting of an Embedding layer, a SimpleRNN layer, and a Dense output layer with softmax activation, was constructed to classify reviews into three sentiment categories: negative, neutral, and positive. During training, the model achieved approximately 75% accuracy, and the Mean Squared Error (MSE) was 0.1354. However, evaluation using the classification report and confusion matrix revealed that the model was biased toward the negative class due to class imbalance, failing to correctly classify neutral and positive reviews. The high overall accuracy was misleading, as the model's performance was limited by the dominance of the negative class. These results highlight the limitations of using a simple RNN architecture for multi-class sentiment classification in imbalanced datasets. To improve performance, future work should consider balancing the dataset through resampling or synthetic data generation and employing more advanced sequential models, such as LSTM or GRU, possibly combined with attention mechanisms or pretrained language models, to better capture the characteristics of all sentiment classes.

Keywords: sentiment analysis; mobile legends; RNN; deep learning.

1. INTRODUCTION

In the current digital era, the growth of mobile-based applications has increased exponentially along with the widespread adoption of smart devices [1]–[4]. Application distribution platforms such as the Google Play Store and the App Store provide user review features that function as interactive media between developers and users. These reviews not only reflect user satisfaction with an application but also serve as a highly valuable data source for developers in conducting evaluations and continuous improvements. In the context of online gaming applications, such as Mobile Legends: Bang Bang, the number of reviews submitted daily is very large and diverse, representing a wide range of user experiences and expectations toward the application [5].

User comments in these reviews can generally be categorized into two main polarities, namely positive comments and negative comments [6]–[9]. Positive comments usually express appreciation for application quality, such as enjoyable gameplay, satisfactory graphics, or fair matchmaking systems. Conversely, negative comments often contain criticism related to bugs, unstable connections, or imbalance within the game. A comprehensive understanding of these types of comments is crucial in assessing application service quality and supporting decision-making in subsequent development processes [10].

However, manual analysis of user comments has significant limitations, particularly due to the massive volume of data and the unstructured nature of textual information. In addition, the focus of user comments varies widely, ranging from technical performance issues (such as lag and server stability), game features (such as skin systems and hero updates), to satisfaction with developer policies. This heterogeneity poses challenges in opinion classification, thus requiring a systematic approach capable of extracting meaning from text with high accuracy. Previous research conducted by [8] employed a Recurrent Neural Network with Long Short-Term Memory for sentiment analysis of Instagram comments related to STMIK AKAKOM Yogyakarta. The testing accuracy obtained was 65%, while the implementation accuracy reached 79.46%. Subsequently, another study by [4] analyzed sentiment toward BPJS Kesehatan using the Recurrent Neural Network method, showing that 55.1% of opinions were negative and 44.9% were positive. The best model was achieved using a data partition of 90% for training and 10% for testing, resulting in an accuracy of 86.67%.

As a solution, a deep learning-based approach, particularly the Recurrent Neural Network (RNN), can be applied to automatically perform sentiment analysis on user reviews. RNNs have the capability to understand context in sequential text data, making them more effective in capturing emotional nuances in comments compared to classical methods [6], [11]–[13] . By applying RNNs to analyze reviews of the Mobile Legends application, it is expected that a more structured and informative overview of user perceptions regarding application performance and features can be obtained, ultimately supporting strategic decision-making by developers [14].

2. RESEARCH METHODS

2.1. Sentiment Analysis

Sentiment analysis is a branch of Natural Language Processing (NLP) that aims to identify and classify opinions or emotions contained in a text. This approach enables computer systems to understand users' subjective orientations toward a particular topic, product, service, or other entities. In an increasingly digitalized world, sentiment analysis has become important because it is capable of revealing public perceptions automatically and on a large scale.
Technically, sentiment analysis focuses on detecting the polarity of a text, namely whether a statement is positive, negative, or neutral. This classification can be performed at various levels of granularity, ranging from the word level, phrase level, sentence level, to the entire document. In the context of applications, such as user reviews on the Google Play Store, sentiment analysis is able to provide an overall picture of user satisfaction or dissatisfaction with the features available in an application [4].

