Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

Abstractive Summarization of Indonesian Islamic Stories Using Long Short-Term Memory (LSTM)

Aisya Gusti Savila[*1], Supriyono[2], Roro Inda Melani[3]

[1,2,3]Informatics Engineering, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia

Email: [1]aisyag209@gmail.com

Received : Jun 20, 2025; Revised : Sep 6, 2025; Accepted : Sep 22, 2025; Published : Feb 15, 2026

Abstract

The length of narratives in stories often poses a challenge for many readers, especially those with time constraints or difficulty understanding the entire story. In this case, summarization offers a solution, but manual summarization is not always efficient in meeting the need for quick and concise information. This study aims to develop an automatic text summarization system for Islamic stories using the Long Short Term Memory (LSTM) algorithm. The study employs three data splitting scenarios for training and testing: 90:10, 80:20, and 70:30. Testing results show that the highest training accuracy was achieved in the 80:20 scenario with a value of 89.44%. This does not entirely indicate that a smaller proportion of training data will always result in higher accuracy, as this improvement can be influenced by data variation, overfitting conditions, and early stopping performance. Therefore, the data division ratio influences the training process. Although the highest training accuracy was obtained in the 80:20 scenario, the best semantic summary quality was found in the 90:10 scenario. In the 90:10 scenario, the ROUGE-1 evaluation score achieved a precision of 0.4147, a recall of 0.2516, and an F1-score of 0.3027. Meanwhile, ROUGE-2 achieved a precision of 0.1022, a recall of 0.0568, and an F1-score of 0.0684. Meanwhile, ROUGE-L achieved a precision of of 0.2017, recall of 0.1209, and F1-score of 0.1459.

Keywords : *Abstractive Summarization, Islamic Stories, Long Short Term Memory, ROUGE, Text Summarization, Word2Vec*

1. INTRODUCTION

Text summarization is an important process in conveying information, especially in this modern era when the amount of digital content is increasing exponentially[1]. In the context of Indonesian-language Islamic stories, these texts often contain moral messages, life values, and religious teachings that are highly relevant to readers from various backgrounds[2]. However, the length of the narratives in these stories often poses a challenge for many readers, especially those with time constraints or difficulty understanding the entire story. In this regard, summarization offers a solution, but manual summarization is not always efficient in meeting the need for quick and concise information[3][4].

Artificial intelligence models such as Long Short-Term Memory (LSTM) offer more practical and effective solutions. LSTM, as part of an artificial neural network, has the ability to understand temporal context in text, enabling it to generate concise yet meaningful summaries[5]. This advantage makes it a promising alternative to manual methods, especially in the processing of long and complex texts[6][7].

Previous research has shown that LSTM excels at generating better summaries than traditional methods and other models such as Transformer and RNN. For example, in the case of TEDx transcripts, LSTM has been shown to provide time efficiency while maintaining the quality of the results[8]. However, the application of this model to Indonesian presents its own challenges. Indonesian has a

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

unique syntactic and semantic structure, so the LSTM model requires special adaptation with relevant training data in order to produce accurate summaries[9].

In previous studies, many algorithms have been used to summarize news texts or documents. The LSTM algorithm has higher accuracy than other algorithms such as Transformer or RNN[10]. However, the implementation of LSTM has not yet been explored in the context of summarizing Islamic stories. Therefore, this study proposes the development of an LSTM-based automatic text summarization system, specifically for videos containing Islamic stories in Indonesian. This system aims to generate summaries that are not only concise and efficient but also preserve the moral messages, life values, and religious teachings contained in the text[11].

2. METHOD

This study uses a quantitative approach with an experimental design to test the application of the Long Short-Term Memory (LSTM) model in summarizing Indonesian-language Islamic stories[12]. The methodology applied in this study consists of several main stages, starting from data collection, data preprocessing, feature extraction, LSTM model training, to evaluation of the text summarization results. This process is described in detail in Figure 1 below.

