2022

MASTER THESIS

A Neural Network Approach to Bilingual Dictionary Induction for Indonesian Ethnic Languages

ACADEMIC SUPERVISOR: MURAKAMI Yohei

Graduate School of Information Science and Engineering
Ritsumeikan University

MASTER'S PROGRAM
MAJOR in Advanced Information Science and Engineering

STUDENT ID: 6611200091-8

NAME: KARTIKA Findra Resiandi

A Neural Network Approach to Bilingual Dictionary Induction for Indonesian Ethnic Languages

Kartika Findra Resiandi

Abstract

Indonesian ethnic languages are endangered. The most important stage in enriching low-resource languages is to create a bilingual dictionary. It has been demonstrated that the constraint-based technique aids in the induction of bilingual lexicons from two bilingual dictionaries via the pivot language, especially for the closely related ones. However, a common concern with the pivot-based approach if there are any mistakes made in the source-to-pivot translation will be carried over to the pivot-to-target translation. Furthermore, the low number of dictionaries for the input and the small size of the input dictionaries limit the number of the generated translation pairs.

Therefore, we proposed a neural network approach to create a bilingual dictionary between ethnic languages by extracting transformation rules of translation pairs. Once acquiring the rules, we can generate many translated words from a source word list. For example, between Indonesian and Minangkabau, the last phoneme "a" in Indonesian tends to turn "o" in Minangkabau, or the middle phoneme "ia" appears to turn "i", and many more. To this end, we employed a sequence-to-sequence model, where the encoder reads the input word and then extracts a feature of the input while the decoder generates a translated word from the feature. In this research, we focus on ethnic languages that have closely related to Indonesian. This research addresses the following two research issues.

Effective word tokenization for translating word pairs

In a sequence-to-sequence model, the encoder and decoder need to receive and generate a word as sequence data, respectively. Although the simplest sequence data is a sequence of characters, the length of the sequence becomes long. Therefore, we need to find effective tokenization to balance the kinds of tokens and the length of the sequence.

Low Resource Language has Low Data Size

There are several types of neural network architecture that receive sequence data as a word. Moreover, translation pairs data for training are small. Therefore, we need to adjust neural network architecture and model parameters such as epoch, batch size, and learning rate to prevent the model from overfitting and to improve the accuracy of the model.

To address the first issue, this research used Bi-LSTM as the encoder and LSTM as the decoder processes. We must identify the kind of tokens that are useful for extracting the transformation patterns. The first approach by character-level one hot embedding and the second approach uses the SentencePiece which implements Byte Pair Encoding where vocabulary size is required for the tokenization, which will have an impact on the input to the encoder and decoder.

To adjust network architecture and model parameters, we implemented learning rate schedule technique is learning rate decay. We set a learning rate is 0.001 then the learning rate decreases by 1% for every epoch above the 15th

We validated the proposed method by applying it to three language pairs from Indonesian to ethnic languages with fewer word pair translations: Indonesian to Minangkabau, Palembang, and Malay.

The following are the research's contributions:

Effective word tokenization for translating word pairs

Our character level model performance with an average precision of 83,92% outperforms Byte Pair Encoding (with vocabulary size from 33 to 300) models. We tried 7 times experiment with various vocabulary sizes, the maximum size used is 300. In general, the results are higher when the vocabulary is larger. However, the experimental results showed approximately the same vocabulary size as character level based achieved the highest performance among BPE models. Our neural network architecture works effectively for three Indonesian ethnic languages even with about half the size of input dictionaries (Minangkabau, Palembang, Malay) with an average precision of 74,6% 65.2%, 65.08% respectively.

Low Resource Language has Low Data Size

We also compare performance according to each pattern with a simple rule-based. The neural network approach outperforms the simple rule-based with an average precision of 66% and 34%.

A Neural Network Approach to Bilingual Dictionary Induction for Indonesian Ethnic Languages

Chapter 1 Introduction

Indonesia's riches extend beyond natural resources such as minerals, vegetation, and fauna. Furthermore, the archipelago's culture is highly diversified, and so does a variety of ethnic languages in Indonesia. The Austronesian language family includes Indonesian, derived from the Malay language. Since prehistoric times, Indonesian ethnic languages have developed, resulting in a different language for each ethnic group in Indonesia [12].

Currently, the phenomenon of ethnic language extinction in Indonesia has become a problem that grabs the attention of scholars, especially linguists. The Summer Institute of Linguistic states that the local languages are endangered and may cease to be spoken in Indonesia. The most important stage in enriching low-resource languages is to create a bilingual dictionary. It has been demonstrated that the constraint-based technique aids in the induction of bilingual lexicons from two bilingual dictionaries via the pivot language, especially for the closely related ones. However, a common concern with the pivot-based approach if there are any mistakes made in the source-to-pivot translation will be carried over to the pivot-to-target translation. They will be produced in all the target languages.