2.2. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to process sequential data or data sequences, such as text, speech, or time-series data [8]. RNNs have the ability to retain information from previous time steps through memory or hidden states, making them suitable for problems involving sequences or temporal dependencies. In an RNN, the input at a given time step is influenced by previous inputs through a neural network structure that contains loops. This mechanism allows RNNs to recognize patterns in sequential data and make predictions based on those patterns [4]. The basic formulation of an RNN is given as follows:

$$h_t = \sigma(W_h h h_{t-1} + W_x h x_t + b_h) \quad (1)$$

where :

h_t	:	the hidden state at time t
h_{t-1}	:	the hidden state at time t−1, which carries information from the previous time step
x_t	:	the input at time t
$W_h h$	:	the weight matrix connecting the previous hidden state to the current hidden state
$W_x h$	:	the weight matrix connecting the current input to the current hidden state
b_h	:	the bias of the hidden state
σ	:	the activation function, typically sigmoid or tanh

After obtaining the hidden state h_t, the output at time tcan be generated using the following equation:

$$y_t = W_h y h_t + b_y \quad (2)$$

Where :

y_t	:	the output at time t
$W_h y$	:	the weight matrix connecting the hidden state to the output
b_y	:	the bias of the output

3. RESULTS AND DISCUSSION

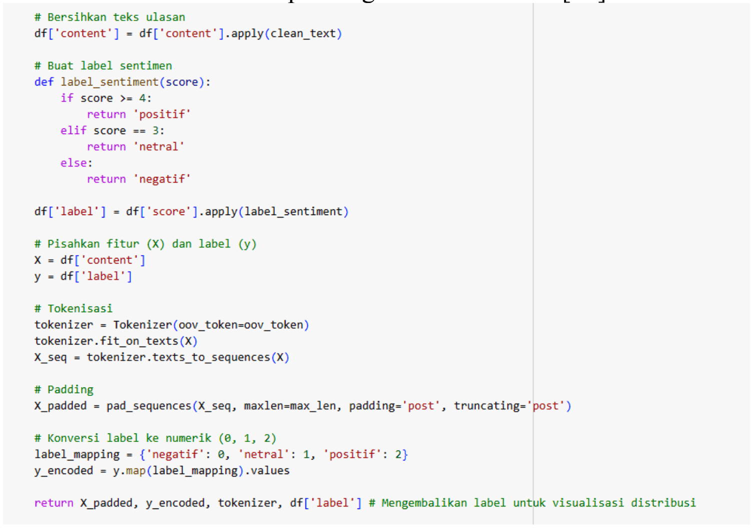
This section explains the stages involved in building and implementing the Recurrent Neural Network (RNN) algorithm for sentiment classification based on review data from the Mobile Legends application. This algorithm was chosen due to its capability to process textual data sequentially, enabling it to understand the contextual relationships of words within a sentence. The stages carried out in this study are as follows:

3.1. Data Uploading and Reading

The training and testing datasets were loaded from CSV files using the load_data function, which takes train_file_path and test_file_path as input parameters corresponding to the respective file locations. The function reads both datasets using the pandas library. When the data loading process is completed successfully, the training and testing datasets are obtained in the form of DataFrames. To ensure data integrity and robustness, the function incorporates error-handling mechanisms. In cases where the specified CSV files cannot be located, an appropriate error notification is generated. Furthermore, any unexpected errors encountered during the data reading process are captured and reported along with relevant error details. If such errors occur, the loading process is terminated and no dataset is returned.

3.2. Data Preprocessing

In the data preprocessing stage, textual data were first cleaned using the clean_text function. This function is responsible for removing unwanted characters, such as punctuation marks and other symbols. The text was also converted to lowercase to maintain consistency in the analysis, and excessive whitespace was eliminated to ensure text cleanliness. Subsequently, the preprocess_data function was applied to perform further preprocessing steps. This function not only calls clean_text to clean the text, but also performs tokenization, converting the text into sequences of numerical values based on the words present in the text. In addition, padding was applied to ensure that all text sequences have a uniform length according to the predefined parameters. The function also transforms sentiment scores (e.g., 1–2 for negative, 3 for neutral, and 4–5 for positive) into sentiment labels suitable for use in the model. Thus, the output of the preprocess_data function consists of cleaned text that has been converted into numerical sequences and annotated with the corresponding sentiment labels [15].

