A Madurese–Indonesian machine translation system using an encoder-decoder model with an attention-based LSTM architecture

Mulaab[1*], *Firdaus* Solihin[1], and *Bihubbil* Choir Aidifta[1]

[1]Department of Informatics, University of Trunodjoyo Madura, Bangkalan, East Java, Indonesia

Abstract. Indonesia's linguistic diversity includes Madurese, widely used as a first language by the Madurese community. However, limited proficiency in Indonesian—the national language—hinders their communication with speakers from other regions. In this study, a Neural Machine Translation (NMT) system based on the encoder-decoder architecture is implemented, designed to preserve full sentence structure. As the foundational framework, Long Short-Term Memory (LSTM) networks are employed. To address the limitation of the conventional encoder-decoder model—namely, its tendency to lose information due to the decoder's reliance solely on the encoder's final hidden state—an Attention mechanism is integrated. This enables the decoder to dynamically attend to all encoder outputs during the decoding process. As a result, this approach improves alignment between source and target words and significantly enhances translation quality. Empirical results show that this Attention-enhanced model works best with longer sentences (3–37 words), achieving a BLEU score of 0.0767—even with a modest dataset of 3,734 sentence pairs. Conversely, when translating very short sentences (≤2 words), performance declines significantly: despite a much larger corpus of 28,247 pairs, the BLEU score drops to 0.0344. This suggests that sentence length critically impacts model effectiveness, and that Attention-based NMT is more suited to moderately complex input. These findings highlight the importance of corpus characteristics in low-resource language translation and offer insights for developing better Indonesian–Madurese language tools.

1 Introduction

Language serves as the fundamental medium used by humans to communicate and interact. Indonesia, as a vast archipelagic nation with extraordinary geographic, social, ethnic, and cultural diversity, possesses a large number of regional languages[1–3]. One of these is the Madurese language—primarily spoken by communities in East Java, particularly on Madura Island. For the majority of its speakers, Madurese functions as their first language and primary means of daily communication. In contrast, proficiency in Indonesian—the national official language—is often limited to certain segments of society, especially those with formal educational backgrounds. This linguistic disparity can pose significant communication challenges, particularly for non-Madurese speakers who do not understand the language[4]. To address this issue, an efficient tool is needed to support cross-lingual understanding of words and sentences. Such a solution is commonly known as a machine translator[5, 6]. A machine translation system is an automated tool capable of converting text from one language to another without requiring manual word lookup in a dictionary. In computer science, this capability falls under the field of Natural Language Processing (NLP), specifically within the subdomain of Machine Translation (MT) [7][8].

Over the years, various MT approaches have been developed, including Rule-based Machine Translation (RBMT)[9], Statistical Machine Translation (SMT)[10], Phrase-based Machine Translation (PBMT)[11], Phrase-based Statistical Machine Translation (PBSMT)[12], and, most recently, Neural Machine Translation (NMT)[13]. Among these methods, NMT stands out as the most advanced and effective approach, producing translations that are not only fluent but also contextually accurate—building on the strengths while overcoming the limitations of earlier models.

NMT systems are typically built upon a sequence-to-sequence (Seq2Seq) encoder-decoder framework. In this architecture, the encoder processes the input word sequence in the source language and converts it into a fixed-size vector representation. The decoder then uses this representation to generate the corresponding output sequence in the target language, usually employing Recurrent Neural Networks (RNNs)[14]. However, standard RNNs struggle with handling long sequences due to their inability to retain information over very long contexts. To bridge this gap, more advanced RNN variants—namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)—were introduced,

* Corresponding author: mulaab@trunojoyo.ac.id

offering improved capacity for long-range dependencies[15].

Both LSTM and GRU incorporate gating mechanisms that regulate the flow of information at each time step, thereby enhancing their ability to capture and utilize long-range dependencies. The main difference between the two lies in structural complexity: GRU simplifies the LSTM design by eliminating the separate memory cell and using only two gates instead of the three gates in LSTM. As a result, GRU models are generally faster to train and execute. However, numerous empirical studies across various NLP tasks—including translation—often show that LSTM yields higher translation accuracy compared to GRU.

Despite these improvements, traditional RNN-based encoder-decoder models (including those using LSTM or GRU) still suffer from a fundamental limitation: the decoder relies solely on the final hidden state of the encoder as a summary of the entire input. In long source sentences, this can lead to the loss or distortion of important information from earlier in the input. The introduction of the Attention mechanism addresses this issue by enabling the decoder to dynamically assign weights to each encoder hidden state based on its relevance to the target word being generated[16]. This allows the model to more effectively preserve and utilize information from the entire input sequence[17].

