Transformer Hyperparameter Tuning for Madurese-Indonesian Machine Translation

Fika Hastarita Rachman
Informatics Department, University of Trunojoyo Madura, Bangkalan, Indonesia
fika.rachman@trunojoyo.ac.id (corresponding author)

M. Wildan Mubarok Asy Syauqi
Informatics Department, University of Trunojoyo Madura, Bangkalan, Indonesia
willnode@wellosoft.net

Noor Ifada
Informatics Department, University of Trunojoyo Madura, Bangkalan, Indonesia
noor.ifada@trunojoyo.ac.i

Imamah
Department of Information Systems, University of Trunojoyo Madura, Bangkalan, Indonesia
i2m@trunojoyo.ac.id

Sri Wahyuni
Mechatronic Department, University of Trunojoyo Madura, Bangkalan, Indonesia
s.wahyuni@trunojoyo.ac.i

Received: 6 December 2024 | Revised: 28 January 2025 | Accepted: 5 March 2025

 DOI: https://doi.org/10.48084/etasr.9851

ABSTRACT

The main problem arising in using Neural Machine Translation (NMT) for the Madurese language is the limitation of training data due to the unavailability of an adequate parallel corpus. In addition, the model must overcome the difference in words caused by the level of politeness in the Madurese language (coarse, moderate, and smooth). The rules-based approach requires many rules to represent these differences. In contrast, the statistical approach relies on the frequency of words in the training data, which cannot accurately capture variations in politeness levels. To overcome this problem, a parallel corpus was created to provide adequate training data, and an embedding matrix based on Skip Gram with Negative Sampling (SGNS) was used to produce better word representations for processing with transformers. This study also employs two types of evaluation: model configuration based on dataset size (large and small) and two tokenization methods (word and subword levels). The best results were obtained with the large dataset using word-level tokenization, achieving 0.70% accuracy for entirely correct text, 78.87% for partially correct text, and a BLEU score ranging from 4.76 to 27.63 with a maximum *n*-gram value from 1 to 4. This approach improved translation accuracy and shows significant potential for developing NMT systems for languages with limited resources, such as the Madurese language.

Keywords-machine translation; neural machine translation; transformers; Madurese ; Indonesian; subword tokenization; word pieces

I. INTRODUCTION

Based on the 2010 census, local languages are widely used in the daily lives of Indonesians [1]. In 2023, East Java became the most popular area for domestic tourism [2]. East Java has three local languages: Java, Madurese, and Bajo [3]. Madurese is one of the four local languages with the most speakers and has many more word variations than Indonesian [4, 5]. Immigrant communities or tourists can positively affect local languages because they can make local communities aware of their cultural assets. However, they can also have a negative impact due to the large number of foreign terms used in daily life, leading to language erosion or shifts [5]. One of the efforts to preserve local languages is by developing translation machines [6]. AI-powered translation services can contribute to language revitalization to preserve cultural heritage [7].

Machine Translation (MT) is a computer-based application that can change text from one language to another automatically [8, 9], involving automated systems that generate translations either independently or with human support [10]. In general, MT was built by applying the concepts of Rule-Based Machine Translation (RBMT), Statistical Machine Translation (SMT) [11], and Neural Machine Translation (NMT) [12, 13]. The Madurese language has three levels of politeness: rude, moderate, and polite. RBMT and statistical approaches have been limited in addressing this challenge. RBMT is created based on syntactic patterns or rules defined in each language [14]. Rule-based approaches require many rules to represent these different politeness levels, which are complicated and time-consuming to create and difficult to update as the language and social context change. This makes the rule-based approach less flexible and inefficient because it cannot automatically learn these patterns from the data.

In statistical-based translation, the model analyzes pairs of sentences in the source and target languages, also known as a parallel corpus [15], to calculate the probability that a particular word or phrase will be translated into the target language. The model decides the most likely translation based on the frequency of word occurrence in the training corpus. In Madurese, politeness differences can depend heavily on who speaks and to whom. Statistical systems tend to select words based on the frequency of appearance in the training data without considering the appropriate politeness level. As a result, although the translated word occurs frequently in the data, it may not match the correct politeness level in the conversation context. Therefore, the most appropriate approach to address this problem is NMT [16], which uses a neural network-based method. Neural network methods have been used in speech recognition [17], text mining, and image and video recognition topics. NMT is an MT approach that uses a large neural network.

Neural networks have achieved incredible accuracy on complex speech and visual object recognition problems. NMT is used in research as a neural network-based MT method that can read a sentence, produce a correct translation, and be applied to different language topics. NMT has been used by Google Translate since 2016 after Google published research on NMT-based MT [8]. Currently, Google Translate can translate regional languages such as Javanese and Sundanese [4]. Madurese is a popular regional language but has never been studied as an object of NMT-based MT. Transformers are one of the NMT-based MT models, developed to improve translation with word dependencies regardless of sentence length [18, 19]. In [20], MT for Madurese-Indonesian was applied using RBMT. Madurese language research is difficult due to the low resources of Madurese language data. Thus, in previous research, we created the MadureseSet [21] as the main corpus for research in the field of Madurese-Indonesian MT.

The problem with implementing transformer-based MT is that the accuracy results vary depending on the available dataset, the configuration of the model used, and the type of tokenization of language objects [22]. One model configuration can use hyperparameters adjusted for many datasets in the millions of text pairs between the two languages [18]. Another model configuration can use hyperparameters adjusted for a small number of datasets in the tens of thousands of text pairs [22]. Tokenization is broadly divided into two types: tokenization per word and subword [23, 24]. Subword tokenization is useful for improving the translation accuracy of rarely used words [25].

Based on these problems, this study aimed to implement an MT system for Madurese-Indonesian using the NMT method. The proposed approach uses the translated text in MadureseSet [21] as a database to build a transformer-based MT model. The contributions of this study are as follows.