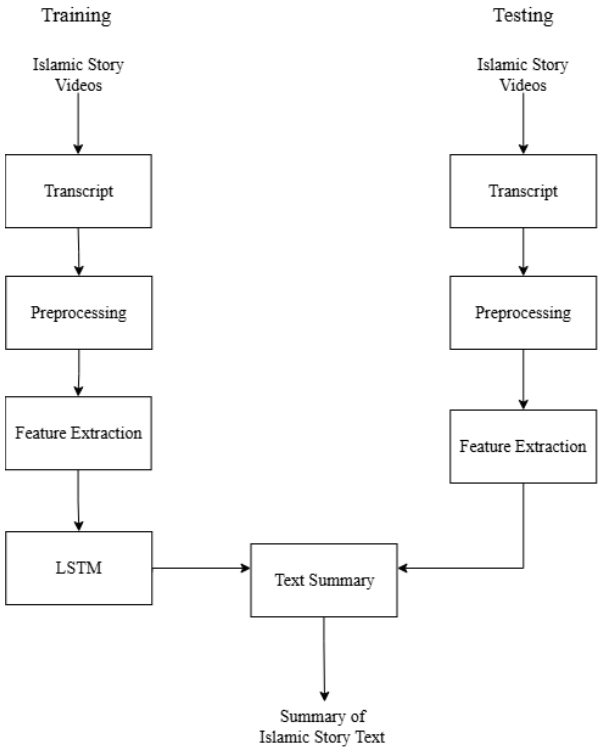

Figure 1. Research Design

2.1. Data Collection

The data used in this study was collected from YouTube, which provides various videos containing Islamic stories in Indonesian. These videos contain stories about the lives of the Prophet, his companions, and Islamic figures that are full of moral messages and religious values.

Transcripts of these videos were obtained using the YouTube API to obtain data in text form relevant to the research topic. Videos were selected based on specific criteria, namely videos that were long in duration and contained significant moral messages. Once the transcripts were obtained, the data was further processed in the preprocessing stage.

P-ISSN: 2723-3863 https://jutif.if.unsoed.ac.id
E-ISSN: 2723-3871 DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

2.2. Transcript

Automatically extracting transcripts from each video, translating them into Indonesian, and saving them to a CSV file on Google Drive. The process begins by mounting Google Drive so that the CSV file can be saved directly to that directory. Next, the YouTube API Key is configured to access the YouTube Data API and establish a connection to the service using the build.

Next, the get_videos_from_playlist() function retrieves all videos from the specified playlist via the playlist_id, including videoId, title, and link information. For each video obtained, the get_transcript() function attempts to retrieve the transcript using the YouTubeTranscriptApi library, trying various languages sequentially such as Indonesian, English, Arabic, and Malay. The retrieved transcript text will first be processed by removing words such as “clapping,” “music,” or “laughing,” as well as non-alphabetic characters.

Once the transcript is obtained, the translate_to_indonesian() function will attempt to translate the text into Indonesian using the deep_translator library. If the translation process fails or the text is empty, the original transcript is retained. All data (video title, link, and translated transcript) is then saved to a CSV file using the csv library, with the file name FULL KISAH.csv stored in the Google Drive folder /content/drive/My Drive/CRAWLING DATA/. Every time a video is successfully processed. Once all processes are complete, the system will display a message indicating that the data has been successfully saved.

2.3. *Preprocessing*

In the preprocessing process, the collected data will be inputted for processing, starting from data cleaning, lowercasing, and then tokenization[13]. The purpose of this preprocessing process is to clean the data so that it is ready to be processed and entered into the system.

a. Data Cleaning
   The first stage in processing is data cleaning, which involves removing all punctuation marks such as periods, commas, exclamation marks, and others from the text. Next, all numbers are removed from the text because they typically do not provide important information in the context of text summarization processing[14]. Then, excessive spaces are cleaned up, which involves replacing double spaces or tabs with single spaces and removing spaces at the beginning or end of sentences. Finally, single quotes (') and double quotes (") are removed because they are considered to add no significant meaning to the sentence[15]. With these cleaning steps, the text becomes cleaner, more uniform, and ready for further processing such as tokenization.
b. Lowercasing
   The second stage in data processing is lowercasing. This process aims to standardize the format of words to prevent duplication caused by differences in capitalization. For example, the words “School,” “school,” and “SCHOOL” actually refer to the same word, but if they are not converted to lowercase, all three will be considered different tokens in text analysis. By applying lowercasing, the entire text such as “I Study at School” will be converted to “i study at school.” This process helps improve data consistency and is highly beneficial in subsequent stages[16].
c. Tokenization
   The third stage of processing is tokenization, where text is divided into smaller parts, called tokens, which can be words, phrases, or characters. Tokenization enables computers to understand text in a structured manner and facilitates further analysis. The purpose of this process is to simplify the text and convert it into units that are easier for natural language processing algorithms to analyze[14]. Tokenization is an important first step before performing other processes.

Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

2.4. Feature Extraction with Word2Vec

At this stage, feature extraction is performed by converting words in the text into numerical representations in the form of vectors using word embedding techniques, one of which is Word2Vec[17]. Word2Vec converts words in the text into low-dimensional vectors that reflect the semantic relationships between words based on their usage context. There are two main approaches in Word2Vec: Continuous Bag of Words (CBOW), which predicts the target word based on the surrounding words, and Skip-gram, which predicts the contextual words based on the target word[18]. This study uses the Skip-gram approach, where the model is given the target word and attempts to predict the contextual words within a specified window[19]. This process is calculated using conditional probability with the softmax formula to determine the likelihood of context words appearing based on the target word. This process is performed by calculating the conditional probability using the softmax formula, which measures the likelihood of context words appearing based on the target word. Before calculating the conditional probability, the average log probability of context words appearing around the target word (w_t) is calculated. With the equation (1).

$$J = \frac{1}{T}\sum_{t=1}^{T}\sum_{-k\leq j\leq k, j\neq 0} \log P(w_{t+j}|w_t) \quad (1)$$

Next, conditional probability calculations are performed using the softmax formula, which describes the relationship between the target word (w_t) and context words (w_c) in equation (2).

$$P(w_c \mid w_t) = \frac{exp\,(v_{w_O}^{T}.v_{w_I})}{exp\,(v_{w}^{T}.v_{w_I})} \quad (2)$$

2.5. *Long Short-Term Memory (LSTM)*

LSTM-based text summarization is one effective approach to addressing the problem of automatic summarization[20]. LSTM, as part of an artificial neural network, has the unique ability to understand temporal context in sequential data, such as text, enabling it to capture complex relationships between words in a document. To obtain an optimal model, the network architecture shown in Figure 2 was created.

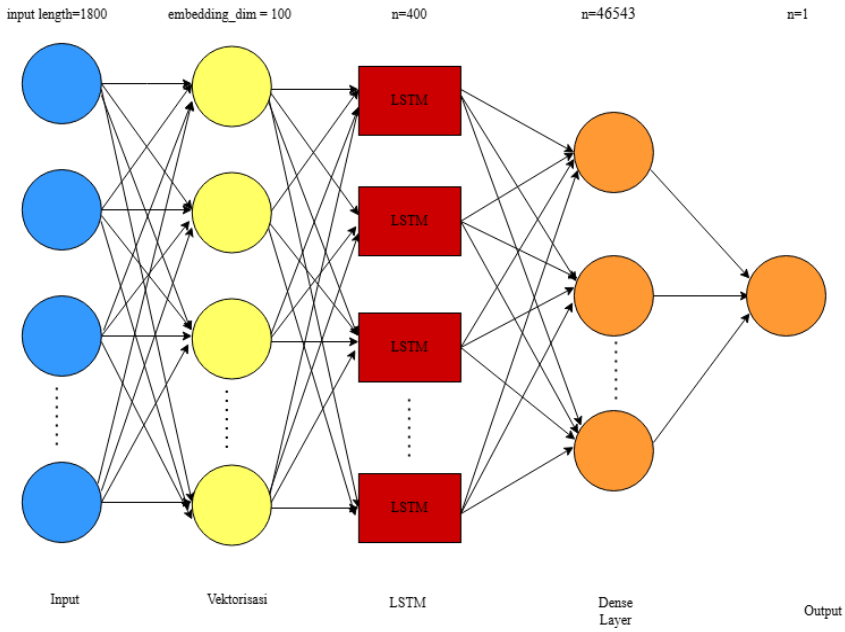

Figure 2. LSTM Network Architecture

Figure 2 shows the LSTM architecture, where this model accepts inputs with a maximum length of 1800, each input consisting of 1800 tokens or words to be processed by the model. This value represents the input data in the form of a text sequence consisting of words or tokens that have been separated. Each word or token in the input is then represented using an embedding technique with a dimension of 100 and a context window of 10, where each word or token is converted into a vector. This embedding dimension allows the model to capture the semantic meaning of each word. After the embedding is done, the input data that has been converted into vectors is passed to the LSTM layer, which has 400 units. This LSTM layer serves to capture sequence patterns and long-term dependencies between words in the text. The output from the LSTM layer is then passed to a dense layer to filter and interpret the information extracted by the LSTM. The number of target classes or labels to be predicted is 46,543 classes, which likely represent the output words or tokens in the scenario. The output from the dense layer is then fed into the softmax layer. The softmax function converts the raw scores (logits) from the dense layer into a probability distribution, where the total sum of all output values is 1[21]. The highest value from the softmax result indicates the class most likely to be the target output, for example, the next word to be predicted in the text sequence.