Therefore, we started the Indonesia Language Sphere project that aims at comprehensively creating bilingual dictionaries between the ethnic languages using a neural network approach in order to conserve local languages on the verge of extinction [6] As an expected result, the vocabulary of the ethnic language will expand, more people will learn it, and if there are no more speakers in the future, the language will become extinct. The current case for experiment focuses on Indonesian to Minangkabau, Palembang, and Malay languages, because the languages have a very high similarity with Indonesian, their geographical proximity is also near, as they are both in the Indonesian province of Sumatera, and since most of the nationalist writers who contributed to the early development of Indonesian were of Minangkabau ethnicity. Minangkabau language (closely linked to Malay) significantly influenced Indonesian in its formative years [10].

Because the Indonesian ethnic language is a low resource language, and it has a limited amount of data, we chose Minangkabau as the language to implement the proposed method in this study. We tried the Palembang and Malay language while learning the Minangkabau language with a neural network model, despite the lack of data. The Indonesian and Minangkabau languages have significant similarities, between two languages, we presume they have several phonetic transformation rules. For example, there appears to be a rule in Indonesian and Minangkabau that the last phoneme ”a” in Indonesian tends to turn ”o” in Minangkabau, while the middle phoneme “ia” appears to turn “i”. There are many more patterns in the language. Although this rule isn’t always applicable, it can help predict a rough translation as a preliminary translation. This study predicts the translation using character level embedding, compared to the SentencePiece method using the Bi-LSTM sequence-to-sequence model. Besides that, in this study we also compare the neural network performance with simple rule based as baseline to compare our model. When compared to simple rules, how effectively the proposed model can reproduce the well-known pattern.

Chapter 2 Related Work

This chapter explains the background of the study, the challenges of developing bilingual dictionary induction, and reference research.

2.1 Bilingual Dictionary Induction

Creating a bilingual dictionary is the first crucial step in enriching low-resource languages. Especially for the closely related ones, it has been shown that the constraint-based approach helps induce bilingual lexicons from two bilingual dictionaries via the pivot language [7, 8]. The low number of dictionaries for the input and the number of the generated translation pairs depends on the size of the input dictionaries. However, implementing the constraint-based approach on a large scale to create multiple bilingual dictionaries is still challenging in determining the constraint-based approach's execution order to reduce the total cost. Plan optimization using the Markov decision process is crucial in composing the order of creation of bilingual dictionaries considering the methods and their costs [9, 11].

In this research, we would like to extract transformation rules from the Indonesian to Minangkabau language. Table 1 shows the example of Indonesia Minangkabau dictionary.

Indonesian	Minangkabau
Apa	Apo
Merupakan	Marupokan
Kesudahannya	Kasudahnyo
Patuang	Patung
Balik	Baliak
Menderita	Mandarito
Panas	Paneh
Sekelilingnya	Sakaliliangnyo

Table 1. Example of Indonesian-Minangkabau words

2.2 A Neural Network Approach

Heyman et al. [2] have proposed a method to make bilingual lexical induction as a binary classification task in the biomedical domain for English to Dutch. They create a classifier that predicts whether a pair of words is a translation using character and word level, LSTM method. This study reveals that character-level representations successfully induce bilingual lexicons in the biomedical domain. In the charpairs experiment, the total average F score for the test data consisting of translation pairs with Greek or Latin origin is 55.04. Character level encoder representation in charpairs is their technique. The word level in this example is unrelated to this study, however they do have four experiments that mix the word and character levels.

Recently, deep learning is the most popular approach, utilize sequence-to-sequence learning, which consists of an encoder and a decoder [15]. Zhang et al. [17] presented a character-level sequence-to-sequence learning approach proposed in this study. RNN is the encoder-decoder technique used to generate character-level sequence representation for the task of English-to-Chinese. For the construction of an RNN encoder-decoder in the field of machine translation [19], Yang et al. [20] developed the encoder Bidirectional LSTM and the decoder LSTM based sequence-to-sequence model. Wazery et al [21] utilizing Sequence-to-Sequence model with encoder as BI-LSTM and LSTM as decoder to summarize the Arabic text. They also compare with GRU and LSTM. Bi-LSTM is the best result by BLEU score is 0.41.

Chapter 3 Sequence-to-Sequence Model

3.1 Overview

Deep learning techniques called sequence-to-sequence are employed to address machine translation issues. Sequence-to-sequence (Seq2Seq) is a model based on recurrent neural network that predicts the words in the output one at a time, which will then be integrated into a sentence, after reading each word from an input sentence one at a time. Two Recurrent Neural Networks (RNN), the encoder and decoder combine to generate the Seq2Seq model. The target sequence is generated by the decoder using the context vector as the "seed" by the encoder's RNN network, which encodes the input sequence into a fixed-size context vector. Therefore, the Seq2Seq model is often also referred to as the encoder-decoder model. RNN have a difficulty called vanishing gradients, which is why some sequence-to-sequence models use a development of recurrent neural networks called long short-term memory (LSTM) [17]. LSTM has also been used frequently to represent intelligence in language processing.