Most contemporary NMT research depends on parallel corpora containing complete sentences of at least four words. In dictionary-based translation—such as for single words, short phrases, or brief sentences—this assumption fails[18]. The research contribution applies an Attention-based NMT model with an LSTM architecture for a Madurese–Indonesian translation system, focusing on its performance with very short linguistic units (sequences of fewer than four words). The MadureseSet dataset, a digital bilingual dictionary of Madurese–Indonesian word and phrase pairs, is used. The model employs Additive Attention, chosen for its strength in modeling sequential dependencies in translation

2 Proposed methods

Bridge communication gaps between Madurese and Indonesian speakers—especially where human translators are scarce—a reliable, context-aware automatic translation system is vital. Due to the lack of Madurese linguistic resources like large parallel corpora and standardized language tools, traditional data-driven methods often fall short. This study proposes a Neural Machine Translation (NMT) method built on an LSTM encoder-decoder architecture with an Attention mechanism.

This method addresses Madurese's unique traits: rich morphology, dialectal diversity, and limited data. The system uses the MadureseSet bilingual dataset—a digital dictionary of Madurese–Indonesian word and short phrase pairs—for training. Unlike most NMT systems that use longer sentences, this research focuses on very short units (under four words), reflecting dictionary-style translation used daily.

This methodology is designed to address the unique features of the Madurese language: rich morphology, dialectal variation, and limited training data. The system uses the MadureseSet bilingual dataset, a digital dictionary of Madurese–Indonesian word and short phrase pairs, as its training foundation. While most modern NMT systems use long sentences, this research targets very short units (under four words), reflecting the dominance of dictionary-style translation patterns in daily use.

The methodological process comprises several key stages as in figure 1:

- Data preprocessing, which involves standardizing text (normalization), addressing differences in Madurese dialects, and removing irrelevant or incorrect data (cleaning);
- Adaptive tokenization that accounts for Madurese morphology;
- Construction of word vector representations, also known as word embeddings, which are numeric representations of words capturing their meanings and relationships;
- Training of the neural machine translation (NMT) model using a Long Short-Term Memory (LSTM) neural network architecture, which is enhanced with Additive Attention—a mechanism that helps the model focus on relevant parts of the input—to improve translation accuracy for short phrases;
- Performance evaluation utilizing automatic metrics—such as BLEU (Bilingual Evaluation Understudy), which compares machine output to reference translations, and METEOR, which measures alignment of words and phrases—as well as qualitative assessment by native speakers.

By combining modern neural models with local adaptations, this approach aims to develop a Madurese–Indonesian translation system that is accurate, practical, and accessible. This effort supports the preservation of regional languages in the digital era. Due to its smaller speaker base, Madurese has not been a primary focus of large-scale NLP technology

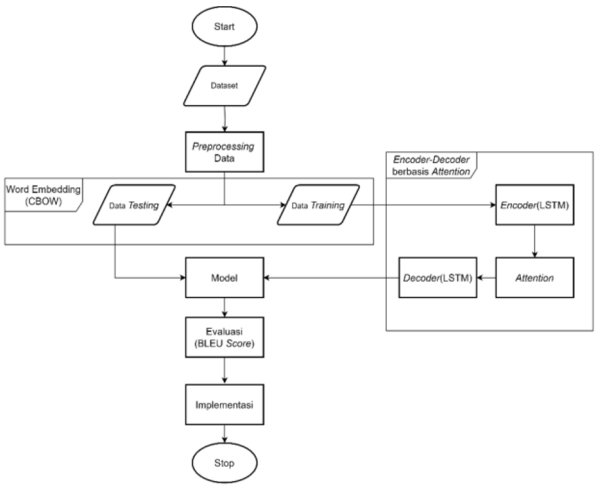

Fig. 1. Proposed methods workflow.

2.1 Dataset

In this study, the dataset is a combination of Madurese–Indonesian parallel corpora from these sources: Tata Bahasa Bahasa Madura with 316 sentence pairs (1–37 words)[19]; Pedoman Umum Ejaan Bahasa Madura Yang Disempurnakan with 74 pairs (3–11 words)[20]; Morfologi dan Sintaksis Bahasa Madura with 618 pairs (2–25 words)[21]; and the Madurese–Indonesian digital dictionary (MadureseSet) with 36,086 pairs (1–31 words)[22].

After compiling data from these four sources, a total of 37,094 parallel sentence pairs was obtained. During this compilation, overlapping or duplicate sentence pairs were also identified. Specifically, two duplicates were found in Morfologi dan Sintaksis Bahasa Madura , and 162 duplicates were identified in the MadureseSet digital dictionary. No overlapping pairs were found between Tata Bahasa Bahasa Madura and Pedoman Umum Ejaan Bahasa Madura Yang Disempurnakan.

2.2 Preprocessing

After the dataset collection phase, the next step is data preprocessing. This process is carried out to remove any remaining "noise" in the collected data. In this study, preprocessing consists of four stages: case folding, punctuation removal, tokenization, and finally, padding addition.