- Develops a parallel corpus for Madurese-Indonesian translation, extracted from the MadureseSet dataset.
- Constructs an embedding matrix that leverages the Skip-Gram with Negative Sampling (SGNS) model for enhanced representation.

The formation of the Madurese-Indonesian NMT considers the influence of the tokenization model (word tokenization and subword tokenization [8]) and the hyperparameters of the transformer models (small [22] and large configuration [18]).

II. TRANSFORMER MODELS IN NMT

Transformers are neural network-based models that consist of encoder and decoder components. Transformers receive input as tokens in the form of an embedding matrix. The embedding matrix enters the encoder process and outputs it as token probabilities that are repeated in the decoder process.

A. Encoder and Decoder

Figure 1 shows the architecture of the transformer-based model. The encoder and decoder layers are repeated *N* times, each process referred to as encoding and decoding. Figure 2 illustrates the iteration of the encoder and decoder layers. This loop aims to preserve the sentence context better. The encoder and decoder consist of several layers, as shown in Figure 3. Inside each encoder or decoder layer, there is a self-attention layer, followed by a Feed-Forward Network (FFN) layer. A layer of residual connections envelops the two layers.

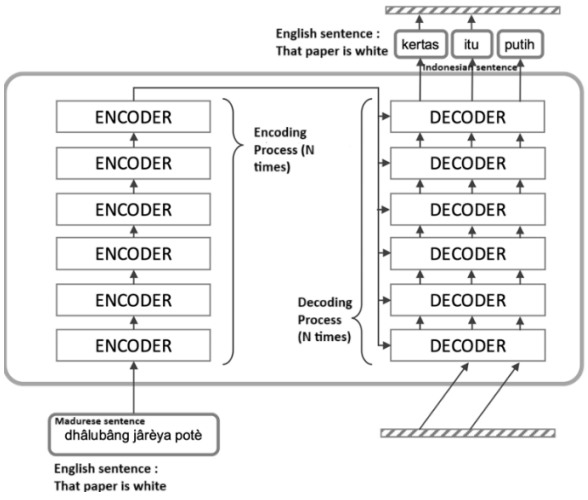

Fig. 1. Illustration of encoder and decoder.

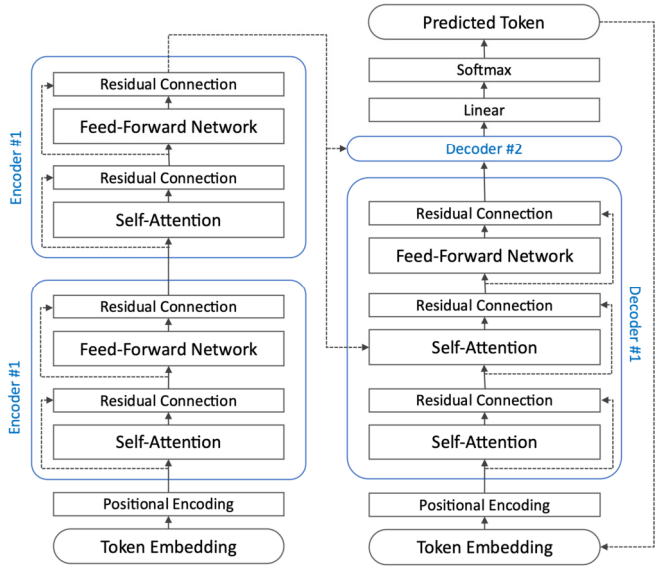

Fig. 2. Illustration of the layers inside the encoder and the decoder.

Once the embedding matrix has entered and passed through the encoding process, the final matrix is inserted into the second self-attention layer in each decoder layer. The decoding process is different from the encoding process, where the former needs to be repeated per token (word for word) until a sequence of tokens is obtained that forms the prediction of the translated sentence. The flow difference between the encoding and decoding processes is illustrated in Figure 2.

B. Self Attention

The self-attention mechanism works using matrix weights referred to as query, key, and value, notated as sequentially sized. When the input of the $W^Q W^K W^V d_{model} \times d_k d_{model} \times d_k d_{model} \times d_v$ embedding matrix is multiplied by each weight, resulting in a matrix of queries, keys, and values, notated as, in order of size, $QKVw \times d_k w \times d_k w \times d_v$. After obtaining the matrix, the attention value (QKV) is calculated using (1). The result of calculating the attention value is in a matrix form of $w \times d_v$.

$$attention\ (Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_{model}}}\right)V \quad (1)$$

where Q,K,V are the matrices *of* queries, keys, and values used, and d_{model} is the length of embedding dimensions. The following formula is used for the softmax function:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (2)$$

where $\sigma(\vec{z})_i$ is the softmax function for the dimension $\vec{z}_i$, and e is the Euler's constant.

The purpose of calculating the attention value is the weighting between certain and other words, referred to as features in the text. To achieve this weighting with many features, the concept of the multi-head attention function is used, which is the attention function that is repeated h times, and then combined with a weight matrix. This calculation is described in (3) and (4).

$$head_i = attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (3)$$

$$MultiHead(Q,K,V) = Concat(head_1, \dots\ head_h)\ W^O \quad (4)$$

C. Position-wise Feed-Forward Network (FFN)

After the addition of an attention sublayer, in each encoder and decoder layer, a position-wise fully connected FFN layer is added. This layer is applied to each embedding token or vector position. This layer consists of two linear transformations with ReLU activation, described by:

$$FFN(x) = (0, W_1 x + b_1)\ W_2\ + b_2 \quad (5)$$

where x is the input vector (separated by position from the input/embedding matrix), W_1 are the weight parameters for the first layer, dimensioned $d_{model} \times d_{ff}$, b_1 are the bias parameters for the first layer, dimensioned $1 \times d_{ff}$, W_2 are the weight parameters for the second layer, dimensioned $d_{ff} \times d_{model}$, and b_1 are the bias parameters for the second layer, dimensioned $1 \times d_{model}$.