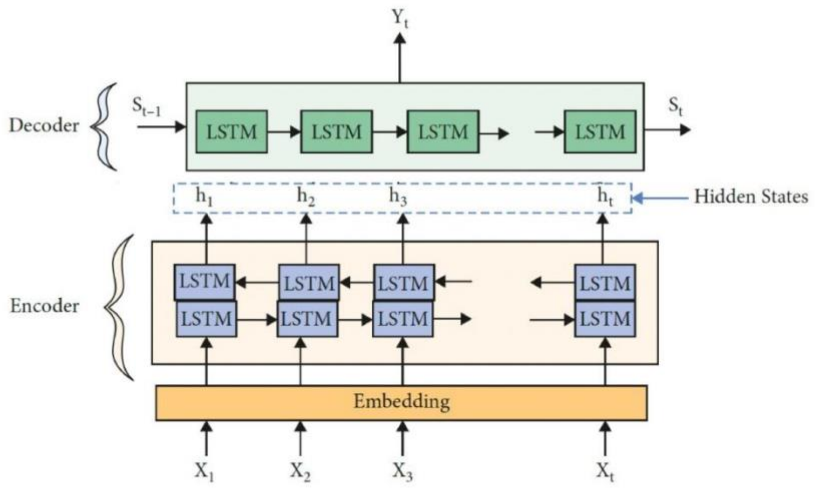


Figure 3.1: Se2Seq Model consisting of LSTM and Bi-LSTM [21]

3.2 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) is an upgraded Recurrent Neural Network (RNN) that is used to overcome the problem of vanishing and exploding gradients [3]. LSTM addresses the problem of long-term RNN reliance, in which

RNNs are unable to predict input data stored in long-term memory but can make more accurate predictions based on current information. The LSTM architecture can store large amounts of data for lengthy periods of time. They are applied to time-series data processing, forecasting, and categorization. Memory cells and gate units are the key components of the LSTM architecture. Forget gate, input gate, and output gate are the three types of gates in an LSTM. Figure 3.1 illustrates the structure of the LSTM model.

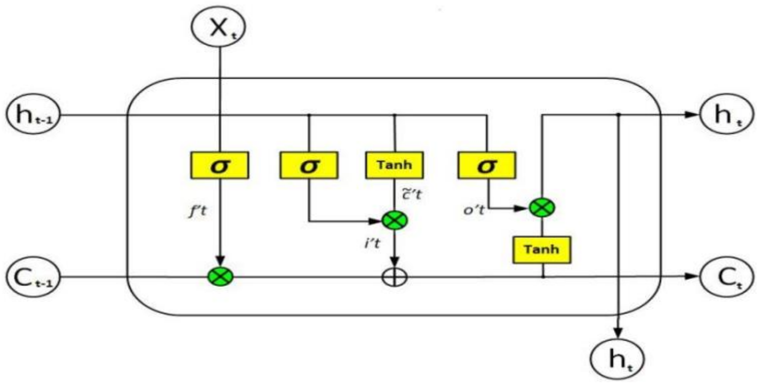


Figure 3.2: Unit structure of the LSTM [18]

Cell memory tracks the dependencies between components in the input sequence. New values that enter the cell state are handled by the input gate. The LSTM unit utilizes a forget gate to select the value that remains in the cell state. The value in the cell state that remains will be sent to the output gate, where the LSTM activation function, also known as the logistic sigmoid function, will be used to start the calculation. The tanh and sigma symbols represent the types of activation functions employed in the neural network's training layers.

Allowing information to flow through it unmodified, a sigmoid gate, which restricts how much information may pass through, is another essential feature of LSTM. The outputs of the sigmoid layer, which vary from zero to one, specify how much of each component should be permitted to pass. The equation that controls the LSTM flow is as follows:

$$f_t = \sigma(w_f \cdot \left[h_{t-1}, x_t\right] + b_f$$
$$i_t = \sigma(w_i \cdot \left[h_{t-1}, x_t\right] + b_i$$
$$C_t = \tanh\,(w_c \cdot \left[h_{t-1}; x_t\right] + b_c$$

$$\tilde{C}_t = f_t \times C_{t-1} + i_t * C_t$$
$$o_t = \sigma(w_o \cdot \left[h_{t-1}, x_t\right] + b_o$$
$$h_t = o_t \times tanhC_t$$

where

o_t	: at time t, ouput gate
i_t	: at time t, input gate
h_t	: output at time t
f_t	: forget gate, at time t
x_t	: input at time t
σ	: sigmoid function
C_t	: the state of the cell at time t
$w_{o,} w_f, w_i, w_c$	: weights that have been trained
b_c, b_i, b_f	: trained biases