2.2.1 Case folding

At this stage, the data previously obtained during the dataset collection phase is examined string by string to check for the presence of uppercase letters. If uppercase letters are found, all such characters are converted to their lowercase equivalents. The results of the case folding process applied to the collected text data are presented in Table 1.

Table 1. Case folding results.

Madura	Indonesia
jhāhāl jhuwālaghi tang bhārāng	coba jualkan barang saya
ḍanté' aghi bulā	nantikan saya
cē' agherrā sē dāpa'a	sangat ingin untuk segera sampai

2.2.2 Punctuation removal

In the next step, punctuation marks—including special characters and digits—are removed from the data that has already undergone case folding. The results of this step are presented in Table 2

Table 2. Punctual removal results.

Punctual Removal	
Madurese	**Indonesia**
jhāhāl jhuwālaghi tang bhārāng	coba jualkan barang saya
ḍanté' aghi bulā	nantikan saya
cē' agherrā sē dāpa'a	sangat ingin untuk segera sampai

2.2.3 Tokenization

The third stage, or tokenization, involves splitting the sentences from the dictionary data, which has already undergone punctuation removal, into individual words[23]. The results of this tokenization step are shown in Table 3

Table 3. Tokenization results.

Madura	Indonesia
"jhāhāl", "jhuwālaghi", "tang", "bhārāng"	"coba", "jualkan", "barang", "saya"
"ḍanté", "aghi", "bulā"	"nantikan", "saya"
"cē", "agherrā", "sē", "dāpa'a"	"sangat", "ingin", "untuk", "segera", "sampai"

2.2.4 Adding <START> and <END> tokens to the target language corpus

The addition of <START> and <END> tokens serves to inform the model when a sentence begins and ends. The results can be seen in the following Table 4.

Table 4. Adding <START> and <END> tokens.

Madura	Indonesia
jhāhāl, "jhuwālaghi", "tang", "bhārāng"	<START>, "coba", "jualkan", "barang", "saya", "<END>"
ḍanté, "aghi", "bulā"	<START>, "nantikan", "saya", "<END>"
cē, "agherrā", "sē", "dāpa'a"	<START>, "sangat", "ingin", "untuk", "segera", "sampai", "<END>"

The target language, with the addition of both tokens, is used only as input to the decoder, whereas for the decoder's target output, the target language corpus receives only one additional token: <END> at the end of each sentence.

2.2.5 Add padding

Adding padding is a common technique used to equalize the length of sequences. This technique involves appending special padding elements to shorter sequences until they match the length of the longest sequence in the batch. In this word translation task, padding is applied to align the sentence lengths between the source (Madurese) and target (Indonesian) languages. The results of the add padding step are shown in Table 5.

2.2.6 Text-to-sequence

After equalizing sentence lengths between Indonesian and Madurese using the add padding process, the final step of data preprocessing—the Text-to-Sequence step—was performed. This step translates each word in every sentence into a numerical representation based on the index of unique words in their respective dictionaries. For example, the Text-to-Sequence step

applied to the Madurese sentence “jhājhal jhuwālaghi tang bhārāng” produces the results shown in Table 6.

Table 5. Add padding results.

Madura	Indonesia
"""jhāhāl”, “jhuwālaghi”, “tang”, “bhārāng”, “<PAD>”, “<PAD>”, “<PAD>”	"<START>", "coba", "jualkan", "barang", "saya", "<END>", "<PAD>"
“ḍantè”, “aghi”, “bulâ”, “<PAD>”, “<PAD>”, “<PAD>”, “<PAD>	"<START>", "nantikan", "saya", "<END>", "<PAD>", "<PAD>", "<PAD>"
“cè”, “agherrā”, “sè”, “dápa’a”, “<PAD>”, “<PAD>”, “<PAD>”	"<START>", "sangat", "ingin", "untuk", "segera", "sampai", "<END>"

Table 6. Text-to-sequence output.

Indeks	jhâjhâl: 0 jhuwâlaghi: 1 tang: 2 bhârâng: 3 dântè: 4
Hasil Teks to Sequence (“jhâjhâl”, “jhuwâlaghi”, “tang”, “bhârâng”)	0,1,2,3

2.2.7 Word embedding (CBOW)

At this stage, the data is split into two parts: training data and testing data. The purpose of this data splitting is to allow the researcher to evaluate the developed machine translation model. For the data split ratio between training and testing sets, the author employs a 90:10 proportion using K-Fold Cross-Validation with K=10. After the data split, the next step is to convert the numerical word representations into word vectors that capture semantic relationships (i.e., proximity or similarity) between words within sentences. In this study, the author uses the Continuous Bag-of-Words (CBOW) method [24].