D. Residual Connection

Each sublayer in the encoder and the decoder has an additional residual layer followed by a normalization layer. The residual layer is used to prevent data degradation caused by multiple layers by including the data values before and after the layer [26]. The residual layer is formulated in (6) and illustrated in Figure 4.

$$a^t = F(x) + x \quad (6)$$

where $F(x)$ is the output of a layer, x is the input value of a layer, and a^t is the residual finish of the layer. To make the range of values after the residual layer not to be too large due to the summation operation between two values, the values need to be normalized. The normalization process is formulated in (7), aided by the equation of the mean value and standard deviation in (8) and (9) [27].

$$h^t = \frac{g}{\sigma^t} \odot (a^t - \mu^t) + b \quad (7)$$

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \quad (8)$$

$$\sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)} \quad (9)$$

where a^t denotes the residual vector values before normalization, h^t are the residual vector values after normalization, μ^t is the average value of each element in a^t, σ^t is the standard deviation value for each element in the a^t, H is the length of the inner element a^t, b are adaptive parameters for bias vector values, and g are adaptive parameters for gain vector values.

This process requires bias and gain values, which are adaptive parameters (adjusted during training). Bias and gain are initialized with vectors with values of 0 and 1.

E. Decoder Process, Linear and Softmax Layer

The process at the decoding stage is different from encoding. If in the encoding process all the inputs in the text

are in the form of an embedding matrix, the input in the decoding process is done iteratively, predicting word for word until the translated sentence is completed. The way the decoder searches for words to start and end sentences is similar to the text embedding process in the encoder process. A special token that marks the Beginning Of a Sentence (BOS) is used and inserted into the input decoder from the far right of the token sequence. Then, at the end of the decoder, the next translation word prediction is determined based on the softmax layer containing the probability value, or what is called logits, and then added to the input of the decoder again until a token that marks the End Of the Sentence (EOS).

Another difference in the decoder's self-attention layer is that the layer should only be calculated by the position of the token that has been processed before. Therefore masking is required for tokens whose position has not been processed. Masking is performed by replacing the vector value before the softmax process in (1). In addition, there is also an additional self-attention layer that calculates the key vector and the value of the encoder's output vector at the very end.

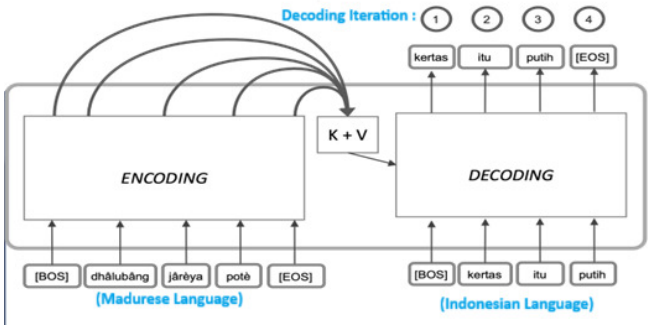

Fig. 3. Decoder process in transformers model.

At the end of the decoder process, the result obtained is a matrix value. To make the matrix value a token prediction $w \times d_{model}$, there is a linear transformation layer which is a neural network layer that has an output in the form of a sized matrix, where $w \times T$ is the number of tokens in the target language. Each row in the matrix projects a target probability called a log. Then all the probability values in the logits are input into the softmax function so that all the values add up to 1.0. The equations of linear and softmax functions are described in (10) and (2).

$$Linear(x) = xW + b \tag{10}$$

where x are the vector inputs (separated by position from the decoder output matrix), W are the weight dimensional parameters ($d_{model} \times T$ tokens), and b are the bias parameters, dimensioned $1 \times T$.

After the softmax process, the position of the logits with the highest value in the last row of the softmax result matrix is examined. This position determines which token is the prediction (word) after the previous predictions, and then, the decoding process is repeated until the prediction of the special token that marks the EOS is selected.

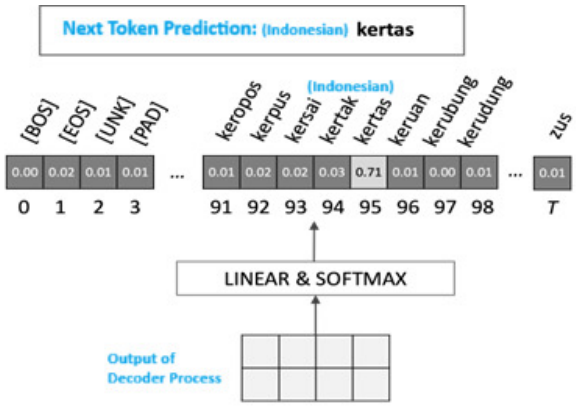

Fig. 4. Process of selecting a token prediction through the highest logits.

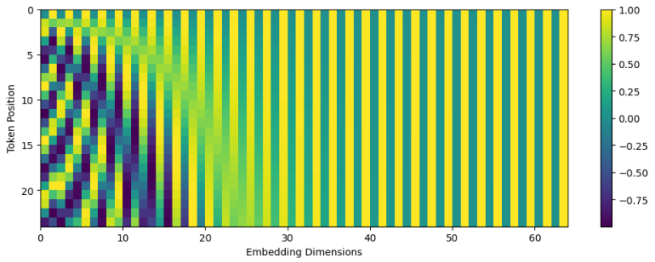

Fig. 5. Visualization of the positional encoding function

F. Positional Encoding

This transformer model has no iterative model process or convolution, so for the model to know the position of the token in the vector order, it needs to enter information about its relative or absolute position. In this process, the embedding vector input is added with a positional encoding vector that has the same dimensions as the embedding. The d_{model} value of the encoding position in the vector for the transformer model is formulated as follows [3]:

$$PE_{(pos,2i)} = sin\ sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{11}$$

$$PE_{(pos,2i+1)} = cos\ cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{12}$$

where $PE_{(pos,2i)}$ is the positional encoding for even positions (0, 2, 4, 6..), $PE_{(pos,2i+1)}$ is the positional encoding for odd positions (1, 3, 5, 7..), pos is the token position, i is the dimensional position in the embedding, and d_{model} is the number of embedding dimensions. As a result, the vector formed will be in the form of a sinusoidal that runs odd-even, as shown in Figure 7. For the value of the embedding vector not to be lost after being added with positional encoding, the vector will be multiplied before being added with the $\sqrt{d_{model}}$ positional encoding vector [3].