3.3 Bi-Directional Long Short-Term Memory (Bi-LSTM)

RNN has an advantage in the reliance between coding inputs. However, LSTM has an advantage in resolving RNN's long-term issues. Improvements are made with Bi- RNN because only one direction of previous contextual information can be used by LSTM and RNN [13]. As a result of the advantages of each technique, the LSTM form is kept in the cell memory, and Bi-RNN can process information from the previous and next contexts, resulting in Bi-LSTM [13]. Bi-LSTM can leverage contextual information and generate two separate sequences from the LSTM output vector. Each time step's output is a mixture of the two output vectors from both directions, as the Figure 3.3 below, where h_t is the forward or backward state [16]. The Bi-LSTM network computes the output vector sequence = (y_1, y_2,.., y_n) and the hidden vector sequence h = (h_1, h_2 , ... , h_n). The semantic representation of the input sequence is thoroughly examined by Bi-LSTM in both directions. In contrast to the decoder, which seems sequential from right to left, the encoder's process appears sequential from left to right.

Forward hidden is represented in a symbol $\vec{h}$ and backward hidden is represented by the symbol $\overleftarrow{h}$. Following the equation below, the forward hidden sequence and the backward hidden sequence are iterated at each time step. For the front layer in the first equation, iteration begins at t = 1 and continues through N.

$$\vec{h} = \mathrm{H}\,(b_{\vec{h}} + W_{\vec{h}\vec{h}}\overrightarrow{h_{t-1}} + W_{x\vec{h}}X_t)\ (1)$$
$$\overleftarrow{h} = \mathrm{H}\,(b_{\overleftarrow{h}} + W_{\overleftarrow{h}}\overleftarrow{h}_{t-1} + W_{x\overleftarrow{h}}X_t)\ (2)$$

Iteration for the backward layer in the second equation begins at $t = N$ and goes to 1. In all parameters, forward layers are represented by arrows pointing left to right, and backward layers by arrows pointing right to left. In order to encode the information, the letters in the input are first represented as $t - t_h$ and then encoded in h_t. W and b are variables that represent the weight of the matrix and the bias vector, respectively. Figure 3 depicts the combination of LSTM and Bi-RNN.

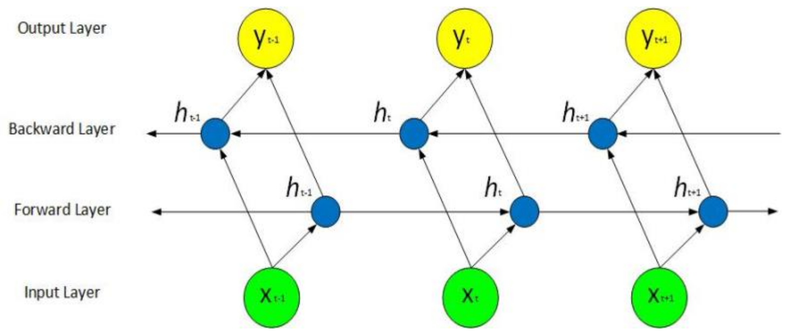


Figure 3.3: Bi-LSTM Architecture [18]

Chapter 4 Tokenization for Sequence-to-Sequence Model

4.1 Character Level Sequence-to-Sequence Model

The first approach is using character level one hot embedding where words will be separated as characters, and each vector has the same length size adjusted by total characters. Then, sequence-to-sequence (Seq2Seq) model, which employs RNN encoders and decoders. In this study, Bi-LSTM encoder and decoder processes are employed. The Bi-LSTM encoder creates a representation of the input words by parsing each character of the word in the source language (Indonesian). The LSTM decoder uses the encoder's output as input and outputs the target language character by character (Minangkabau). Figure 4.1 describe the example of Indonesian word “ada” change into character level one hot embedding vectorization with the same size of all characters.

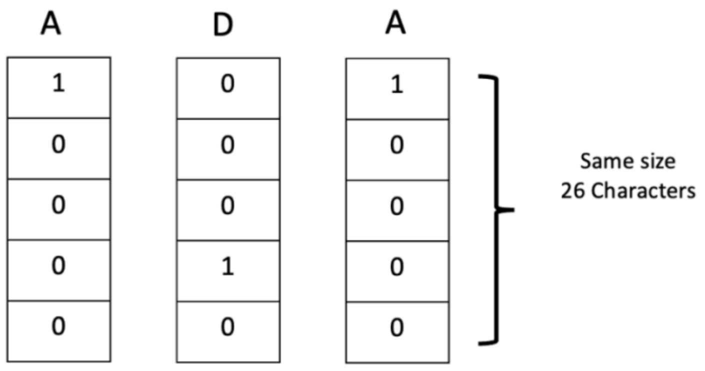


Figure 4.1: Example of one hot embedding vector

The character-level encoder computes a vector representation from a word of character sequences in the source language. Replace each character by a one-hot vector. A one-hot vector is a vector with all zeros except at the dimension that corresponds to the position of the character in the vocabulary. For instance, the one-hot encoding of the character a would be <1, 0, 0, 0, ... >, the one-hot encoding of b would be <0, 1, 0, 0, ... > and so on. The sequence of vectors will be the input to Bi-LSTM.