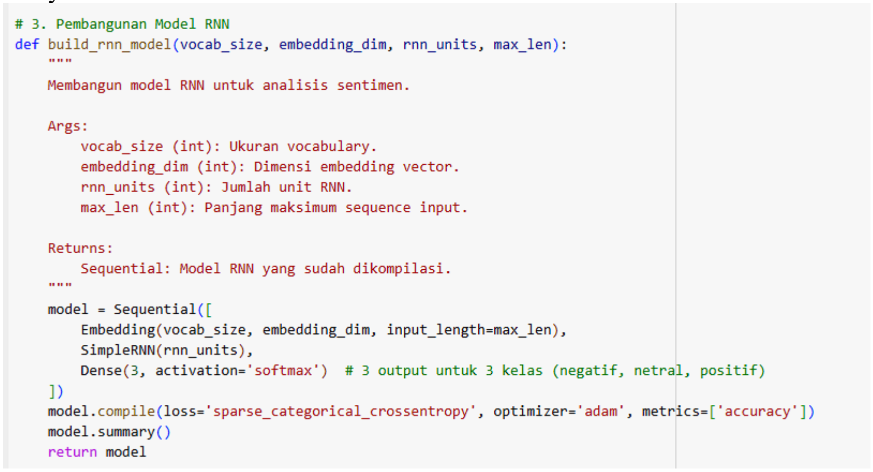


```python
# Bersihkan teks ulasan
df['content'] = df['content'].apply(clean_text)

# Buat label sentimen
def label_sentiment(score):
    if score >= 4:
        return 'positif'
    elif score == 3:
        return 'netral'
    else:
        return 'negatif'

df['label'] = df['score'].apply(label_sentiment)

# Pisahkan fitur (X) dan label (y)
X = df['content']
y = df['label']

# Tokenisasi
tokenizer = Tokenizer(oov_token=oov_token)
tokenizer.fit_on_texts(X)
X_seq = tokenizer.texts_to_sequences(X)

# Padding
X_padded = pad_sequences(X_seq, maxlen=max_len, padding='post', truncating='post')

# Konversi label ke numerik (0, 1, 2)
label_mapping = {'negatif': 0, 'netral': 1, 'positif': 2}
y_encoded = y.map(label_mapping).values

return X_padded, y_encoded, tokenizer, df['label'] # Mengembalikan label untuk visualisasi distribusi
```