G. Dropout

Dropouts as added to regularize value changes. Dropout is a random probability distribution in which several connections

are deleted (set to 0) randomly according to a given probability. The goal is to avoid a connection from focusing too much on adjusting to the specifics of the dataset (overfitting) in the process of forming the model [17].

Dropouts are applied in two ways: First, processing the residual connection in each sublayer output before being added with the sublayer input and normalized, and second, processing after the addition between input embedding and positional encoding in both the encoder and decoder processes.

H. Label Smoothing

A smoothing label is added as a regularization of the process (second in addition to dropout). Label smoothing is the probability of flattening (or decreasing) the comparison of values in the logits after the linear and softmax functions, before determining the token candidate with the highest probability. The smoothing label is formulated as:

$$q'(x) = (1 - \varepsilon) \cdot q(x) + \frac{\varepsilon}{K} \quad (13)$$

where $q'(x)$ is the value of the probability in each vector element after label smoothing, $q(x)$ is the value of the probability in each vector element before label smoothing, ε is the parameter constant of the smoothing label (between 0 and 1), and K is the number of tokens or the length of the probability vector dimension.

If the equation has a fixed value, it produces the same probability vector for all elements. The purpose of label smoothing is that when calculating the loss function, the loss value is not too large, so that a logit does not tend to adjust to the same value (overconfident).

The loss function is applied using the Kullback-Leibler divergence loss (KL div loss) because it can apply a penalty if the prediction probability value differs from its uniform probability. The KL div loss per position equation is formulated in (14). The value of the loss in the formation of the model in each iteration is calculated by the sum of all $L(y_{pred}, y_{true})$.

$$L(y_{pred}, y_{true}) = y_{true} \cdot \left(log\ log\ y_{true} - log\ log\ y_{pred}\right) \quad (14)$$

where $L(y_{pred}, y_{true})$ is the loss value between the predicted and correct results, y_{true} is the correct probability value (0 or 1), and y_{pred} is the probability value of the prediction result (between 0 and 1).

III. SUBWORD TOKENIZATION

Subword tokenization is the separation of one token in the form of a word into several tokens consisting of several short words that form it. Subword tokenization aims to help detect long or rarely used words, thereby reducing the possibility of words that are not in the previous embedding data and providing a maximum limit on the number of tokens so that the embedding is not filled with many rare words [28].

Wordpiece is one subword tokenization method based on Byte Pair Encoding (BPE) adapted for NLP, which was originally used as a data compression method. Wordpiece is better than BPE because the latter is susceptible to differences in the way a word is separated, which affects the quality of the result. Wordpiece solves this problem by adding considerations about whether a text is better separated or not [8].

To process an unknown word, the word needs to be broken down into several subwords that are known from the training data from Wordpiece. Special boundary symbols are given before training so that the original word can be composed of subwords without any uncertainty [8]. An example of the word order in a sentence and the token output from Wordpiece follows:

Sentence: Jet makers feud over seat width with big orders at stake

Subword using Wordpiece: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

In this example, the word Jet is split into two words "_J" and "et", and the word "feud" is split into two words "_fe" and "ud". The other words remain as single words. "_" is a special character added to mark the beginning of a word [8].

IV. WORD EMBEDDING SKIP-GRAM WITH NEGATIVE SAMPLING (SGNS)

The transformer model requires input data in the form of a matrix (as a series of vectors), so a process is needed to change the series of tokens in an input text into a series of vectors. This additional process is called word embedding. The benefit of using word embedding is that the dimensionality of the resulting vector can be drastically reduced, making it easy to compute. SGNS is an advanced model of continuous skip-gram with some additional extensions to shorten the accuracy and training speed through negative sampling [29]. Negative sampling comes from the Noise Constrastive Estimation (NCE) method. Negative sampling works by giving a negative sample to each positive token. Since the goal of skip-gram is to print a high-quality embedding vector, negative sampling can be further simplified by simply taking a few tokens at random.

V. MATERIALS AND METHODS

A. Data

The training data consisted of 50,084 pairs of Madura-Indonesian texts taken from the Dictionary of Indonesian Madurese [21]. The data retrieved from this digital dictionary is a collection of text pairs (parallel corpus) between Madurese language texts and their translations in Indonesian. The test data amounted to 284 pairs of sentences taken from the book Madurese Language [30]. This book explains the morphology and syntax of Madurese and has quite a lot of examples of perfect sentences along with translations. Since the test data source is an old book, many text syntaxes have changed. Before used for testing, all the letters "q" were replaced with quotation marks to match the syntax used in the dictionary.

Example sentence pairs for testing:

Madurese: Jhâuèpon dâri kaqdinto tello polo kilo

Indonesian: Jauhnya dari sini tiga puluh kilo (English: It's thirty kilometers from here)

B. Design System

Figure 6 shows a flowchart used in this study. The data construction stage processed the data into structured data to be used in forming models. The model formation stage was used to form an MT model according to the respective configuration variations. The model evaluation stage aimed to determine the reliability of the system. Model testing was carried out using new data, different from the previous data extraction process.