Figure 4.2 shows the Seq2Seq model considered in this study with a two-layered Bi-LSTM encoder and LSTM decoder. The encoder's functions are to character by character read the input sequence, build context, and extract a summary of the input. The decoder will provide an output sequence in which the previous character affects every character in each time step as well as the next character that emerges. The marker <eos> denotes the end of a sentence, and it will determine when we stop predicting the following character in a series [15]. Following the construction of the encoder and decoder network architectures in this typical end-to-end framework, a training approach may be utilized to obtain an optimal word pair translation model and to keep the character order is referred to as a cell state or memory cell since the horizontal line going across the bottom of the diagram is in the source and target words, the input (Indonesia) and output (Minangkabau) sequence must be treated in time order. For the Indonesian language, there are 28 input tokens used, and there are 31 output tokens as Minangkabau language.

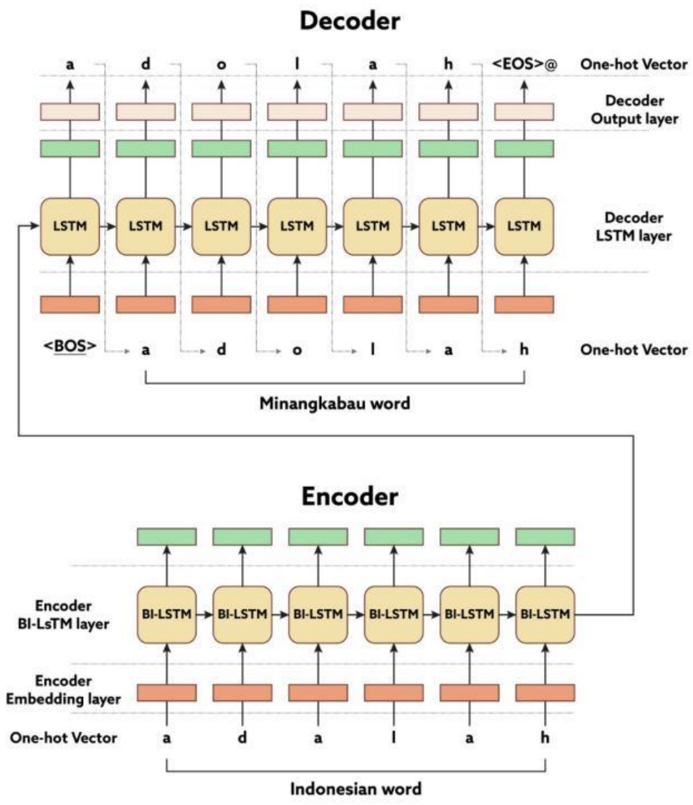


Figure 4.2: Character Level Sequence-to-Sequence Model [18]

4.2 Byte Pair Encoding-based Tokenization

BPE builds a base vocabulary consisting of all symbols found in the set of unique words, then learns merge rules to combine two symbols from the base vocabulary to create a new symbol. It continues to do until the vocabulary has grown to the required size. BPE algorithm replaces the data byte pairs that occur most frequently with a new byte until the data can no longer be compressed since no byte pair occurs most frequently. The steps in the training procedure are as follows [14]:

1) Gather a huge amount of training data.
2) Determine the vocabulary's size.
3) At identify the end of a word, add an identifier (</w>) to the end of each word, and then calculate the word frequency in the text.
4) Calculate the character frequency after dividing the word into characters.
5) Count the frequency of consecutive byte pairs from the character tokens for a predetermined number of rounds and combine the most frequently occurring byte pairing.
6) Repeat step 5 until performed the necessary number of merging operations or reached the specified vocabulary size.

The input text is treated as a sequence of unicode characters by SentencePiece. Whitespace is also treated like any other symbol. SentencePiece expressly handles whitespace as a fundamental token by first escaping it with the meta symbol "___" (U+2581) [5]. Meanwhile the symbol of '\n' is the end of string.

4.2.1 SentencePiece

The second method we presented is SentencePiece as subword tokenization. According to Kudo [5], subword tokenization implements SentencePiece, subword-nmt, and wordpiece model features. Subword vocabulary is built by using the BPE segmentation method to train a SentencePiece tokenization model, which divides words into chunks of characters based on vocabulary size to make pattern detection easier. According to Kudo and Richardson [5], a SentencePiece is made up of four primary parts:

1. *Normalizer* is a module that may canonically normalize identical Unicode characters.
2. Trainer, trains the model for subword segmentation of the normalized corpus.
3. Encoder, uses the subword model taught by the trainer to tokenize and normalize the input text into a list of subwords.
4. Decoder, which converts subwords into normalized text, reverses their order.

