JIKO (Jurnal Informatika dan Komputer) Accredited KEMDIKTISAINTEK, No.0173/C3/DT.05.00/2025
Vol. 8, No. 3, Desember 2025, pp. 269-274 p-ISSN: 2614-8897
DOI: 10.33387/jiko.v8i3.10783 e-ISSN: 2656-1948

AN LSTM-BASED APPROACH FOR INDONESIAN NEWS CATEGORIZATION: PERFORMANCE ANALYSIS OF HYPERPARAMETER TUNING AND PREPROCESSING

Iwan La Udin[1*], Firman Tempola[2], Abdul Mubarak[3], Muhammad Sabri Ahmad[4], Munazat Salmin[5], Saiful Do Abdullah[6]

[1,2,3,4,5] Program Studi Informatika, Fakultas Teknik, Univeristas Khairun, Ternate, 97719, Indonesia
Email: [*1]iwanlaudin@gmail.com, [2]firman.tempola@unkhair.ac.id ,[3]amuba@unkhair.ac.id,
[4]m.sabriahmad@unkhair.ac.id, [5]munazatsalmin@unkhair.ac.id, [5]saiful.abdullah@unkhair.ac.id

(Received: 30 September 2025, Revised: 10 November 2025, Accepted: 12 December 2025)

Abstract

News disseminated through internet-based systems or news portals is generally classified into specific categories, such as politics, sports, economy, entertainment, technology, health, and others. Currently, this categorization is performed manually, requiring a thorough reading of the entire news content. To address this inefficiency, an automatic classification system for Indonesian news articles is necessary to categorize them based on predetermined categories. This research employs a Natural Language Processing (NLP) approach and implements the Long Short-Term Memory (LSTM) architecture. The study was conducted using several testing scenarios, including (1) hyperparameter tuning of the learning rate to 0.01 and 0.001, (2) the application and omission of stemming, and (3) various dataset comparison ratios of 60:40, 70:30, 80:20, and 90:10. The evaluation utilized a dataset of 10,000 articles across 5 categories and was measured using accuracy, precision, recall, and f-measure metrics. From the three scenarios, seven training models were generated. The second model, with a learning rate of 0.001, without stemming, and a 90:10 dataset ratio, achieved the highest accuracy of 90.7%, with average precision, recall, and f-measure scores of 91%. The third and fourth models, which applied stemming, did not demonstrate a performance improvement, both yielding an accuracy of 89%. The fifth model, with a 60:40 dataset ratio, produced an accuracy of 90%, while the sixth and seventh models, with 70:30 and 80:20 ratios, resulted in accuracies of 79% and 88%, respectively.

Keywords: *Hyperparameter, Indonesian News Categorization, LSTM, NLP*

**Corresponding Author: Iwan La Udin*

1. INTRODUCTION

Human access to the latest news has become increasingly easy and widespread due to current technological advancements [1]. Technology has transformed the distribution of news from traditional media such as newspapers, magazines, radio, and television to internet-based systems or news portals [2]. Generally, news distributed through these internet-based systems or news portals is classified into specific categories, such as politics, sports, economics, entertainment, technology, health, and others [3].

The current methodology for news classification into designated categories is a manual process, necessitating a comprehensive reading of each article to ensure accurate categorization. This approach is notably inefficient, particularly given the large volume of news articles requiring classification [2]. Moreover, an additional challenge arises from the deliberate miscategorization of articles into more popular categories by certain parties, a practice aimed at artificially inflating the article's readership [1].

To simplify the news categorization process, this research aims to develop a system that can automatically classify news articles into specific categories. One of the deep learning methods proposed in this study is the Recurrent Neural Network (RNN), utilizing the Long Short-Term Memory (LSTM) architecture. LSTM was developed to address the exploding and vanishing gradient problems encountered when training traditional RNNs. Its ability to handle significant complexity makes LSTM a suitable machine learning algorithm for document classification [4].

Previous research on news category classification was conducted by [5] [6], who applied the Support Vector Machine (SVM) method. Such classification requires a feature extraction process, which can become slow and computationally intensive when dealing with large datasets. In contrast, this study implements a Recurrent Neural Network (RNN) model. RNNs have been previously applied in various text classification tasks, as demonstrated by [7]-[10], and have shown to achieve excellent model performance.

2. RESEARCH METHOD

The research stages, which guide the researcher through the research process, are illustrated in Figure 1.