The first step in the LSTM process is to determine which information needs to be stored or deleted in the memory cell through the forget gate. This forget gate functions to decide whether information from the previous cell should be retained or deleted. This process is carried out by applying the sigmoid function, which outputs values between 0 and 1, where 0 indicates that the information should be completely forgotten, and 1 means that the information should be retained[22]. The following is equation (3)

$$ft = (Wf[ht-1, xt] + bf) \tag{3}$$

The next step is to update the cell state by determining the information to be stored in it. This is done through the input gate layer in equation (4), which determines the value to be updated using the sigmoid activation function, and the tanh layer, which generates new candidate values using the tanh activation function in equation (5).

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{4}$$

$$C_t = tanh\,(W_C \times [h_{t-1}, x_t] + b_C) \tag{5}$$

The next step is to update the old cell state to the new cell state by integrating information from the forget gate and input gate. This process is done by multiplying the forget gate value by the previous cell state, then adding the result to the value from the input gate according to equation (6).

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{6}$$

The final step in the LSTM process is to determine the output of the cell state. This is done using the output gate in equation (7), which determines which part of the cell state will become the output. This output is then fed into the tanh layer to be given the tanh activation function, and the result is multiplied by the output of the output gate as in equation (8).

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \times tanh(C_t) \tag{8}$$

After output from the last LSTM layer, the data is passed to the Dense layer. This layer is responsible for converting complex feature representations from LSTM into raw scores (logits). These scores are linear values that have not been translated into probabilities. In the Dense layer, calculations are performed using weights and biases in equation (9).

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

$$z = W.h + b \quad (9)$$

The results from the Dense Layer are passed to the Softmax layer to be converted into a probability distribution. Softmax ensures that all output values are in the range [0, 1] and that their total sum is 1.

3. RESULT

3.1. Data

Data preparation involves dividing the dataset into two main parts: training data and testing data[23]. Proper data division is essential to ensure that the model can be trained properly and evaluated objectively. Table 1 shows three scenarios for dividing training and testing data with different ratios.

Table 1. Ratio Data

Rasio	Data	
	Training	***Testing***
90:10	270	30
80:20	240	60
70:10	210	90

3.2. Testing Scenario

To test the performance of the LSTM model in text summarization, two testing scenarios were applied, namely testing with direct data division (train-test split) and testing using ROUGE[24].

3.2.1. Split 90:10

Testing with a ratio of 90:10 was carried out by dividing the dataset into 90% for training with 270 data points and 10% for testing with 30 data points.

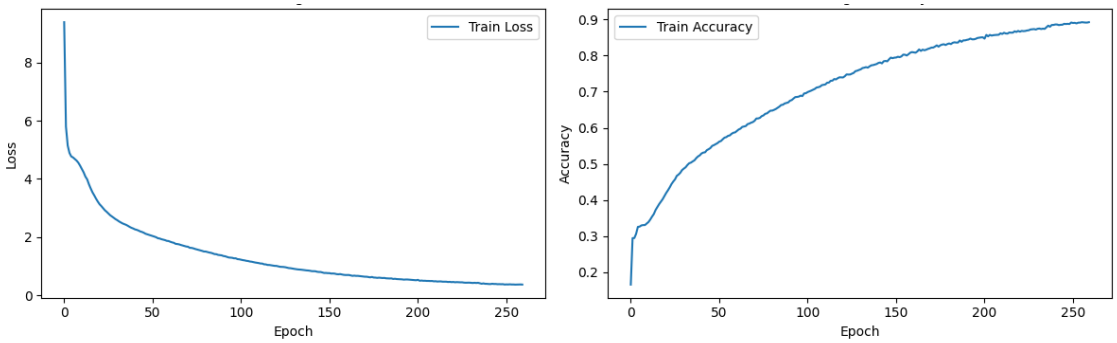

Figure 3. Accuracy and Training 90:10

Figure 3 shows the performance of the LSTM model training in completing the task of summarizing Islamic stories using a learning scheme of 228 epochs using 90% of the training data (270 data). On the left graph, it can be seen that the training loss value shows a significant decrease from the start of training, beginning at 10.0120 in the first epoch and consistently decreasing until reaching 0.4879 in the 228th epoch. Meanwhile, the graph on the right shows that the training accuracy continues to increase from an initial value of 0.1060 (1.06%) to 0.8582 (85.82%) at the end of the 228th epoch. The learning rate starts at 0.0010 and decreases adaptively, which helps stabilize training in the final stage, thereby avoiding overfitting. These results indicate that the model successfully learned from the training data, despite some fluctuations in accuracy values during the middle of the epoch.