2.3 Attention-based long short-term memory

Next, after all training data have undergone the CBOW process, the subsequent step is to process the data into sequential form using an encoder-decoder architecture with an Attention-based LSTM model[25]. For a more detailed overview of the specific architecture employed in this research, refer to Figure 2 and attention weight refer to Figure 3.

a. Encoder

Based on Figure 2, the Encoder-Decoder Architecture with Attention-based LSTM model, the first step is to process the data in the encoder. In this study, the author utilizes the LSTM model described in Figure 2, specifically the Long Short-Term Memory (LSTM) model, where the input data consists of sentence embeddings from the Madurese or Indonesian.

b. Attention

After the data is processed in the encoder stage, the next step is to apply the Attention Mechanism to distribute each encoder hidden state as input to each target time-step decoder. With its implementation, the input hidden state for each decoder time step no longer uses only the hidden state from the previous single time step (h-1), but also employs a new hidden state resulting from the distribution of all encoder hidden states toward the target decoder time step.

c. Decoder

The data previously processed by the encoder is subsequently used as a reference for the decoder process. This decoder process functions to translate the data from the encoder; thus, the first input word is a special “start” symbol, followed by sequences of words derived from the encoder input. The initial state input for this decoder model in the LSTM model is the final cell state from the encoder. Additionally, since this model incorporates attention into its encoder-decoder architecture, the input hidden state in the LSTM model includes an additional context vector, namely the attention context.

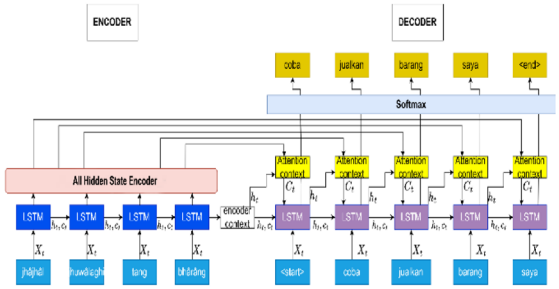

Fig. 2 Attention-based long short-term memory.

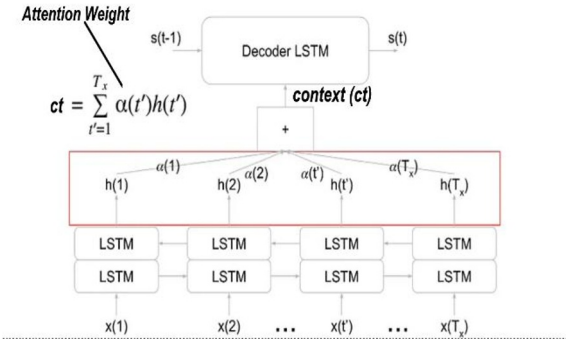

Fig. 3. Attention weight.

- s_{t-1}: hidden state of the decoder at time step $t-1$
  $h_{t'}$: hidden state of the encoder at time step t'
- $[s_{t-1}, h_{t'}]$: concatenation of the decoder’s previous state and the encoder’s hidden state at position t'.
- NeuralNet$(\cdot)$:a small feed-forward network that computes an unnormalized attention score.
- $\alpha_{t'}$: attention weight for encoder state $h_{t'}$ when generating the current decoder output.
- $\alpha_{t'} = \text{NeuralNet}([s_{t-1}, h_{t'}]), \quad t' = 1, \ldots, T_x$

These raw scores $(\alpha_{t'})$ are passed through a softmax function to ensure they sum to 1 and represent a probability distribution over the input sequence positions:

$$\alpha_{t'} = \frac{exp(score(s_{t-1}, h_{t'}))}{\sum_{k=1}^{T_x} exp\,(score(s_{t-1}, h_k))} \quad (1)$$

Context Vector Computation (ct) is calculate as

$$ct = \sum_{t'=1}^{T_x} \alpha_{t'}\, h_{t'} \quad (2)$$

This is a weighted sum of all encoder hidden states $h_{t'}$, where weights are the attention scores $\alpha_{t'}$.

2.4 Model evaluation

To assess the quality of the translation model, this study employs the Bilingual Evaluation Understudy (BLEU) metric. BLEU evaluates model accuracy by computing a score that reflects how closely the machine-generated translation matches one or more human reference translations[26].

The general formula for the BLEU score is defined as

$$\text{BLEU} = BP \cdot exp\left(\sum_{n=1}^{N} w_n\, logp_n\right) \quad (3)$$

or equivalently, using the geometric mean form

$$\text{BLEU} = BP \cdot \left(\prod_{n=1}^{N} p_n^{w_n}\right) \quad (4)$$

- p_n: modified n-gram precision for n-grams of order n
- w_n: weight assigned to each n-gram order $w_n = \frac{1}{N}$; for example in BLEU-4, $w_1 = w_2 = w_3 = w_4 = 0.25$).
- N: number of n-gram orders used.
- BP: **Brevity Penalty**, Brevity Penalty, which penalizes translations that are too short compared to the reference translation(s).