2.1 Data Collection

The data utilized in this study is sourced from the Indosum dataset [11], which is available in the kata-ai repository at https://github.com/kataai/indosum. This dataset comprises 19,000 news article pairs across six categories: entertainment, inspiration, showbiz, headlines, technology, and sports, collected from online news portals such as CNN Indonesia and Kumparan. For the purpose of this research, a subset of 10,000 data points from five of these news article categories was used. A more detailed breakdown is presented in Table 1.

2.2 Text Preprocessing

Text preprocessing is a crucial stage for transforming raw text data into a more structured and manageable format. There are no rigid rules governing the exact sequence of preprocessing steps; the specific stages employed depend on the nature of the data being processed [1]. The primary text preprocessing stages include case folding, tokenizing, stopwords removal, and stemming. An explanation of each stage is provided below.

Case Folding: This step involves converting all alphabetic characters ('a' through 'z') in the text document to a single case, typically lowercase. Any characters that are not letters are removed or treated as delimiters. This ensures uniformity across the text.

Tokenizing: This is the process of breaking down a stream of text into individual words or terms, known as tokens. The text is typically split based on spaces between words. This stage often includes the removal of numbers and punctuation marks to isolate the meaningful words.

Stopword Removal: This stage focuses on filtering out common words that provide little semantic value to the text. These "stopwords" are typically function words or irrelevant vocabulary that appear frequently but do not help in differentiating documents, such as conjunctions ("and," "but"), prepositions ("in," "on"), and articles ("the," "a").

Stemming: The goal of stemming is to reduce words to their root or base form (the "stem"). This is achieved by removing prefixes and suffixes from each word. For example, the words "running," "ran," and "runner" would all be reduced to the stem "run." This helps in grouping related words and reducing the dimensionality of the data.

2.3 Long short-term memory

Long Short-Term Memory (LSTM) is a variant of the Recurrent Neural Network (RNN) that was created to address the long-term dependency problem found in traditional RNNs [12]. The architecture of an LSTM unit is illustrated in Figure 2.

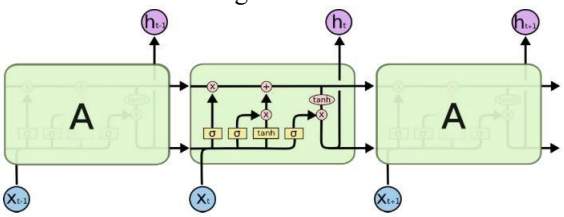


Figure 2. LSTM Architecture

In the workflow of each memory cell within every LSTM neuron, there are four activation function processes known as gate units. These gate units consist of the forget gate, input gate, cell gate, and output gate [13].

The initial step in the LSTM process is the forget gate, which determines what information will be discarded from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." This layer processes ht−1 and xt as inputs and produces an output of either 0 or 1 for the cell state Ct−1 [14]. The equation for the forget gate is detailed in Equation 1.

$$f_t = \sigma(W_f \,.\, [h_{t-1}, x_t] + b_f) \quad (1)$$

The second step is to decide what information to store in the cell state. This stage consists of two parts. The first part is a sigmoid layer, called the input gate layer, which decides which values will be updated. Next, a tanh layer creates a vector of new candidate values, C~t, that could be added to the cell state. In the subsequent step, the outputs of the input gate layer and the tanh layer are combined to update the cell state [11]. The equations for the input gate and the new candidate values are detailed in Equations 2 and 3.

$$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i) \quad (2)$$

$$C_t = tanh(W_c \,.\, [h_{t-1}, x_t] + b_c) \quad (3)$$

The third step is to update the old cell state, Ct−1, to the new cell state, Ct. This is accomplished by multiplying the old cell state by ft to discard the information that was decided to be forgotten in the forget gate layer. Then, the result is added to the product of it and C~t−1, which represents the new candidate values that will be used to update the cell state [11]. The cell state equation is as follows:

$$C_t = \sigma(f_t * C_{t-1} + i_t * C_{t-1}) \quad (4)$$

The fourth step in the LSTM method is to determine the output. This output is based on the processed cell state. First, a sigmoid layer decides which parts of the cell state will be included in the output. Then, the cell state is passed through a tanh

layer and multiplied by the output of the sigmoid gate. This ensures that the final output aligns with the parts selected by the sigmoid layer [15]. The equations for the output gate and the hidden state are detailed in Equations 5 and 6, respectively.