Figure 1. Data Preprocessing

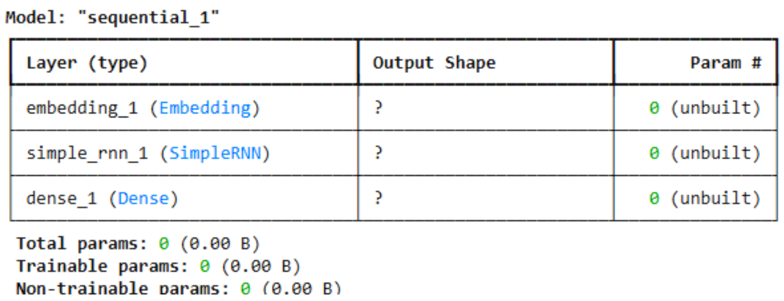
3.3. RNN Model Construction

In this stage, an RNN model was constructed using the Sequential architecture from Keras. The model consists of three main layers. First, an Embedding layer was employed to transform word tokens into fixed-dimensional vectors, facilitating the model's understanding of word context. Second, a SimpleRNN layer was used to process sequential data and capture temporal relationships between words. Finally, a Dense layer with three neurons and a softmax activation function was implemented to classify the output into three sentiment categories: negative, neutral, and positive. The model was compiled using the sparse_categorical_crossentropy loss function, the Adam optimizer, and accuracy as the evaluation metric.



```python
# 3. Pembangunan Model RNN
def build_rnn_model(vocab_size, embedding_dim, rnn_units, max_len):
    """
    Membangun model RNN untuk analisis sentimen.

    Args:
        vocab_size (int): Ukuran vocabulary.
        embedding_dim (int): Dimensi embedding vector.
        rnn_units (int): Jumlah unit RNN.
        max_len (int): Panjang maksimum sequence input.

    Returns:
        Sequential: Model RNN yang sudah dikompilasi.
    """
    model = Sequential([
        Embedding(vocab_size, embedding_dim, input_length=max_len),
        SimpleRNN(rnn_units),
        Dense(3, activation='softmax')  # 3 output untuk 3 kelas (negatif, netral, positif)
    ])
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.summary()
    return model
```

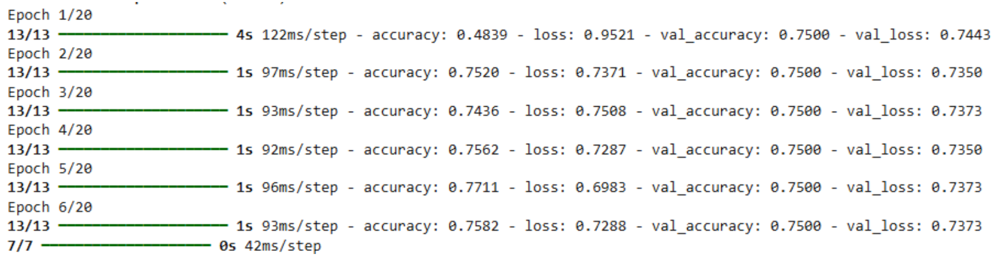
Figure 2. Construction of the RNN Model



```
Model: "sequential_1"
┌──────────────────────────────┬──────────────────────┬──────────────┐
│ Layer (type)                 │ Output Shape         │      Param # │
├──────────────────────────────┼──────────────────────┼──────────────┤
│ embedding_1 (Embedding)      │ ?                    │  0 (unbuilt) │
├──────────────────────────────┼──────────────────────┼──────────────┤
│ simple_rnn_1 (SimpleRNN)     │ ?                    │  0 (unbuilt) │
├──────────────────────────────┼──────────────────────┼──────────────┤
│ dense_1 (Dense)              │ ?                    │  0 (unbuilt) │
└──────────────────────────────┴──────────────────────┴──────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
```

Figure 3. Output of the RNN Model Construction

Figure 3 shows the result of calling the model.summary() function, which is used to display the architecture of the constructed RNN model. The figure illustrates that the model is organized sequentially (Sequential) and consists of three main layers: Embedding, SimpleRNN, and Dense. However, all layers are still in an unbuilt state because the model has not yet received input data, so the Output Shape and the number of parameters (Param #) are not defined. This is a common occurrence when a model is only defined without undergoing training or without being explicitly provided with input. Once the model receives actual input during training (via the fit function), the output shape of each layer and the total number of parameters will be calculated and displayed in full. The Embedding layer transforms words into numerical vectors, the SimpleRNN layer processes the sequential data temporally, and the Dense layer serves as the output layer with a softmax activation function for sentiment classification into three classes: negative, neutral, and positive.