4.2.1 BPE-Based Tokenization in Indonesian Ethnic Languages

BPE was added to our research methodology because Indonesian ethnic languages now utilize an alphabet script established by the Dutch despite having original scripts in the past. Dutch people appeared to assign a chunk of alphabets to phonemes of Indonesian ethnic languages when teaching the alphabets to them [12]. As a result, all Indonesian ethnic languages can use the same tokens. Furthermore, with each phonetic development, languages belonging to the same language family descended from the same proto language. As a result, we assume a phonetic-based strategy is preferable to a character-based method. The number of words to be processed into tokenization is known as vocabulary size, which in this case refers to the number of most often occurring characters, including the symbol like </unk>, and whitespace. We employ a wide range of vocabulary sizes. The following step is the same as the first method. Figure 4.3 shows that the encoder and decoder input results because of character splitting from BPE in this illustration of the seq2seq model. This approach differs from Figure 4.2 in that the encoder (Indonesian word) and decoder (Minangkabau word) inputs are different.

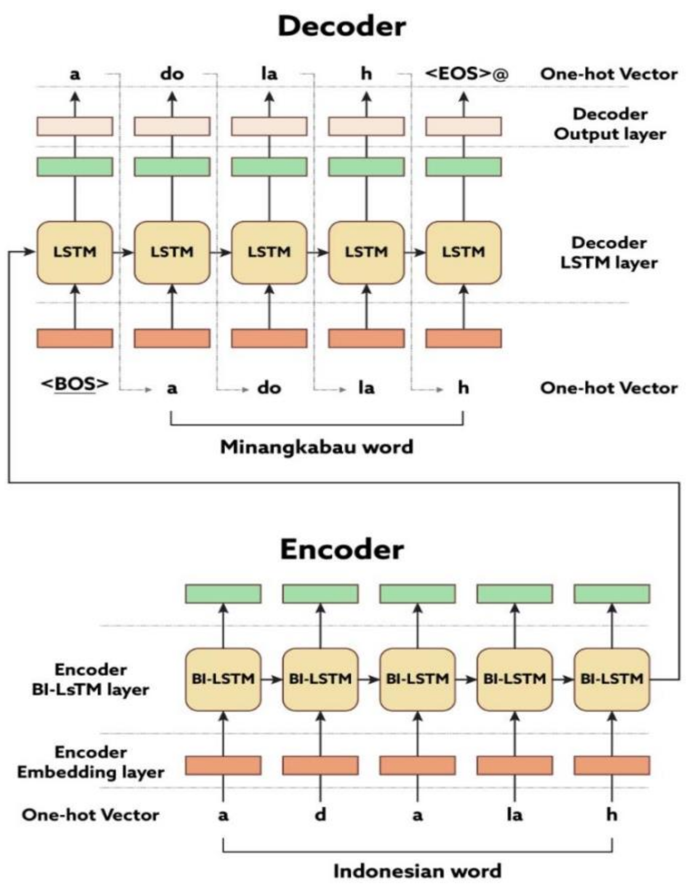


Figure 4.3: SentencePiece Sequence-to-Sequence Model [18]

The results of the chunk of characters from the BPE will vary when utilizing a higher vocab size. Except for alphabets, the vocabularies obtained from BPE 40 and 100 are summarized in the Table 4.1 and 4.2. The number of vocabularies in Indonesian and Minangkabau is the same overall (7 and 66, respectively), the number of vocabularies in Indonesian and Palembang is 10 and 70 also 9 and 69, respectively, and the number of vocabularies in Indonesian and Minang is 10 and 70 also 9 and 68, respectively. According to the Table 4.1, character pieces are more obtained if use larger vocabulary sizes. The alphabet following the “_” symbol is a piece of characters at the beginning of the term in vocabulary that begins with the "_" symbol. Example in the Minangkabau language, the difference between the character pieces sa and _sa is that sa indicates that the character is not at the beginning of the word.

Tokenization BPE is generated by library SentencePiece [5]. The tokenization with vocab size=40 is done almost one by one like character-based tokenization because vocab size=40 is nearly the same as the number of alphabets. The more vocabulary size is used, the more character pieces are generated. Tokenization results refer to the Table 4.1,4,3, and 4,5 that shows the words utilizing vocabulary size =100 there seems to be a wider variety of character pieces compared to utilizing vocabulary size =40.