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

3. RESULT AND DISCUSSION

The results of the research are based on a logical sequence to form a story. The contents show facts/data. Can use Tables and Numbers but do not repeat the same data in pictures, tables, and text. To further clarify the description, can use subtitles.

Discussion is the basic explanation, relationship, and generalization shown by the results. The description answers a research question. If there are any dubious results then show them objectively.

4. 1 Dataset

The dataset used in this study consists of 10,000 data entries. The data is distributed equally across five categories, with 2,000 entries for each: entertainment, sports, showbiz, headlines, and technology. This distribution is illustrated in Figure 2.

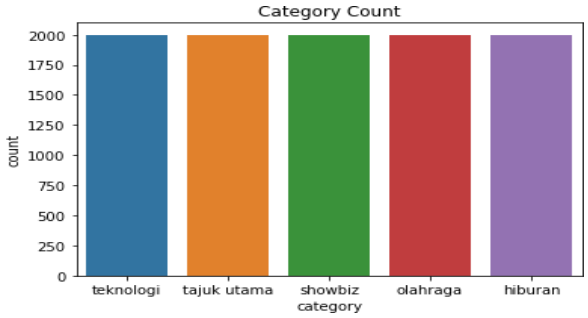


Figure 2. dataset details graph

3.2 Dataset Splitting

Before being processed by the LSTM model, the dataset is first divided into training data (90%) and testing data (10%). The dataset is randomly partitioned using the train_test_split function from the scikit-learn (sklearn) library.

3.3 Training Model

In this stage, three training scenarios will be developed. The first scenario involves Hyperparameter Tuning for the model, which includes adjusting the number of recurrent units, the dropout rate, the batch size, the learning rate, and the maximum number of epochs. The hyperparameter scenarios are detailed in Table 2.

Tabel 2. Skenario *Hyperparameter*

Model	*Recurrent Units*	*Dropout*	Lr	*Batch Size*	*Epoch*
1	128	0,5	0,01	128	50
2	128	0,5	0,001	128	50

Next, the second scenario investigates the influence of data preprocessing, specifically comparing data with and without the application of stemming, on the hyperparameter scenario outlined in Table 2.

The third scenario involves experimenting with different training and testing data split ratios: 60:40, 70:30, and 80:20.

In each scenario, a callback function was implemented to halt the training process when the validation error reached its minimum value. In other words, the callback function stops the training if overfitting occurs, even if the maximum number of epochs has not been reached.

Model 1

Model 1 was trained without applying stemming and with a learning rate of 0.01. The training process yielded an accuracy of 88.11% with a loss value of 0.34, and a validation accuracy of 89.22% with a validation loss of 0.42. As shown in Table 2, the average test values for precision, recall, and F-Measure were 88%, with a final testing accuracy of 88.4%. Although overfitting did not occur, the resulting loss value was still high, which prevented the model from achieving maximum accuracy. The comparative graph of accuracy and loss can be seen in Figure 3.

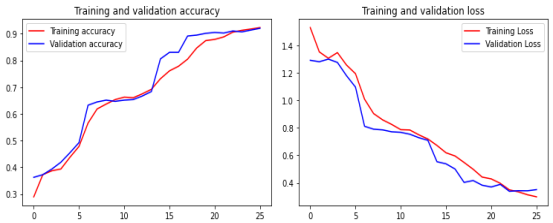
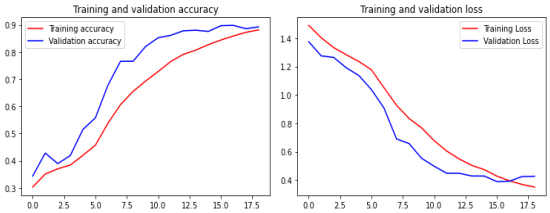


Figure 3. Accuracy and Loss Graph of Model 1

Model 2

Model 2 was trained without the application of stemming, utilizing a learning rate of 0.001. During the training process, it achieved an accuracy of 92.26% with a loss value of 0.29. The validation accuracy reached 92% with a corresponding validation loss of 0.35.

The average testing values for precision, recall, and F-Measure, as presented in Table 3, were 91%, with a final testing accuracy of 90.7%. In other words, this model yielded the highest accuracy compared to the seven other models that were trained. Overfitting did not occur, as indicated by the lower validation loss value and the consistent increase in accuracy. The



comparative graphs for the accuracy and loss of Model 2 can be observed in Figure 4.