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

3.2.2. Split 80:20

Testing with a ratio of 80:20 was carried out by dividing the dataset into 80% for training with 240 data points and 20% for testing with 60 data points.

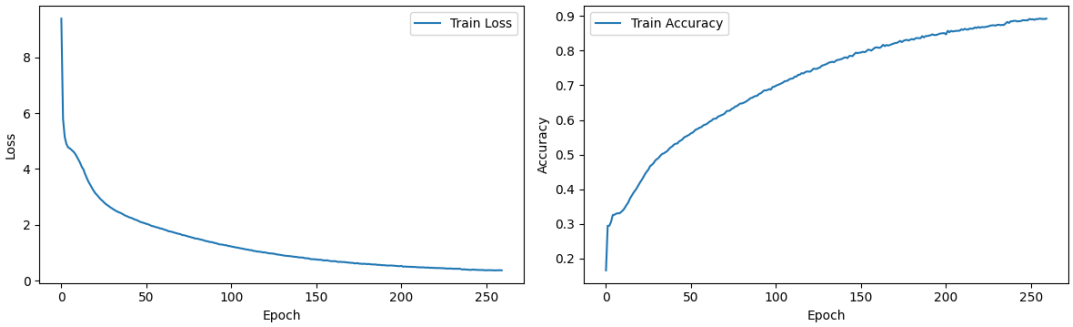

Figure 4. Accuracy and Training 80:10

Figure 4 shows that the model training process was carried out for 280 epochs, with performance improvements marked by increased accuracy and decreased loss. In the first epoch, the model showed an initial accuracy of 0.0896 with a loss value of 10.1044, reflecting initial inaccuracy in predicting the output. During training, the model's accuracy gradually increased to 89.44% in epoch 260, with a significant decrease in loss to 0.3589. The learning rate started at 0.0010 and gradually decreased to prevent overfitting and maintain model stability at the end of training. Although there were small fluctuations in accuracy at some epochs, the overall trend showed that the model was able to learn patterns from the training data well. These results confirm the model's success in gradually understanding the data, producing predictions with minimal error at the end of training.

3.2.3. Split 70:30

Testing with a 70:30 ratio was done by splitting the dataset into 70% (210 data points) for training and 30% (90 data points) for testing.

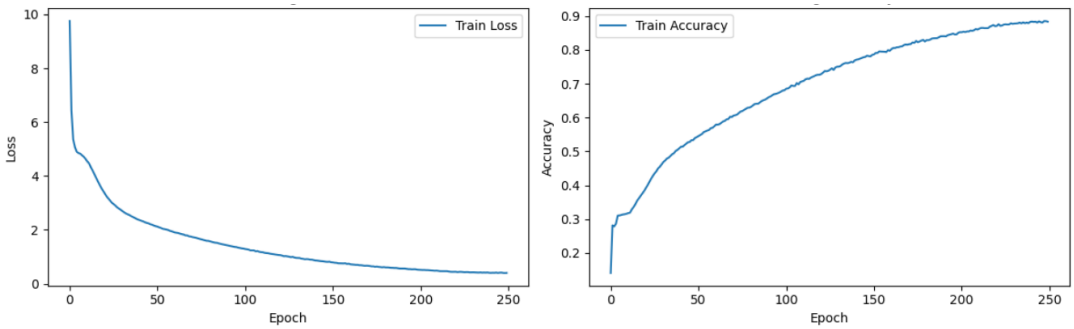

Figure 5. Accuracy and Training 90:10

Based on Figure 5, it shows the training process of the LSTM model over 250 epochs with a training data proportion of 70% (210 data points) of the total dataset. The graph on the left shows a consistent decrease in the loss value from the beginning to the end of the 250-epoch training. At the first epoch, the loss value was still very high at 10.3007, but it continued to decrease until it approached 0.4131 at the 250th epoch. This decrease indicates that the model successfully learned patterns in the data gradually and steadily. Meanwhile, the graph on the right shows an increase in training accuracy as

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

the number of epochs increases. Accuracy at the beginning of training was only around 0.0760 (7.6%), but it continued to increase significantly and reached 0.8805 (88%) at the last epoch.