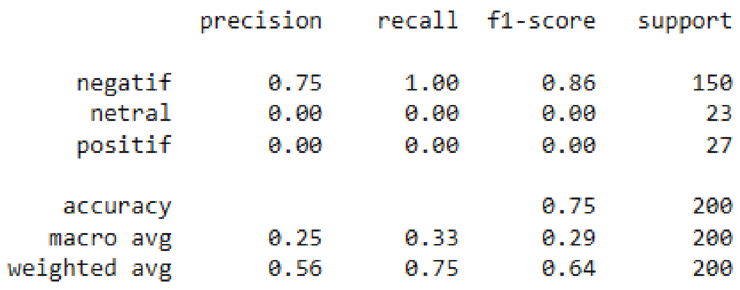
```
Epoch 1/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 122ms/step - accuracy: 0.4839 - loss: 0.9521 - val_accuracy: 0.7500 - val_loss: 0.7443
Epoch 2/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 97ms/step - accuracy: 0.7520 - loss: 0.7371 - val_accuracy: 0.7500 - val_loss: 0.7350
Epoch 3/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.7436 - loss: 0.7508 - val_accuracy: 0.7500 - val_loss: 0.7373
Epoch 4/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 92ms/step - accuracy: 0.7562 - loss: 0.7287 - val_accuracy: 0.7500 - val_loss: 0.7350
Epoch 5/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 96ms/step - accuracy: 0.7711 - loss: 0.6983 - val_accuracy: 0.7500 - val_loss: 0.7373
Epoch 6/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.7582 - loss: 0.7288 - val_accuracy: 0.7500 - val_loss: 0.7373
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
```

Figure 4. Model Training Stages

Figure 4 illustrates the training process of the RNN model from epoch 1 to epoch 6. Each row represents the model's performance during a single epoch, including metrics such as time per step (xx ms/step), training accuracy (accuracy), training loss (loss), validation accuracy (val_accuracy), and validation loss (val_loss). It can be observed that the model's accuracy improved from the first epoch, while the val_accuracy remained stable at 0.7500, indicating that the model began to exhibit overfitting or stagnation, which is typically addressed using techniques such as early stopping. After the sixth epoch, the training process was halted according to the early stopping mechanism, as there was no significant improvement in validation accuracy.

3.4. Testing Using the RNN Algorithm

Testing with the RNN algorithm was conducted to evaluate the model's performance on the test data after the training process had been completed.



```
              precision    recall  f1-score   support

     negatif       0.75      1.00      0.86       150
      netral       0.00      0.00      0.00        23
     positif       0.00      0.00      0.00        27

    accuracy                           0.75       200
   macro avg       0.25      0.33      0.29       200
weighted avg       0.56      0.75      0.64       200
```

Figure 5. Classification Report

Figure 5 presents the evaluation results of the RNN model in the form of a classification report. The report indicates that the model was only able to accurately recognize the negative class, as evidenced by a precision of 0.75, recall of 1.00, and F1-score of 0.86. In contrast, the neutral and positive classes both exhibited precision, recall, and F1-score values of 0.00, indicating that the model failed to classify these classes. Although the overall accuracy appears high (75%), this is primarily due to the dominance of the negative class in the dataset (150 out of 200 test samples), causing the model to be biased toward a single class. The low macro average and weighted average F1-scores further indicate that the model's performance is not balanced across all classes. Therefore, improvements are required in both the dataset and the model architecture to enable more accurate recognition of all classes.

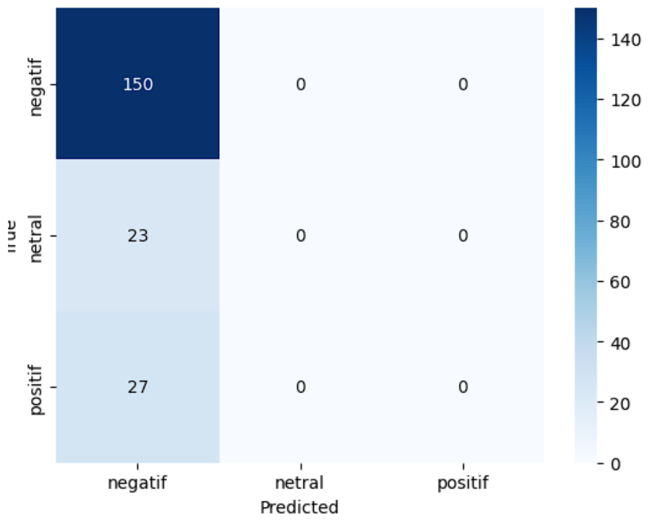


Figure 6. Confusion Matrix

The figure presents the confusion matrix of the RNN model predictions. Based on the visualization, the model classified all samples into the negative class: all 150 negative samples were correctly classified, while all neutral (23) and positive (27) samples were incorrectly classified as negative. This observation reinforces the results from the previous classification report, indicating that the model is biased toward a single class and fails to recognize the other two classes. Such a condition may be caused by class imbalance in the dataset or suboptimal model architecture in capturing the distinctive features of each class.