Table 4.1: Example of tokenization BPE with different vocabulary size

Vocab Size=40		**Vocab Size=100**	
Indonesian	Minangkabau	Indonesian	Minangkabau
[_,n,an]	[_,y,a,ng,\'n']	[_,n,an]	[_,ya,ng,\'n']
[_pa,d,o]	[_,p,a,d,a,\'n']	[_pa,do]	[_pa,da,'\n']
[_a,d,o,la,h]	[_,a,d,a,l,a,h,\'n']	[_a,do,la,h]	[_a,da,la,h,'\n']
[_,s,a,g,i,r,o]	[_,s,e,g,e,ra,'\n']	[_**sa**, gi, ro]	[_se,g,e,ra,'\n']
[_,d,a,s,an,y,o]	[_,d,a,s,a,r,nya,'\n']	[_,da,**sa**,nyo]	[_,da,sa,r,nya,'\n']

Table 4.2: Vocabularies obtained from BPE Indonesian-Minangkabau

Language	**Vocabsize=40**	**Vocabsize = 100**
Indonesian	an , ng, nya, ta, kan, _di, _men	an, ng, kan, ta, _di, la, nya, ra, da, si, _ke, _ber, ti, ba, li, ga, ri, ja, er, tu, bu, _se, at, in, _men, ma, sa, _per, ka, en, di, wa, ku, _meng, ya, na, _me, _pen, te, mp, ca, _p, _ter, ru, du, _mem, de, pa, or,un, ar, ju, is, _ka, bi, _ko,_ma, re, on, _ba, _pe, _pem, tan, pu, gu, al, ran, asi
Minangkabau	an, ang, _pa, _di, _ma, _ba, ng	an, ng, _di, _ba, ra, si, la,_pa, nyo, _ka, ta, da, ang, _ma, ik, kan, li, ri, ti, ak, tu, ka, _sa, _man, ja, ah, _ta, bu, ga, ek, in, ba, ku, sa, ma, su, di, ru, ya, _a, mp, _pan, to, wa, pa, ca, ran, du, ro, lu, tan, lo, mba, angan, ju, bi, pu, re, han, en, te, do, de, ko, gu, gi, _mam

Table 4.3. Example of tokenization BPE with different vocabulary size

Vocab Size=40		**Vocab Size=100**	
Indonesian	Palembang	Indonesian	Palembang
[_,c,a,b,a,i]	[_,c,a,b,i,k,\’n’]	[_,ca,b,a,i]	[_,c,a,b,ik’n’]
[_s,e,d,e,r,e,t]	[_ba,d,e,r,e,t,\’n’]	[_se,de,r,e,t]	[_ba,de,r,e,t’\n’]
[_,a,d,i,l]	[_,a,d,e,l,\’n’]	[_,a,di,l]	[_a,de,l’\n’]
[_,b,at,a,s,nya]	[_ba,t,a,s,nye’\n’]	[_ba, ta,s, nya]	[_ba,t,as,nye\n’]
[_,a,d,a,p,u,n]	[_,a,d,e,p,u,n’\n’]	[_,a,da,p,un]	[_,a,de,p,un’\n’]

Table 4.4: Vocabularies obtained from BPE Indonesian-Palembang

Language	**Vocabsize=40**	**Vocabsize = 100**
Indonesian	an, kan, nya, ng, _ber, _di, _pe, ang, si, at	an, kan, ng, nya, _ber, _di, at, ta, si, la, ang, _se, er, da, ga, _ke, ja, in, tu, _men, bu, ar, ti, _meng, ah, en, _per, ca, wa, ri, mp, _p, di, li, _ter, _ba, _mem, un, sa, as, ya, ak, ma, is, du, al, _per, ju, or, ku, _ka, ru, lu, de, us, ur, mu, _te, na, gu, ik, ung, _pen, _ma, te, el, _be, on, _sa, nda
Palembang	an, ke, ng, nye, _ba, at, ang, _di, la	an, ke, nye, ng, _ba, at, _di, ang, ak, ar, _n, _me, si, _ta, _se, _pe, da, ja, en, ah, _be, _p, er, as, mp, al, li, in, ek, am, ti, _ke, _te, la, de, tu, _ka, _g, di, _sa, du, ur, te, wa, _ma, or, ju, su, on, un, ok, ap, ik, _c, uk, ung, _pa, ut, se, ge, is, _da, _bu, _nga, lu, us, _tu, gha, asi

Table 4.5: Vocabularies obtained from BPE Indonesian-Malay

Vocab Size=40		**Vocab Size=100**	
Indonesian	Malay	Indonesian	Malay
[_,a,k,t,a]	[_,a,k,t,e,\'n']	[_, ak, ta]	[_a, k, te, \'n']
[_ber,k,i,s,ar]	[_be,k,i,s,a,\'n']	[_ber, k, is, ar]	[_be, k, i, sa, \'n']
[_,s,y,u,k,u,r,an]	[_,s,y,u,k,o,r,an,\'n']	[_, s, y, uk, ur, an]	[_, s, y, u, k, o, r, an, \'n']
[_,t,e,b,i,ng]	[_,t,e,b,e,ng,'\n']	[_te, b, ing]	[_te, be, ng, \'n']
[_ber,t,a,j,u,k]	[_be,ta,j,o,k'\n']	[_ber,ta,ju,k]	[_be,ta,j,ok'\n']