Table 2. ROUGE Score Results All Ratios

Skenario	Metriks	Precision	Recall	F1
SKENARIO 1 (90:10)	ROUGE-1	0.4147	0.2516	0.3027
	ROUGE-2	0.1022	0.0568	0.0684
	ROUGE-L	0.2017	0.1209	0.1459
SKENARIO 2 (80:20)	ROUGE-1	0.3871	0.2450	0.2863
	ROUGE-2	0.0858	0.0511	0.0595
	ROUGE-L	0.1874	0.1147	0.1353
SKENARIO 3 (70:30)	ROUGE-1	0.3762	0.2434	0.2805
	ROUGE-2	0.0812	0.0478	0.0550
	ROUGE-L	0.1823	0.1159	0.1341

Table 2 shows that the 90:10 scenario produces the highest ROUGE score compared to the 80:20 and 70:30 scenarios. This is true for all three ROUGE metrics, namely ROUGE-1, ROUGE-2, and ROUGE-L, as well as for all three evaluation aspects, namely Precision, Recall, and F1-score. With a ROUGE-1 Precision value of 0.4147, which is the highest value compared to other scenarios. This indicates that when there is more training data (90%), the model has a better ability to generate words that are relevant to the reference summary. ROUGE-1 Recall is also the highest in this scenario, at 0.2516, followed by an F1-score of 0.3027, indicating a good balance between the accuracy and completeness of the information generated by the model. In the ROUGE-2 and ROUGE-L metrics, Scenario 1 also recorded the highest score, although the value is still relatively low (for example, ROUGE-2 F1 is only 0.0684), indicating that the model still has difficulty forming complex inter-word relationships or sentence structures. As the proportion of training data decreases in Scenario 2 (80:20) and Scenario 3 (70:30), it can be seen that all metric values decrease, including precision, recall, and F1-score. This indicates that the less training data used, the more the model's ability to understand context, capture important information, and construct meaningful summaries decreases. This decline is due to the fact that a smaller amount of data does not provide sufficient pattern diversity for the model to learn, causing the model to overfit to the limited training data. Therefore, it can be concluded that the availability of more training data significantly contributes to improving the quality of summaries generated by the model.

4. DISCUSION

Based on the results of testing three scenarios in Table 2data split ratios (90:10, 80:20, and 70:30), it can be concluded that differences in ratios do affect the accuracy of model training, but do not directly reflect the quality of the resulting summaries. The 90:10 scenario yields the highest training accuracy of 85.82%, followed by 80:20 at 89.44%, and 70:30 at 88%. This does not entirely indicate that a smaller proportion of training data will always result in higher accuracy, as this improvement can be influenced by data variability, overfitting conditions, and early stopping performance. Therefore, the data splitting ratio influences the training process but is not always linearly related to the quality of the summary results. However, in terms of evaluating summary quality using the ROUGE metric, the 90:10 scenario

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

actually shows the best performance in generating relevant, complete, and manual-like summaries. This indicates that high training accuracy does not always correlate with the model's ability to understand and reconstruct text narratives, so the selection of the data ratio should not only consider accuracy but also the balance between training performance and the semantic quality of the model's output. Thus, the 90:10 ratio was chosen as the optimal configuration in this study.

Table 3. Comparison of Results with Previous Researchers

Researcher	ROUGE-1	ROUGE-2	ROUGE-L
[25]	25.64	8.77	24.05
[26]	27.71	1.82	27.27
Researcher	30.27	6.84	14.59

Based on a comparison of Table 3 with previous studies, the model used by researchers in this study produced a ROUGE-1 score of 30.27, a ROUGE-2 score of 6.84, and a ROUGE-L score of 14.59 on the Indonesian-language Islamic story dataset. Although the metric values are still lower than those of international studies, the ROUGE-1 score yields higher results than previous researchers and demonstrates stable performance on religious narrative texts. This indicates that the LSTM model with simple preprocessing (cleaning, lowercasing, tokenization) is effective for Islamic narrative texts, although there is still room for performance improvement, particularly in sentence structure and word pairs (ROUGE-2).