Brevity Penalty (BP) is calculated as:

$$BP = \begin{cases} 1 & if\ c > r \\ exp(1 - \frac{r}{c}) & if\ c \le r \end{cases} \quad (5)$$

Where

- c: the cumulative total length of the candidate translation (in number of tokens)
- r: the effective length of the closest reference translation (i.e., the number of tokens in the reference sentence whose length is closest to that of the candidate)

BLEU ranges from 0 to 1, although in practice the score is often reported on a 0–100 scale.

3 Experimental results

In this study, experimental scenarios were conducted involving the configuration of the dataset, preprocessing steps, and parameters used to train the Madurese–Indonesian attention-based LSTM machine translation model. Regarding the dataset, this research employs three dataset variations as follows:

- Dataset resulting from selection, containing only corpus with a maximum length of 2 words based on the Madurese language corpus; this dataset consists of 28,247 corpus pairs (Scenario 1).
- Dataset resulting from selection, containing only corpus with a minimum length of 3 words based on the Madurese language corpus; this dataset consists of 3,734 corpus pairs (Scenario 2).
- Original dataset (without corpus length filtering), which contains 31,981 corpus pairs (Scenario 3).

For further details regarding dataset settings and model parameters, refer to the following Table 7:

Table 7. Parameter model.

Parameter	Value
Model Dataset	Corpus with a maximum length of 2 words; Corpus with a minimum length of 3 words; Original dataset (without corpus length filtering)
Metode Word Embedding	CBOW
Learning Rate	0.002
CBOW Windows Size	5, 2
Split Data (train:test)	90:10 (K-Fold Cross Validation, K=10)
Batch Size	80
Epoch	56 and 100
N-Gram Bleu Score	1, 2, 3, and 4
Optimizer	Adam

In Scenario 1, several hyperparameter combinations are tested on the filtered dataset containing a corpus with a maximum length of 2 words (based on the Madurese language corpus), aiming to identify the optimal hyperparameter configuration for this dataset based on its BLEU score. The hyperparameter combinations tested in this scenario include: [Window size=5, Epochs=56]; [Window size=2, Epochs=56]; [Window size=2, Epochs=100]; and [Window size=5, Epochs=100]. The test results are presented in Figure 4.

Meanwhile, with a larger window size, the BLEU score improved slightly when more epochs were used; however, even then, the resulting score remained lower than that obtained using the combination with smaller parameter values, despite nearly doubling the number of epochs. This suggests that for datasets containing sentences with a maximum length of 2 words, a smaller embedding window size is more suitable.

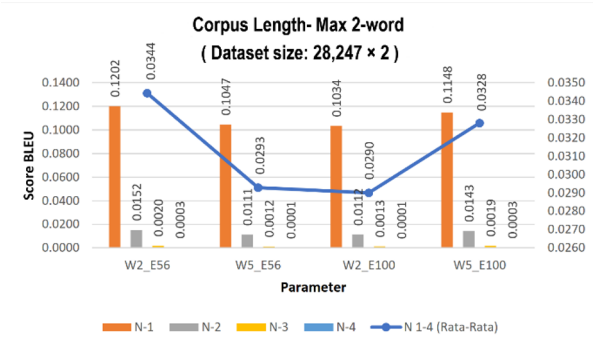

Fig. 4. Test results of scenario 1 (Corpus with a maximum length of 2 words).

For more detailed data regarding the average BLEU scores across each fold, refer to Table 8.

Table 8. Average BLEU scores per fold (Scenario 1).

Fold/	W2_ E56	W5_ E56	W2_ E100	W5_ E100
1	0.0234	0.0231	0.0223	0.0446
2	0.0371	0.0269	0.0226	0.0314
3	0.0411	0.0296	0.0219	0.0426
4	0.0441	0.0280	0.0407	0.0299
5	0.0376	0.0261	0.0302	0.0250
6	0.0293	0.0296	0.0321	0.0300
7	0.0418	0.0310	0.0264	0.0441
8	0.0240	0.0330	0.0219	0.0300
9	0.0241	0.0392	0.0414	0.0261
10	0.0417	0.0265	0.0305	0.0244
Average	0.0344	0.0293	0.0290	0.0328

In Scenario 2, the same hyperparameter combinations as in the previous scenario are tested: [Window size=5, Epochs=56]; [Window size=2, Epochs=56]; [Window size=2, Epochs=100]; and [Window size=5, Epochs=100], aiming to identify the most suitable hyperparameter configuration for this dataset — specifically, the filtered dataset containing corpus with a minimum length of 3 words (based on the Madurese language corpus). Scenario 2 also serves as a comparison to Scenario 1, allowing for an evaluation of the effectiveness of corpus length on the resulting BLEU scores. For more detailed test results, refer to Figure 5.