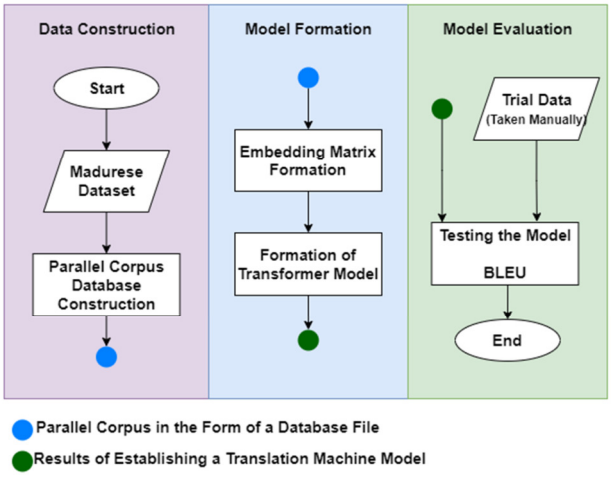

Fig. 6. Proposed NMT system for Madurese-Indonesian.

C. Data Construction

In the data construction stage, the parallel corpus [31] used was divided into two types, namely parallel corpus with per-word tokenization and parallel corpus with subword tokenization [32]. The latter was formed using the Wordpiece algorithm. The parallel corpus database with basic tokenization uses concepts tokenized per word, while the parallel corpus database with subword tokenization was formed using the Wordpiece algorithm [33].

D. Model Formation

The formation of the model was divided into two stages. First, an embedding model produces an embedding matrix that is used as input for the formation of the second model, a transformer-based model for MT. The formation of the transformer-based model was performed by entering parallel corpus data modified through the previously formed embedding matrix. The formation of the embedding matrix was carried out using the SGNS model. Table I shows some parameters recommended in research related to transformers.

TABLE I. MODEL CONFIGURATION FOR EMBEDDING MATRIX

Symbol	Description	Recommendation Value
T	Number of tokens	As per dataset
d_{model}	Embedded length	512
C	Tokens considered close together	10
k	Negative sampling per pair	20
t	Threshold dropout	10^{-5}

The transformer-based MT model has several hyperparameters that can affect the accuracy of the results depending on the data. Table II shows the hyperparameters used in this study. The MT model used a parallel corpus stored in a matrix form through the embedding stage. The model weights were optimized using Adam. Table III shows the testing scenarios.

TABLE II. TRANSFORMER MODEL CONFIGURATION

Symbol	Description	Recommended value
N	Number of encoders/decoders	6
h	Number of attention head	8
d_k	Vector key length	$d_{\text{model}}/h = 64$
d_v	Vector value length	$d_{\text{model}}/h = 64$
d_{ff}	FFN length	2048
P_{drop}	Droupout rate	0.1
ϵ_{ls}	Label smoothing	0.1

TABLE III. VARIATION CODE ON IMPLEMENTATION

Testing scenario	Description	Code
#1	Large dataset with per-word tokenization.	HIGH_BASE
#2	Large dataset with subword tokenization.	HIGH_WP8K
#3	Small dataset with per-word tokenization.	LOW_BASE
#4	Small dataset with subword tokenization.	LOW_WP8K

E. Model Evaluation

The evaluation was carried out using accuracy percentages and Bilingual Evaluation Understudy scores (BLEU) [34]. Accuracy is a simple comparison between the model results and the correct translation reference. This accuracy percentage classifies translation results into three categories: 1) Correct are sentence results between the two texts that are the same and identical. 2) Partially correct is that at least one word is the same between the two texts. 3) Incorrect refers to no words being the same between the two texts. The BLEU score is a number between 0 and 100. A BLEU score of 100 indicates that all predicted sentences are identical with the correct sentence references. Since not all correct translated sentences have to be identical, a good BLEU score, even with human translation evaluation, does not always have to be a score close to 100.

VI. RESULTS AND DISCUSSION

A. Parallel Corpus Database

Results on the parallel corpus databases are divided into two variations, namely the parallel corpus with tokenization per word and tokenization for subwords (Tables III and IV).

It should be noted that the characteristics of dictionary translation examples are mostly in the form of word fragments, so the average length of tokens in one text tends to be low. This can be confirmed through the distribution graph in Figure 7. The graph compares a text (horizontal) and the number of tokens in the text (vertical). The sloping distribution on the graph will affect the translation results later.

B. Matrix Embedding

Table VI shows the results of the embedding matrix formation process. The table lists the number of epochs (data iterations during the formation process) and losses (penalty indicators for wrong predictions) in each variation. A high number of epochs causes the loss figure to fall close to zero. The statistics of the parallel corpus database are divided into two variations: a parallel corpus with word-level and another with subword-level tokenizations.

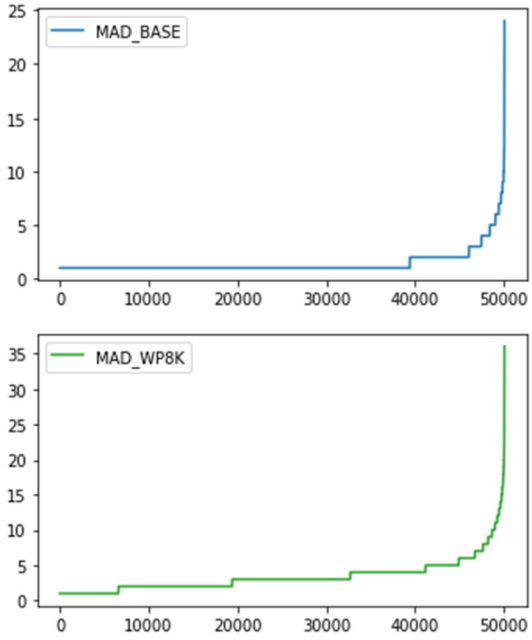

Fig. 7. Token length distribution for each text in Madurese.