5. CONSLUSION

This study successfully developed an automatic text summarization system using the LSTM algorithm for Islamic stories. The results showed that the model was able to produce summaries that were quite relevant to the original content of the stories, especially in recognizing important words. However, its ability to construct complete and coherent sentences still needs to be improved. The quality of the generated summaries is also influenced by the ratio of training and testing data. The larger the proportion of training data, the better the model tends to perform in understanding and organizing the content of the summary. Based on the results of testing three scenarios for dividing training and testing data (90:10, 80:20, and 70:30), it can be concluded that the LSTM algorithm is capable of generating summaries of Islamic stories with varying degrees of accuracy, depending on the amount of training data used. Although the highest training accuracy was obtained in the 90:10 scenario at 85.82%, followed by the 80:20 scenario at 89.44%, and the 70:30 scenario at 88%, these accuracy values do not fully reflect the semantic quality of the summaries. This indicates that although the model is considered accurate in training, its ability to generate meaningful summaries still needs to be assessed based on the evaluation results of the summary content itself. The evaluation results using the ROUGE metric show that the 90:10 scenario provides the best performance. On the ROUGE-1 metric, the precision value is 0.4147, recall is 0.2516, and the F1-score is 0.3027. On the ROUGE-2 metric, the precision, recall, and F1-score values are 0.1022, 0.0568, and 0.0684, respectively. Meanwhile, on ROUGE-L, the precision reached 0.2017, the recall was 0.1209, and the F1-score was 0.1459. These values indicate that the LSTM model is quite capable of producing summaries that are relevant to the original text, especially in selecting important words, although it still has limitations in constructing complete and coherent sentences. Therefore, it can be concluded that the LSTM algorithm has a fairly good level of accuracy in generating summaries of Islamic story texts, especially when supported by an optimal training data ratio, such as in the 90:10 scenario.

REFRENCES

[1] A. Bahari and K. E. Dewi, “Peringkasan Teks Otomatis Abstraktif Menggunakan Transformer Pada Teks Bahasa Indonesia,” *Komputa J. Ilm. Komput. dan Inform.*, vol. 13, no. 1, pp. 83–91,

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

2024, doi: 10.34010/komputa.v13i1.11197.

[2] D. F. AL-Hafiidh, I. F. Rozi, and I. K. Putri, "Peringkasan Teks Otomatis pada Portal Berita Olahraga menggunakan metode Maximum Marginal Relevance.," *J. Inform. Polinema*, vol. 8, no. 3, pp. 21–30, 2022, doi: 10.33795/jip.v8i3.519.

[3] G. Zakawaly, N. Hayatin, and V. R. S. Nastiti, "Improvisasi Algoritma Dijkstra Pada Peringkasan Teks Otomatis Untuk Artikel Politik," *J. Repos.*, vol. 5, no. 2, pp. 709–716, 2023, [Online]. Available: https://repositor.umm.ac.id/index.php/repositor/article/view/1437

[4] D. Argade, V. Khairnar, D. Vora, S. Patil, K. Kotecha, and S. Alfarhood, "Multimodal Abstractive Summarization using bidirectional encoder representations from transformers with attention mechanism," *Heliyon*, vol. 10, no. 4, p. e26162, 2024, doi: 10.1016/j.heliyon.2024.e26162.

[5] A. Khan, I. Ahmad, Q. E. Ali, M. O. Khan, and U. Sana, "Automated Abstractive Text Summarization Using Multidimensional Long Short-Term Memory," vol. 21, no. 02, pp. 1–11, 2023.

[6] G. A. Sandag, "Perbandingan Algoritma LSTM Untuk Analisis Sentimen Pengguna Twitter Terhadap Layanan Rumah Sakit Saat Pandemi Covid-19," *TeIKa J. Teknol. Inf. dan Komun. (Journal Inf. Comunnication Technol.*, vol. 13, no. 01, pp. 31–40, 2023, doi: 10.36342/teika.v13i01.3063.

[7] A. Ravikumar and H. Sriraman, "A Deep Understanding of Long Short-Term Memory for Solving Vanishing Error Problem," in *Advances in systems analysis, software engineering, and high performance computing book series*, 2023, pp. 74–90. doi: 10.4018/978-1-6684-8531-6.ch004.

[8] S. BAYAT and G. ISIK, "Assessing the Efficacy of Lstm, Transformer, and Rnn Architectures in Text Summarization," *Int. Conf. Appl. Eng. Nat. Sci.*, vol. 1, no. 1, pp. 813–820, 2023, doi: 10.59287/icaens.1099.

[9] M. Alfhi Saputra, "Peringkas Teks Otomatis Bahasa Indonesia secara Abstraktif Menggunakan Metode Long Short-Term Memory," *e-Proceeding Eng. Vol.8, No.2 April 2021* |, vol. 8, no. 2, pp. 3474–3488, 2021.