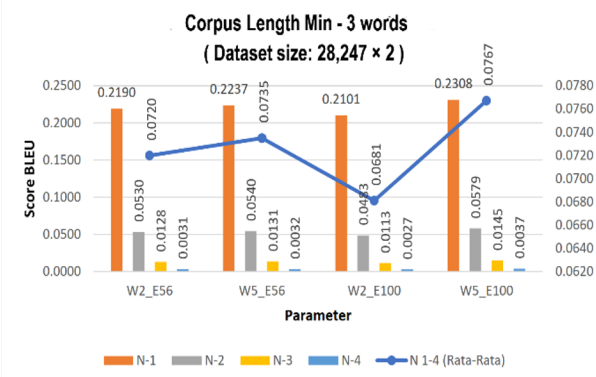

Fig. 5. Test results of scenario 2 (Corpus with minimum corpus length min 3-word.

Furthermore, when compared to Scenario 1, which used only a corpus with a maximum length of 2 words, the model trained on longer sentences (a minimum of three words) achieved a significantly higher average BLEU score, more than doubling its performance (from 0.0344 to 0.0767). For more detailed data on the average BLEU scores across each fold, refer to Table 9.

Table 9. Average BLEU scores per fold (Scenario 2).

Fold/	W2_ E56	W5_ E56	W2_ E100	W5_ E100
1	0.0686	0.0771	0.0795	0.0793
2	0.0721	0.0696	0.0543	0.0756
3	0.0667	0.0675	0.0652	0.0751
4	0.0767	0.0806	0.0714	0.0828
5	0.0672	0.0686	0.0661	0.0721
6	0.0629	0.0763	0.0707	0.0673
7	0.0747	0.0807	0.0722	0.0767
8	0.0835	0.0698	0.075	0.0826
9	0.0741	0.0700	0.0494	0.0799
10	0.0734	0.0747	0.077	0.0759
Average	0.072	0.0735	0.0681	0.0767

Just like the two previous scenarios, this scenario also nvolves testing the same hyperparameter combinations:

- [Window size = 5, Epochs = 56]
- [Window size = 2, Epochs = 56]
- [Window size = 2, Epochs = 100]
- [Window size = 5, Epochs = 100]

These trials aim to identify the most suitable hyperparameter configuration for the model using the original dataset—that is, without any sentence-length filtering. Additionally, Scenario 3 is designed to evaluate how the model performs when trained on a dataset that combines the data used in Scenario 1 and Scenario 2. The results of these experiments are presented in Figure 6.

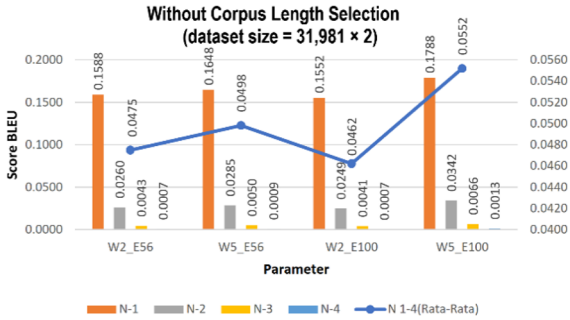

Fig 6. Scenario 3 trial results without corpus length filtering.

The best BLEU score was achieved with the hyperparameter setting [Window size = 5, Epochs = 100], indicating that the model benefits from a larger context window—a pattern also seen in Scenario 2. However, when the datasets from Scenario 1 and Scenario 2 were combined, model performance worsened. This deterioration is reflected not only in lower BLEU scores but also in the increased dataset volume and the higher number of training epochs required to achieve optimal results. Consequently, the training process becomes more computationally intensive and time-consuming, leading to reduced efficiency. Detailed explanation of the experiment see Table 10.

Table 10. Average BLEU scores per fold (Scenario 3).

Fold/	W2_ E56	W5_ E56	W2_ E100	W5_ E100
1	0.0464	0.0464	0.0573	0.0614
2	0.0474	0.0458	0.0391	0.0496
3	0.0531	0.0556	0.0439	0.0541
4	0.0446	0.0459	0.0457	0.0572
5	0.0447	0.0451	0.0517	0.0555
6	0.0437	0.0547	0.0421	0.0525
7	0.0434	0.0462	0.0354	0.0564
8	0.0594	0.0456	0.059	0.0548
9	0.0455	0.0549	0.045	0.0567
10	0.0465	0.0578	0.0429	0.0535
Average	0.0475	0.0498	0.0462	**0.0552**

4 Discussion

To compare performance and evaluate the effectiveness of the Attention-based LSTM Madurese–Indonesian machine translation model according to corpus length, the author uses the best results from Scenario 1 (maximum sentence length of 2 words), Scenario 2 (minimum sentence length of 3 words), and Scenario 3 (original dataset without sentence-length filtering). Further details of this comparison can be seen in Figure 7.