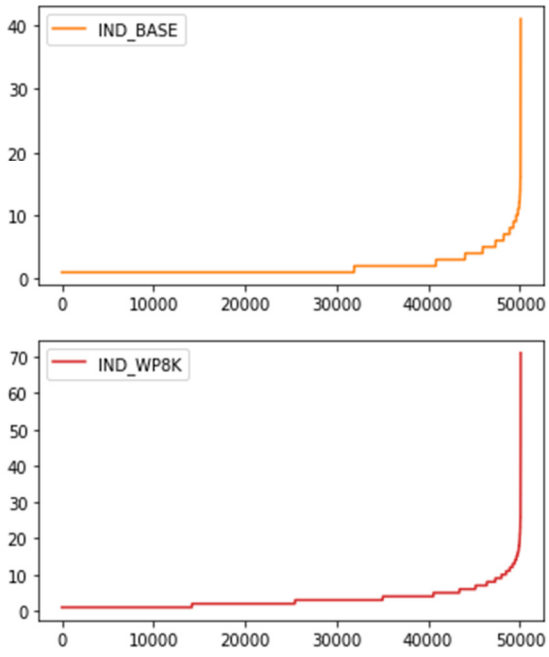

Fig. 8. Token length distribution of each text for Indonesian.

TABLE IV. STATISTICS OF PARALLEL CORPUS WITH WORD-LEVEL TOKENIZATION

Parameters	Count
Number of text pairs	50084
Number of word tokens (not including special tokens)	Madurese:30990 Indonesian: 14686
Maximum token length per text	Madurese:24 Indonesian: 41
Average token length per text	Madurese:1.42 Indonesian: 1.91
Median token length per text	Madurese:1 Indonesian: 1
Average amount of text using a given token	Madurese:2.31 Indonesian: 6.52

TABLE V. STATISTICS OF PARALLEL CORPUS WITH SUBWORD-LEVEL TOKENIZATION

Parameter	Count
Number of text pairs	50084
Number of word tokens (not including special tokens)	Madurese:8000 Indonesian: 8000
Maximum token length per text	Madurese:36 Indonesian: 71
Average token length per text	Madurese:3.37 Indonesian: 3.23
Median token length per text	Madurese:3 Indonesian: 2
Average amount of text using a given token	Madurese:22.57 Indonesian: 23.61

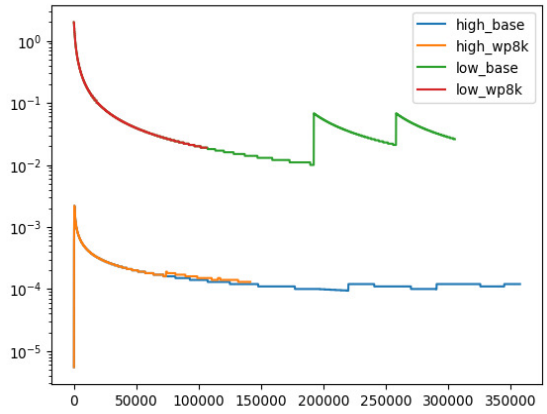

Fig. 9. Learning date iterations used during the training process.

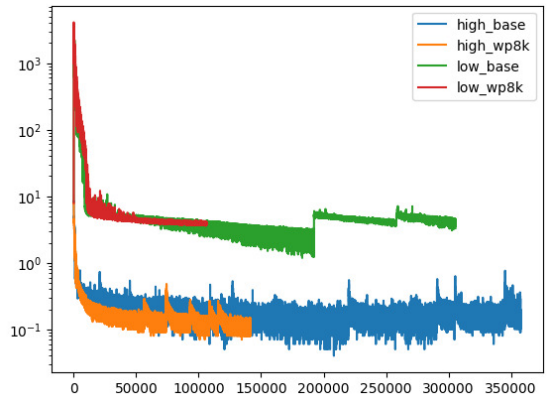

Fig. 10. Loss values generated during the training process.

C. Machine Translation (MT) Model

Tables VI and VI show the results of the embedding matrix and the MT model. It can be observed that by increasing the number of epochs, loss decreases.

TABLE VI. NUMBER OF EPOCHS AND FINAL LOSS FOR EMBEDDING MATRIX

Scenario	Code	Number of epochs	Final loss
#1	HIGH_BASE_MAD	100000	~0.08
	HIGH_BASE_IND	100000	~0.08
#2	HIGH_WP8K_MAD	10000	~0.80
	HIGH_WP8K_IND	10000	~0.80
#3	LOW_BASE_MAD	100000	~0.08
	LOW_BASE_ IND	100000	~0.08
#4	LOW_WP8K_MAD	10000	~0.80
	LOW_WP8K_IND	10000	~0.80

TABLE VII. NUMBER OF EPOCHS AND FINAL LOSS FOR MT MODEL

Scenario	Code	Number of epochs	Final loss
#1	HIGH_BASE	1000	~0.15
#2	HIGH_WP8K	480	~0.10
#3	LOW_BASE	1000	~4.30
#4	LOW_WP8K	420	~3.90

D. Evaluation

Tables VIII and IX show the evaluation results for accuracy percentage and BLEU score. For the BLEU score, to obtain a better picture of the score obtained, the maximum variation of *n*-grams used is also listed. Based on the test scenarios carried out, the best accuracy was achieved using the large dataset with tokenization per word, both in terms of accuracy and BLEU score. The results of this evaluation are illustrated again in Figures 10 and 11.