[10] S. Bayat and G. Isik, "Assessing the Efficacy of Lstm, Transformer, and Rnn Architectures in Text Summarization," *Int. Conf. Appl. Eng. Nat. Sci.*, vol. 1, no. 1, pp. 813–820, 2023, doi: 10.59287/icaens.1099.

[11] H. Singh, S. Sood, H. Maity, and Y. Kumar, "Exploring Pre-processing Strategies and Feature Extraction in practical aspect for Effective Spam Detection," 2024, pp. 1–6. doi: 10.1109/iatmsi60426.2024.10502863.

[12] I. N. Purnama and N. N. Widya Utami, "Implementasi Peringkas Dokumen Berbahasa Indonesia Menggunakan Metode Text To Text Transfer Transformer (T5)," *J. Teknol. Inf. dan Komput.*, vol. 9, p. 4, 2023.

[13] A. Zevana and D. Riana, "Text classification using indobert fine-tuning modeling with convolutional neural network and bi-lstm," 2024, doi: 10.52436/1.jutif.2023.4.6.1650.

[14] S. Ketineni and J. Sheela, "Metaheuristic Aided Improved LSTM for Multi-document Summarization: A Hybrid Optimization Model," *J. Web Eng.*, 2023, doi: 10.13052/jwe1540-9589.2246.

[15] B. N. Yashasvi, S. L. J, S. Raghavendra, M. Tiwari, and K. Ananthi, "Improved tweets in English text classification by LSTM neural network," 2023, pp. 1–8. doi: 10.1109/icaeeci58247.2023.10370893.

[16] L. Tao, Z. Cui, Y. He, and D. Yang, "An explainable multiscale LSTM model with wavelet transform and layer-wise relevance propagation for daily streamflow forecasting.," *Sci. Total Environ.*, p. 172465, 2024, doi: 10.1016/j.scitotenv.2024.172465.

[17] Yuyun, A. D. Latief, T. Sampurno, Hazriani, A. O. Arisha, and Mushaf, "Next Sentence Prediction: The Impact of Preprocessing Techniques in Deep Learning," 2023, pp. 274–278. doi: 10.1109/ic3ina60834.2023.10285805.

[18] M. J. Aufa and A. Qoiriah, "Analisis Sentimen Pengguna Platform Belajar Online Coursera menggunakan Random Forest dengan Metode Ekstraksi Fitur Word2vec," *J. Informatics Comput. Sci.*, vol. 04, pp. 244–255, 2023, doi: 10.26740/jinacs.v4n02.p244-255.

Vol. 7, No. 1, February 2026, Page. 158-168
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.1.4918

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2020.

[20] R. S. Kartha *et al.*, "NLP-Based Automatic Summarization using Bidirectional Encoder Representations from Transformers-Long Short Term Memory Hybrid Model: Enhancing Text Compression," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 5, pp. 1223–1236, 2024, doi: 10.14569/IJACSA.2024.01505124.

[21] V. Boppana and P. Sandhya, "Distributed Focused Web Crawling for Context Aware Recommender System using Machine Learning and Text Mining Algorithms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, 2023, doi: 10.14569/ijacsa.2023.0140370.

[22] R. Chen, X. Jin, S. J. Laima, Y. Huang, and H. Li, "Intelligent modeling of nonlinear dynamical systems by machine learning," *Int. J. Non. Linear. Mech.*, vol. 142, p. 103984, 2022, doi: 10.1016/j.ijnonlinmec.2022.103984.

[23] M. F. Abdillah and K. Kusnawi, "Comparative Analysis of Long Short-Term Memory Architecture for Text Classification," *Ilk. J. Ilm.*, 2023, doi: 10.33096/ilkom.v15i3.1906.455-464.

[24] I. K. Raharjana, F. Harris, and A. Justitia, "Tool for Generating Behavior-Driven Development Test-Cases," *J. Inf. Syst. Eng. Bus. Intell.*, vol. 6, no. 1, p. 27, 2020, doi: 10.20473/jisebi.6.1.27-36.

[25] S. Regundwar, R. Bhagwat, S. Bhosale, R. Chougale, and S. Abbu, "INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Sequence-to-Sequence Abstractive Text Summarization Model for Headline Generation with Attention," vol. 12, no. 3, pp. 842–851, 2024.

[26] G. Karuna, M. Akshith, P. S. Dinesh, B. V. Vardhan, Y. S. Bisht, and M. N. Narsaiah, "Automated Abstractive Text Summarization using Deep Learning," *E3S Web Conf.*, vol. 430, 2023, doi: 10.1051/e3sconf/202343001021.