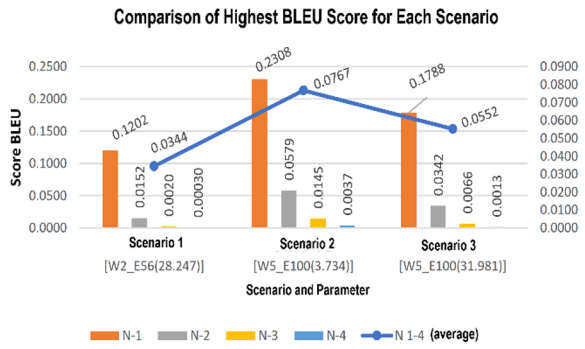

Fig 7. Comparison of the highest BLEU score for each scenario.

Based on Figure 7, comparison of Highest BLEU Scores Across Scenarios, the BLEU scores rank from highest to lowest as follows: Scenario 2, Scenario 3, and then Scenario 1.This clearly indicates that Scenario 2 demonstrates superior performance compared to the other two scenarios.

In addition to achieving the highest BLEU score, Scenario 2 also uses the smallest dataset—approximately 7.7 times smaller than Scenario 1 and 8.8 times smaller than Scenario 3. Despite requiring more training epochs than Scenario 1 (100 vs. 56), Scenario 2 still delivers the highest computational efficiency due to its significantly reduced data volume. Furthermore, the results suggest that using a corpus limited to a maximum sentence length of 2 words is ineffective for the Attention-based LSTM Madurese–Indonesian machine translation model. Not only does this configuration yield the lowest BLEU score, but it also degrades overall performance when combined with longer sentences (i.e., those with a minimum length of 3 words), as observed in Scenario 3.

5 Conclusion

This study uses a Madurese–Indonesian neural machine translation (NMT) system based on an LSTM encoder-decoder architecture with an Attention mechanism—an attempt to address linguistic issues in the diversity of regional languages, particularly Madurese. Empirical results show that the model achieves the best performance (BLEU score = 0.0767) on medium-length sentences (3–37 words), despite using a limited parallel corpus (3,734 sentence pairs). Interestingly, performance declines for very short sentences (≤2 words), even when the model is trained on a much larger subcorpus (28,247 pairs), with a BLEU score of only 0.0344. These findings indicate that sentence length distribution—not just corpus size—has a significant influence on the effectiveness of NMT under resource-constrained conditions. Limitations of this study are the small corpus size and imbalance in terms of sentence length distribution and domain coverage. The tokenization process and the morphological complexity of Madurese have not been fully addressed, potentially affecting alignment quality. Theoretically, this study provides empirical evidence that the Attention mechanism behaves differently across a range of sentence lengths under low-resource conditions—challenging the common assumption that increasing corpus size automatically improves performance. These findings provide a theoretical framework for NMT by introducing structural data characteristics, namely sentence length distribution, as an important predictor of model behavior, particularly for Madurese, a language with limited resources. Furthermore, this study presents a Madurese–Indonesian NMT system as a benchmark and basic framework for future language technology development for Madurese. This study provides general recommendations, namely expanding and balancing parallel corpora, especially for very short and very long sentences. Future research should explore subword-based tokenization (e.g., Byte Pair Encoding/BPE) and morphological normalization to address the agglutinative complexity of Madurese, and develop a transformer-based architecture to compare its robustness to sentence length variations.

References

1. M. R. Abtahian, A. C. Cohn, and Yanti., Language labeling and ideology in Indonesia. International Journal of the Sociology of Language. 147–171 (2022)
2. D. Wahyuni, Y. Maulina, A. Mulia, and S. Sunardi., Cultural Discourse in Reading Texts of Indonesian Language Proficiency Test. International Journal of Language Education. (2021). https://doi.org/10.26858/ijole.v5i4.23590
3. B. Arafah, Kaharuddin, M. Hasjim, A. Arafah, Takwa, and Karimuddin., Cultural Relations Among Speakers of South Halmahera Languages. Theory and Practice in Language Studies. (2022). https://doi.org/10.17507/tpls.1301.19