Figure 4. Accuracy and Loss Graph of Model 2

Model 3

In Model 3, training was conducted by applying stemming and using a learning rate of 0.01. The training process achieved an accuracy of 93.23% with a loss value of 0.23, and a validation accuracy of 91.44% with a validation loss of 0.39. The average testing values for precision, recall, and f-measure, as shown in Table 4, were 89%, with a final testing accuracy of 89.3%.

The accuracy graph (Figure 5) indicates that the model did not experience overfitting, as evidenced by the consistent trend between the training accuracy and validation accuracy percentages, as well as the stable average testing metrics. Furthermore, the resulting loss value in this model was lower than that of Model 1. The comparative graph of accuracy and loss can be seen in Figure 5.

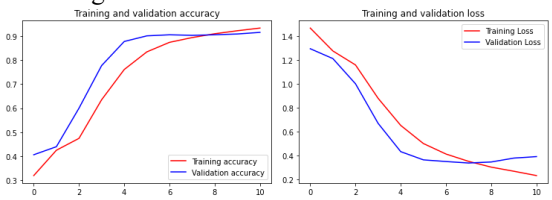


Figure 5. Accuracy and Loss Graph of Model 3

Model 4

In Model 4, training was conducted by applying stemming with a learning rate of 0.001. The training process yielded an accuracy of 92.57% with a loss value of 0.29, and a validation accuracy of 90.67% with a validation loss of 0.36.

As shown in Table 5, the average testing values for precision, recall, and f-measure were 89%, with a final testing accuracy of 89.1%. The progression of loss and accuracy values from the 1st to the 37th epoch indicates that the model did not experience overfitting, as the differences between the training and validation metrics were not significant. This trend is illustrated in the graph in Figure 6.

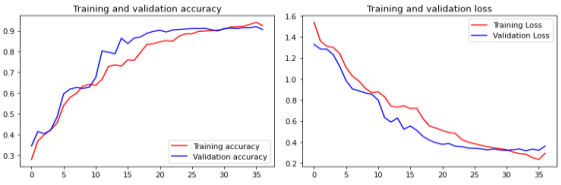


Figure 6. Accuracy and Loss Graph of Model 3

The evaluation process was conducted on seven models, utilizing varied configurations for the learning rate, the application of stemming, and the ratios of training and testing data. The results for all tested models are presented in Table 3.

Tabel 3. Perbandingan Kinerja Evaluasi 7 Model Pengujian

Model	*Units*	Lr	*Acc*	*Stemming*	*dataset*
1	128	0,01	88%	No	90:10
2	128	0,001	91%	No	90:10
3	128	0,01	89%	*Yes*	90:10
4	128	0,001	89%	*Yes*	90:10
5	128	0,001	90%	No	60:40
6	128	0,001	79%	No	70:30
7	128	0,001	88%	No	80:20

Based on Table 3, the highest model accuracy was achieved with a learning rate of 0.001, without the application of stemming, and using a 90:10 dataset ratio, resulting in an accuracy of 91%. In contrast, models that incorporated stemming did not show an improvement in performance, yielding an accuracy of only 89% for the two models tested, which is lower than the models without stemming. This outcome is likely because the stemming process can alter the meaning of words within the articles, making it more difficult for the model to recognize patterns specific to each category.

Furthermore, the influence of the training data size on model performance was inconsistent. Model 5, with a 60:40 dataset ratio, achieved a higher accuracy than both Model 6 (70:30 ratio) and Model 7 (80:20 ratio). However, the performance of Model 5 was still lower than that of Model 2, which used a 90:10 dataset ratio.

3.4 Interface Implementation

The home page consists of two menu options: the model summary page and the identification page. The implementation of the home page can be seen in Figure 7.

Figure 7. page home

On this page, the application will provide the user with information regarding the overall performance of the model. This includes the training and validation accuracy graph, the training and validation loss graph, as well as the percentage values for precision, recall, f-measure, and the overall accuracy of the model. For a clearer illustration, please refer to Figure 8.

Figure 8. Page Summary Model

This page serves as the text identification interface, where users can identify a news article by directly inputting the text into the provided form. This is illustrated in Figure 9.

Figure 9. Text Identification Page

The file identification page allows the user to upload a news article for classification. The system is designed to exclusively process files with a .txt extension on this page, as illustrated in Figure 10.

Figure 10. txt File Identification Page.