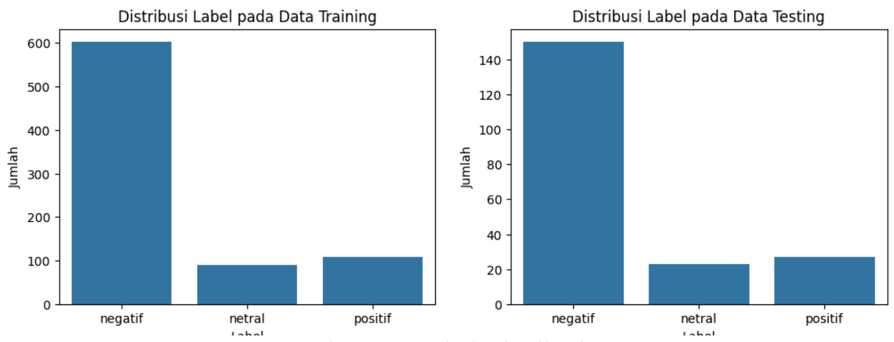


Figure 7. Label Distribution

Figure 7 illustrates the distribution of labels in the training and testing datasets. It can be observed that the negative class dominates the dataset in both training and testing, with a substantially higher number of samples compared to the neutral and positive classes. This imbalance may cause the RNN model to become biased toward the majority class, namely negative, making it difficult to accurately recognize and classify the minority classes. This observation aligns with the previous confusion matrix results, which showed that all predictions were skewed toward the negative class.

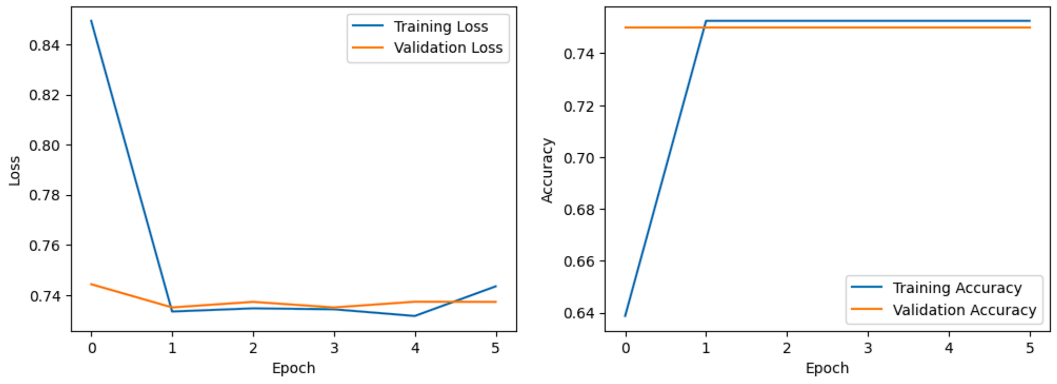
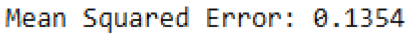


Figure 8. Loss and Accuracy Curves

The figure illustrates the loss and accuracy curves during the RNN model training process. In the left graph, the training loss decreases sharply at the beginning and then stabilizes, while the validation loss remains relatively constant, indicating no significant overfitting. On the right, the accuracy graph shows that both training and validation accuracy rapidly reach a maximum value of approximately 75% and then plateau, suggesting that the model learns quickly but is limited in further performance improvement.

3.5. Mean Squared Error (MSE)

To evaluate the model's performance in terms of prediction errors, the **Mean Squared Error (MSE)** metric was employed.



```
Mean Squared Error: 0.1354
```

Figure 6. Mean Squared Error Value

Figure 6 presents the model evaluation based on the Mean Squared Error (MSE), which was calculated to be 0.1354. MSE is a metric commonly used to assess how well a regression model predicts target values. It is computed as the average of the squared differences between the predicted and actual values. A smaller MSE indicates better model performance, as it demonstrates that the model's predictions are closer to the true values. In this case, an MSE of 0.1354 is considered relatively low, depending on the scale of the data, indicating that the model is reasonably accurate in making predictions, although minor errors remain that should be addressed for further improvement.