4. L. Puspandari and Y. Basthomi., The influence of Madurese dialect toward students' English word stress production. Indonesian Journal of Applied Linguistics. (2022). https://doi.org/10.17509/ijal.v12i1.46896
5. Y. Shen, W. Bao, G. Gao, M. Zhou, and X. Zhao., Unsupervised multilingual machine translation with pretrained cross-lingual encoders. Knowl. Based Syst. **284**, 111304 (2023). https://doi.org/10.1016/j.knosys.2023.111304
6. E. Steigerwald et al., Overcoming Language Barriers in Academia: Machine Translation Tools and a Vision for a Multilingual Future. Bioscience. **72**, 988–998 (2022). https://doi.org/10.1093/biosci/biac062
7. F. Stahlberg., Neural Machine Translation: A Review. J. Artif. Intell. Res. **69**, 343–418 (2019). https://doi.org/10.1613/JAIR.1.12007
8. J. Hirschberg and C. Manning., Advances in natural language processing. Science. **349**, 261–266 (2015). https://doi.org/10.1126/science.aaa8685
9. V. Sánchez-Cartagena, J. A. Pérez-Ortiz, and F. Sánchez-Martínez., A generalised alignment template formalism and its application to the inference of shallow-transfer machine translation rules from scarce bilingual corpora. Comput. Speech Lang. **32**, 46–90 (2015). https://doi.org/10.1016/j.csl.2014.10.003
10. L. Macken and E. Lefever., Translational equivalence in Statistical Machine Translation or meaning as co-occurrence. Linguistica Antverpiensia, New Series – Themes in Translation Studies. (2021). https://doi.org/10.52034/lanstts.v7i.215
11. J. Ning and H. Ban., Design and Testing of Automatic Machine Translation System Based on Chinese-English Phrase Translation. Mob. Inf. Syst. **2021**, 3539155–3539155 (2021). https://doi.org/10.1155/2021/3539155
12. F. Och and H. Ney., The Alignment Template Approach to Statistical Machine Translation. Computational Linguistics. **30**, 417–449 (2004). https://doi.org/10.1162/0891201042544884
13. F. Stahlberg., Neural Machine Translation: A Review. J. Artif. Intell. Res. **69**, 343–418 (2019). https://doi.org/10.1613/JAIR.1.12007
14. Y. Yu, X. Si, C. Hu, and J.-X. Zhang., A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation. **31**, 1235–1270 (2019). https://doi.org/10.1162/neco_a_01199
15. K. ArunKumar, D. Kalaga, Ch. Mohan, S. Kumar, M. Kawaji, and T. Brenza., Comparative analysis of Gated Recurrent Units (GRU), long Short-Term memory (LSTM) cells, autoregressive Integrated moving average (ARIMA), seasonal autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends. Alexandria Engineering Journal. **61**, 7585–7603 (2022). https://doi.org/10.1016/j.aej.2022.01.011
16. G. Brauwers and F. Frasincar., A General Survey on Attention Mechanisms in Deep Learning. IEEE Transactions on Knowledge and Data Engineering. **35**, 3279–3298 (2023). https://doi.org/10.1109/TKDE.2021.3126456
17. E. Ahmadzadeh, H. Kim, O. Jeong, and I. Moon., A Novel Dynamic Attack on Classical Ciphers Using an Attention-Based LSTM Encoder-Decoder Model. IEEE Access. **9**, 60960–60970 (2021). https://doi.org/10.1109/ACCESS.2021.3074268
18. Y. Wan et al., Challenges of Neural Machine Translation for Short Texts. Computational Linguistics. **48**, 321–342 (2022). https://doi.org/10.1162/coli_a_00435
19. A. Mahmud., Tata bahasa bahasa Madura: edisi revisi. (2014).
20. B. Balai., Pedoman Umum Ejaan Bahasa Madura Yang Disempurnakan (Cetakan ke 2). Surabaya: Balai Bahasa Surabaya. 1–45 (2003).
21. M. Moehnilabib., Morfologi dan sintaksis bahasa Madura. (1979).
22. N. Ifada, F. H. Rachman, M. W. M. A. Syauqy, S. Wahyuni, and A. Pawitra., MadureseSet: Madurese-Indonesian Dataset. Data in Brief. **48**, 109035 (2023).
23. C. Toraman, E. Yilmaz, F. Şahinuç, and O. Ozcelik., Impact of Tokenization on Language Models: An Analysis for Turkish. ACM Transactions on Asian and Low-Resource Language Information Processing. **22**, 1–21 (2022). https://doi.org/10.1145/3578707
24. Z. Xiong, Q. Shen, Y. Xiong, Y. Wang, and W. Li., New Generation Model of Word Vector Representation Based on CBOW or Skip-Gram. Computers, Materials & Continua. (2019). https://doi.org/10.32604/CMC.2019.05155
25. J. Zeng, Xiao, and K. Zhou., Enhancing Attention-Based LSTM With Position Context for Aspect-Level Sentiment Classification. IEEE Access. **7**, 20462–20471 (2019). https://doi.org/10.1109/ACCESS.2019.2893806
26. B. Babych and A. Hartley., Automated error analysis for multiword expressions: Using BLEU-type scores for automatic discovery of potential translation errors. Linguistica Antverpiensia, New Series – Themes in Translation Studies. (2021). https://doi.org/10.52034/lanstts.v8i.246