4. CONCLUSION

Based on the testing and evaluation results, it can be concluded that the LSTM method with a hyperparameter tuning of a 0.001 learning rate, without the application of stemming, and using a 90:10 dataset ratio, successfully classified news articles with the highest accuracy of 91%. The model with a 0.001 learning rate achieved a higher accuracy compared to the model with a 0.01 learning rate. Furthermore, models that applied stemming during the data preprocessing stage yielded lower accuracy than those without stemming.

The training and testing dataset ratio that produced the highest accuracy in this research was 90:10, achieving an accuracy of 91%. This is in comparison to the 60:40 ratio, which resulted in 90% accuracy, the 70:30 ratio with 79% accuracy, and the 80:20 ratio with 88% accuracy.

The LSTM method was successfully implemented for the automatic text classification of Indonesian language news articles, achieving an accuracy rate of 91%.

5. REFERENCE

[1] Setiawan, A., Santoso, L. W., & Adipranata, R. (2020). Klasifikasi Artikel Berita Bahasa Indonesia Dengan *Naive Bayes Classifier*. Jurnal Infra, 8(1), 146–151.

[2] Findra Kartika Sari Dewi, T. P. A. (2021). Klasifikasi Berita Menggunakan Metode *Multinomial Naive Bayes*. XVI(2017), 1–8.

[3] Sari, W. K., Rini, D. P., Malik, R. F., & Azhar, I. S. B. (2020). Klasifikasi Teks Multilabel pada Artikel Berita Menggunakan *Long Short Term Memory dengan Word2Vec*. Resti, 1(10), 276–

[4] Sari, W. K., Rini, D. P., Malik, R. F., & Azhar, I. S. B. (2020). Klasifikasi Teks Multilabel pada Artikel Berita Menggunakan Long Short Term Memory dengan Word2Vec. Resti, 1(10), 276–285.

[5] Pooja, S and Khanna, V. Multi-category news classification using Support Vector Machine based classifiers. *Applied Sciene* pp, 1-12. 2020. https://doi.org/10.1007/s42452-020-2266-6.

[6] Rizal, M. Fikry, and U. Khalil. News Opinion Classification Application with Support Vector Machine Algorithm Using Framework Codeigniter. *JITE (Journal of Informatics and Telecommunication Engineering)* Vol 5. No 1. Pp. 160-166. 2021. DOI : 10.31289/jite.v5i1.5189.

[7] H. Hu, M. Liao, C. Zhang and Y. Jing. Text classification based recurrent neural network. *IEEE 5th Information Technology and Mechatronics Engineering Conference*. Pp 652-655.

[8] T. Praha, W. Widodo, and M. Nugraheni. Indonesian Fake News Classification Using Transfer Learning in CNN and LSTM. *JOIV : International Journal on Informatics Visualization*. Vol 8. No 3. Pp. 1213-1221. 2024. http://dx.doi.org/10.62527/joiv.8.3.2126

[9] R. Saputra, A. Waworuntu, A. Rusli. Classification of Indonesian News using LSTM-RNN Method. *6th International Conference on New Media Studies (CONMEDIA)*. 2021. DOI: 10.1109/CONMEDIA53104.2021.9617187

[10] N. Rai, et. all. Fake News Classification using transformer based enhanced LSTM and BERT.

International Journal of Cognitive Computing in Engineering vol. 3. Pp. 98-105. 2022. https://doi.org/10.1016/j.ijcce.2022.03.003

[11] Kurniawan, K., & Louvan, S.. IndoSum: *A New Benchmark Dataset for Indonesian Text Summarization. Proceedings of the 2018*

[12] Rozi, I. F., Wijayaningrum, V. N., & Khozin, N. Klasifikasi Teks Laporan Masyarakat Pada Situs Lapor! Menggunakan Recurrent Neural Network. *Sistemasi*, 9(3), 633-638. 2020.

[13] Rais, I. L., & Jondri, J. Klasifikasi Data Kuesioner dengan Metode Recurrent Neural Network. EProceedings of Engineering, 7(1), 2817–2826. 2020.

[14] W. Afandi, et al. Klasifikasi Judul Berita Clickbait menggunakan RNN-LSTM. *Jurnal Informatika: Jurnal Pengembangan IT*. Vol 7. No 1. Pp. 85-89. 2022.

[15] M. R. Jhaerol and S. Sedianto. Implementation Of Chatbot For Merdeka Belajar Kampus Merdeka Program Using Long Short-Term Memory. *Jurnal Nasional Pendidikan Teknik Informatika*. Vol 12. No 2. Pp. 253-262. 2023. https://doi.org/10.23887/janapati.v12i2.58794