3.6. Discussion

The results of this study indicate both the potential and limitations of using a simple RNN model for sentiment classification on Mobile Legends user reviews. During preprocessing, textual data were cleaned, tokenized, and padded to ensure uniform sequence length, while sentiment scores were converted into categorical labels. This step was crucial for preparing the dataset for sequential modeling, allowing the RNN to process contextual information in each review.

The RNN model, constructed using a Sequential architecture with an Embedding layer, a SimpleRNN layer, and a Dense output layer, was trained to classify reviews into three sentiment categories: negative, neutral, and positive. Training progressed smoothly, as shown in the loss and accuracy curves, where training loss decreased sharply and stabilized, while validation loss remained relatively constant, suggesting no significant overfitting. Both training and validation accuracy quickly reached approximately 75% and then plateaued, indicating that the model learned rapidly but was limited in improving performance further.

Evaluation using the classification report and confusion matrix revealed that the model was heavily biased toward the negative class. All negative samples were correctly classified, but the model completely failed to recognize neutral and positive reviews. This outcome is consistent with the observed label distribution, where the negative class dominated both the training and testing datasets. The class imbalance likely caused the model to favor the majority class, demonstrating the limitations of a simple RNN architecture in handling unbalanced multi-class datasets.

The Mean Squared Error (MSE) of 0.1354 further confirmed that, although the model predictions were reasonably close to the actual values in terms of regression-style error, the categorical performance across all classes was suboptimal. The combination of a high accuracy metric, biased predictions, and low macro/weighted F1-scores highlights that accuracy alone is insufficient for evaluating model performance in the presence of class imbalance. Overall, these findings suggest that while RNNs are capable of learning sequential dependencies in textual data, their effectiveness can be limited in datasets with significant class imbalance and when using a relatively simple architecture. To improve performance, several strategies could be considered:

1. Data-level solutions such as oversampling minority classes, undersampling the majority class, or applying synthetic data generation (e.g., SMOTE) to balance the dataset.
2. Algorithm-level improvements, including using more advanced sequential models such as LSTM or GRU, which have a better capability to capture long-term dependencies in text.
3. Hybrid approaches combining RNNs with attention mechanisms or pretrained language models (e.g., BERT) to enhance feature representation and reduce bias toward the majority class.

4. CONCLUSION

This study investigated the application of a simple Recurrent Neural Network (RNN) for sentiment classification of Mobile Legends user reviews. The model was able to process sequential textual data, capturing contextual relationships between words, and achieved a training and validation accuracy of approximately 75%. The Mean Squared Error (MSE) evaluation indicated that the model predictions were reasonably close to the actual values.

However, the model demonstrated a strong bias toward the negative class due to the significant class imbalance in the dataset, failing to correctly classify neutral and positive reviews. This highlights the limitations of using a basic RNN architecture for multi-class sentiment classification on skewed datasets.

Future improvements should focus on addressing data imbalance through resampling techniques or synthetic data generation, and enhancing the model architecture with advanced sequential models such as LSTM or GRU, potentially combined with attention mechanisms or pretrained language models. Such strategies are expected to improve the model's ability to accurately recognize all sentiment classes and provide a more reliable tool for analyzing user feedback.

REFERENCES

[1] A. Subki and B. Imran, "Implementasi Deep Learning Menggunakan CNN dengan Arsitektur Alexnet Untuk Klasifikasi dan Identifikasi Jenis Kopi Khas Lombok Ahmad," *Explore*, vol. 14, no. 2, pp. 135–140, 2024.

[2] A. Ananta Firdaus, A. Id Hadiana, and A. Kania Ningsih, "Klasifikasi Sentimen pada Aplikasi Shopee Menggunakan Fitur Bag of Word dan Algoritma Random Forest," *Ranah Res. J. Multidiscip. Res. Dev.*, vol. 6, no. 5, pp. 1678–1683, 2024, doi: 10.38035/rrj.v6i5.994.