TABLE VIII. ACCURACY OF DIFFERENCES VARIANTS

MT model variants	Correct	Partly true	Wrong
HIGH_BASE	**0.70%**	**78.87%**	**20.42%**
HIGH_WP8K	0.70%	69.01%	30.28%
LOW_BASE	0.00%	26.05%	73.94%
LOW_WP8K	0.00%	0.00%	100%

TABLE IX. EVALUATION RESULTS USING BLEU

Model variation	BLEU with maximum *n*-gram			
	4	**3**	**2**	**1**
HIGH_BASE	**4.76**	**8.56**	**15.43**	**27.63**
HIGH_WP8K	1.79	4.08	9.17	18.64
LOW_BASE	0	0.27	0.83	2.41
LOW_WP8K	0	0	0	0

The results show that the variant with the best results was the one with a large dataset and per-word tokenization, both in terms of accuracy BLEU score, achieving 0.70% accuracy for correct text and 78.87% for partially correct text, and a BLEU score range of 4.76 to 27.63. The variant with the large dataset and subword tokenization has an accuracy of 0.70% for correct text and 69.01% for partially correct text, and a BLEU score range of 1.79 to 18.64. This model has lower accuracy than the best model, although not as significant as the other models. The variant using a small dataset with word tokenization had an accuracy of 0% for correct text and 26.05% for partially correct text, and a BLEU score range of 0 to 2.41. This model has lower accuracy and is significantly different than the difference between the best model and the model with subword tokenization. The last variant using a small dataset with subword tokenization had an accuracy of 0% for correct text and 0% for partially correct text, and the BLEU score of 0 for all *n*-gram variations. This model had the lowest accuracy among the other model variations and could not correctly translate the texts tested.

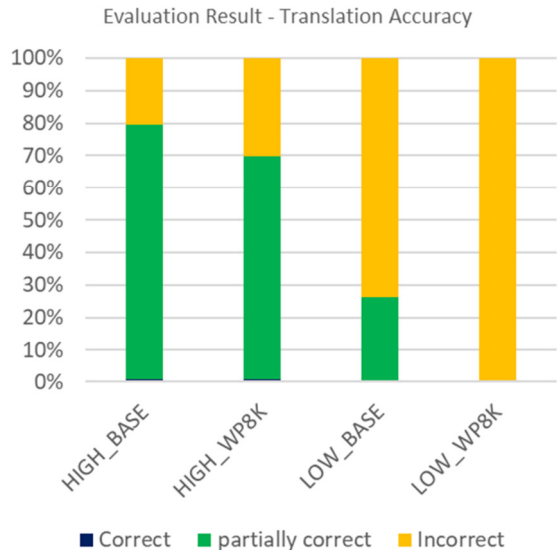

Fig. 11. Translation accuracy results.

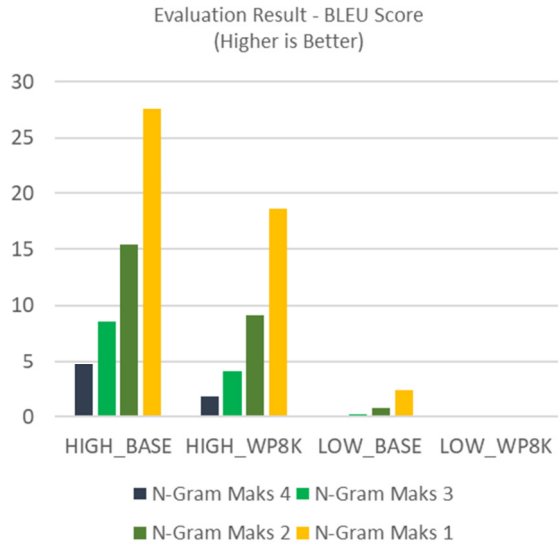

Fig. 12. BLEU score results.

When comparing the overall values for all variants, the use of subword tokenization, whose initial goal is to overcome words outside the dictionary data, actually makes accuracy decrease rather than using word tokenization. Using a small

dataset also reduces the accuracy compared to that with large datasets or the main reference.

Based on the test results, the use of these two variants to improve model accuracy was not proven. Several factors may influence the results of these values, including the final loss value on subword tokenization, which is around 10 times higher than tokenization per word, and the loss value on the small dataset variations, which is around 100 times higher than with the large dataset. These two factors may arise due to differences in computational time in each variant due to the limited available computing resources.

VII. CONCLUSION

The variant of the MT model with the best accuracy was the one that used a model configuration for a large dataset with tokenization per word, achieving 0.70% accuracy for correct text and 78.87% for partially correct text, while the BLEU score ranged from 4.76 to 27.63 in *n*-gram values from 1 to 4. The development of transformer-based MT can be improved if it is equipped with text pairs in the form of complete sentences. Apart from completing the data with text pairs in the form of complete sentences, other ways to improve the MT model is to include an algorithm that can complete or correct the input used, because words in Madurese have many accents and sometimes have different spellings as time changes.

Another way that might help develop MT models is to use RBMT on top of NMT. Several MT services in the early days of NMT development used RBMT on top of NMT to overcome NMT's shortcomings when translating sentences with text with a large number of words.

ACKNOWLEDGMENT

This work was funded by the Indonesian Ministry of Education, Culture, Research, and Technology (KEMDIKBUDRISTEK) for the financial year 2023, grant scheme: Penelitian Dasar Unggulan Perguruan Tinggi (PDUPT), contract numbers: 254/E5/PG.02.00.PT/2022 and 2466/UN46.4.1/PT.01.03/2022.

REFERENCES

[1] S. I. Abdullah and M. C. Yunita, "Distribution of Daily Use Local Language in Indonesia," *International Conference on Education and Language (ICEL)*, vol. 1, May 2014.

[2] antaranews.com, "East Java becomes favorite domestic destination in 2023," *Antara News*, Jun. 09, 2024. https://en.antaranews.com/news/315621/east-java-becomes-favorite-domestic-destination-in-2023.

[3] D. Haerudin, R. Dallyono, U. Kuswari, and D. Koswara, "Examining language attitudes and use: A survey of Indonesian university students' loyalty to their ethnic languages," *Indonesian Journal of Applied Linguistics*, vol. 14, no. 1, pp. 104–117, May 2024, https://doi.org/10.17509/ijal.v14i1.70364.

[4] Anwari, W. Purnaningtyas, and U. Hikmah, "The Level of Language Used by Madurese in Kalidandan, Pakuniran, Probolinggo," presented at the 1st International Conference on Science, Health, Economics, Education and Technology (ICoSHEET 2019), Jul. 2020, pp. 86–89, https://doi.org/10.2991/ahsr.k.200723.021.

[5] Y. Bawono and W. P. Wibowo, "Preserving Madurese Language, Is It Important?," in *Proceeding International Seminar of Multicultural Psychology*, 2023, vol. 3.

[6] S. Bird and D. Chiang, "Machine Translation for Language Preservation," in *Proceedings of COLING 2012*, Dec. 2012, pp. 125–134.

[7] H. Li and M. Ran, "Revitalizing Heritage Language through Natural Language Processing: Innovations and Challenges," *Rajapark International Journal*, vol. 1, no. 1, pp. 82–91, Feb. 2024.

[8] Y. Wu *et al.*, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." arXiv, Oct. 08, 2016, https://doi.org/10.48550/arXiv.1609.08144.

[9] F. A. Khan and A. Abubakar, "Machine Translation in Natural Language Processing by Implementing Artificial Neural Network Modelling Techniques: An Analysis," *International Journal on Perceptive and Cognitive Computing*, vol. 6, no. 1, pp. 9–18, Jul. 2020.

[10] W. J. Hutchins, "Machine Translation: A Brief History," in *Concise History of the Language Sciences*, E. F. K. Koerner and R. E. Asher, Eds. Pergamon, 1995, pp. 431–445.

[11] Y. Yuxiu, "Application of translation technology based on AI in translation teaching," *Systems and Soft Computing*, vol. 6, Dec. 2024, Art. no. 200072, https://doi.org/10.1016/j.sasc.2024.200072.

[12] J. Hu, "Neural Machine Translation (NMT): Deep learning approaches through Neural Network Models," *Applied and Computational Engineering*, vol. 82, pp. 93–99, Nov. 2024, https://doi.org/10.54254/2755-2721/82/20240944.

[13] S. Martin, "Advancements in Neural Machine Translation: Techniques and Applications," *Journal of Innovative Technologies*, vol. 7, no. 1, May 2024.

[14] D. I. De Silva and I. S. Gallage, "The Role of Syntax and Semantics in Rule-Based Translation: A Comprehensive Review," in *2024 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, Bahir Dar, Ethiopia, Nov. 2024, pp. 229–234, https://doi.org/10.1109/ICT4DA62874.2024.10777198.

[15] S. M. U. Qumar, M. Azim, and S. M. K. Quadri, "Addressing the data gap: building a parallel corpus for Kashmiri language," *International Journal of Information Technology*, vol. 16, no. 7, pp. 4363–4379, Oct. 2024, https://doi.org/10.1007/s41870-024-01979-8.

[16] M. A. Faheem, K. T. Wassif, H. Bayomi, and S. M. Abdou, "Improving neural machine translation for low resource languages through non-parallel corpora: a case study of Egyptian dialect to modern standard Arabic translation," *Scientific Reports*, vol. 14, no. 1, Jan. 2024, Art. no. 2265, https://doi.org/10.1038/s41598-023-51090-4.

[17] A. S. Dhanjal and W. Singh, "A comprehensive survey on automatic speech recognition using neural networks," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 23367–23412, Mar. 2024, https://doi.org/10.1007/s11042-023-16438-y.

[18] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[19] M. X. Chen *et al.*, "The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, 2018, pp. 76–86, https://doi.org/10.18653/v1/P18-1008.

[20] F. H. Rachman, N. Ifada, S. Wahyuni, G. D. Ramadani, and A. Pawitra, "ModifiedECS (mECS) Algorithm for Madurese-Indonesian Rule-Based Machine Translation," in *2022 International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, Denpasar, Bali, Indonesia, Nov. 2022, pp. 51–56, https://doi.org/10.1109/ICSINTESA56431.2022.10041470.

[21] N. Ifada, F. H. Rachman, M. W. M. A. Syauqy, S. Wahyuni, and A. Pawitra, "MadureseSet: Madurese-Indonesian Dataset," *Data in Brief*, vol. 48, Jun. 2023, Art. no. 109035, https://doi.org/10.1016/j.dib.2023.109035.

[22] S. Lankford, H. Afli, and A. Way, "Transformers for Low-Resource Languages: Is Féidir Linn!" arXiv, Mar. 04, 2024, https://doi.org/10.48550/arXiv.2403.01985.

[23] S. J. Mielke *et al.*, "Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP." arXiv, Dec. 20, 2021, https://doi.org/10.48550/arXiv.2112.10508.

[24] D. Sundararaman *et al.*, "Syntax-Infused Transformer and BERT models for Machine Translation and Natural Language Understanding." arXiv, Nov. 10, 2019, https://doi.org/10.48550/arXiv.1911.06156.

[25] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, vol. 1, pp. 1715–1725, https://doi.org/10.18653/v1/P16-1162.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[27] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization." arXiv, Jul. 21, 2016, https://doi.org/10.48550/arXiv.1607.06450.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, 2013, vol. 26.

[30] S. Zainudin, S. A. Kusuma, and Barijati, *Bahasa Madura*. Jakarta, Indonesia: Pusat Pembinaan dan Pengembangan Bahasa, 1978.

[31] A. Imankulova, T. Sato, and M. Komachi, "Filtered Pseudo-parallel Corpus Improves Low-resource Neural Machine Translation," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, no. 2, pp. 1–16, Mar. 2020, https://doi.org/10.1145/3341726.

[32] A. K. Ngo Ho and F. Yvon, "Optimizing Word Alignments with Better Subword Tokenization," in *Proceedings of the 18th Biennial Machine Translation Summit :Volume 1: Research Track*, Dec. 2021.

[33] H. Sujaini, "Mesin Penerjemah Situs Berita Online Bahasa Indonesia ke Bahasa Melayu Pontianak," *ELKHA : Jurnal Teknik Elektro*, vol. 6, no. 2, Nov. 2014, https://doi.org/10.26418/elkha.v6i2.9098.

[34] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001, https://doi.org/10.3115/1073083.1073135.