[3] J. Shi, W. Li, Q. Bai, Y. Yang, and J. Jiang, "Syntax-enhanced aspect-based sentiment analysis with multi-layer attention," *Neurocomputing*, vol. 557, no. November 2022, p. 126730, 2023, doi: 10.1016/j.neucom.2023.126730.

[4] F. Faturohman, B. Irawan, S. Si, and C. Setianingsih, "Analisis Sentimen Pada Bpjs Kesehatan Menggunakan Recurrent Neural Network Sentiment Analysis on Bpjs Kesehatan Using Recurrent Neural Network," *e-Proceeding Eng.*, vol. 7, no. 2, pp. 4545–4552, 2020.

[5] R. Subekti *et al.*, *Transformasi Digital: Teori & implementasi Menuju Era Society 5.0*. PT. Sonpedia Publishing Indonesia, 2024.

[6] A. Muhaddisi, "Sentiment Analysis With Sarcasm Detection on Politician's Instagram," *Ijccs (Indonesian J. Comput. Cybern. Syst.*, 2021, doi: 10.22146/ijccs.66375.

[7] L. Nursinggah, T. Mufizar, and U. Perjuangan, "ANALISIS SENTIMEN PENGGUNA APLIKASI X TERHADAP PROGRAM MAKAN SIANG GRATIS," vol. 12, no. 3, 2024.

[8] R. Cahyadi *et al.*, "Recurrent Neural Network (Rnn) Dengan Long Short Term Memory (Lstm) Untuk Analisis Sentimen Data Instagram," *J. Inform. dan Komput.*, vol. 5, no. 1, pp. 1–9, 2020.

[9] W. Irmayani, "PERSEPSI PUBLIK TERHADAP KENAIKAN PPN 12%: PENDEKATAN SENTIMEN PADA KOMENTAR YOUTUBE," *J. KHATULISTIWA Inform.*, vol. 12, no. 2, pp. 112–118, 2024.

[10] D. Tauhida, "Media sosial sebagai arena diskusi keberagamaan: Analisis komentar netizen tentang hijab di Instagram," *Tatar Pas. J. Diklat Keagamaan*, vol. 15, no. 1, 2024.

[11] Y. Yullyana, D. Irmayani, and M. N. S. Hasibuan, "Content-Based Image Retrieval for Songket Motifs using Graph Matching," *Sinkron*, vol. 7, no. 2, pp. 714–719, 2022, doi: 10.33395/sinkron.v7i2.11411.

[12] F. R. Mulyadi and Y. Syahidin, "Rancang Bangun Sistem Informasi Kepegawaian Dengan Metode Waterfall," *Explor. Sist. Inf. dan Telemat.*, vol. 12, no. 2, p. 186, 2021, doi: 10.36448/jsit.v12i2.2056.

[13] Guruh Wijaya, Dudi Irawan, Zainul Arifin, Hardian Oktavianto, Miftahur Rahman, and Ginanjar Abdurrahman, "Studi Klasifikasi Topik Berita Dengan Algoritma Machine Learning," *J-Ensitec*, vol. 11, no. 01, pp. 10202–10206, 2024, doi: 10.31949/jensitec.v11i01.12037.

[14] A. E. Nanda, A. N. Sihananto, and A. M. Rizki, "Analisis Sentimen Pada Pembatalan Tuan Rumah Indonesia Di Piala Dunia U-20 Menggunakan Fasttext Embeddings Dan Algoritma Recurrent Neural Network," *SABER J. Tek. Inform. Sains dan Ilmu Komun.*, vol. 2, no. 2, pp. 246–257, 2024.

[15] H. Hambali, M. Mahayadi, and B. Imran, "Classification of Lombok Songket Cloth Image Using Convolution Neural Network Method (Cnn)," *Pilar Nusa Mandiri*, vol. 17, no. 85, pp. 149–156, 2021, doi: 10.33480/pilar.v17i2.2705.