Table 4.6: Vocabularies obtained from BPE Indonesian-Malay

Language	**Vocabsize=40**	**Vocabsize = 100**
Indonesian	_an, _di, ng, _ber, ar, kan, nya, si, ang, at	an, _di, _ber, ng, kan, nya, ar, da, si, at, _se, ta, ti, ang, la, en, _ter, _ke, ah, in, ra, al, _ba, li, ak, tu, ri, ur, er, di, or, sa, _ka, as, un, is, ga, _te, du, ung, ku, am, us, _bu, ju, ing, asi, on, _pe, um, _ja, _be, _per, ir, uk, ya, _de, ru, _bi, te, lah, ut, ek, _ga, se, _tu, kat, ol, _ta, gu
Malay	_ng, an, _di, _be, la, ba, kan, ta	_di, ng, an, _be, kan, la, ra, nye, ta, _te, _se, si, ba, da, ka, _a, at, ti, ga, _ke, sa, ri, _ba, te, wa, ma, li, ja, ang, tu, ge, se, de, ya, _pe, pe, ku, na, re, ke, _ka, be, _ta, di, _bu, _tu, to, _ha, pu, ju, lu, nd, al, in, du, ong, ok, _sa, pi, _per, bo, gi, _bi, _pa, mpa, lah, on, _ma

Chapter 5 Evaluation and Discussion

5.1 Data Set

The secondary data is obtained from Nasution et al. [10] and Koto et al [4] with a total of 13,761-word translation pairs. Pre-processing the data is completed by deleting duplicate word pairs and constructing an array of word pairs in the form of a data type dictionary given by Python. Because in this case, there are various word pairings of Indonesian to Minangkabau that have several meanings. A dictionary is made up of a set of key-value pairs. Each key-value pair corresponds to a certain value Baidalina et al. [1]. The data is validated to Minang speakers after the duplicate data has been removed. As a result, there are 10277 translation pairs in the complete set of data. In other experiments to create an Indonesian-Palembang and Indonesian-Malay, unfortunately we have a lack of training data. The secondary data is also obtained from [10] in Indonesian-Palembang there are 5098 translation pairs, then divided into 80% of training, and 20% of testing. There are 4078 translation pairs Indonesian-Palembang in the training data set and 1020 in the testing data set. In Indonesian-Malay is 5229 translation pairs, divided into 4183 training, and 1046 testing data set. The model's performance is evaluated using a 5-Fold Cross-Validation.

Table 5.1: Summarizing the dictionary sizes and test and training data sizes.

Language Pairs	Total Translation Pairs	Training Data	Testing Data	Number of Tokens
Indonesian-Minangkabau	10277	8221	2056	28 and 31
Indonesian-Palembang	5098	4078	1020	28 and 29
Indonesian-Malay	5229	4182	1046	27 and 30

5.2 Evaluation Method by K-Fold Cross Validation

In this study, validation will be carried out using K-Fold. Cross. The original data are randomly divided into k for validation. In machine learning, cross validation is an approach to determine whether a model has good generalizations, as shown in figure 5.1 . (Able to have good performance on unseen examples).

Cross validation is a validation technique used to determine whether the network model can generalize data from the training phase into independent data. This technique is frequently used to assess how accurate the outcomes provided by the predictor model throughout the training phase are in the context of forecasting and prediction.

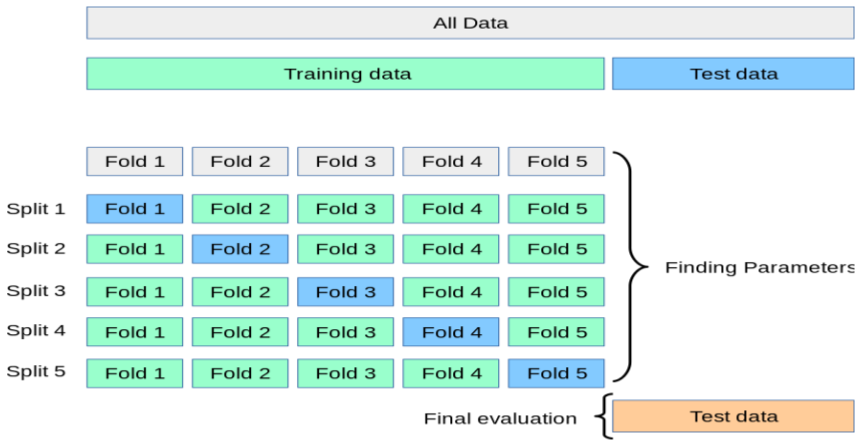


Figure 5.1: K-Fold Cross Validation (source: scikit-learn.org)

5.3 Parameter Design

In the first method, two models to find translation word pairs will be examined by Bidirectional Long Short-Term Memory, and Long Short-Term Memory to improve and compare performance with previous research [2]. We utilize the parameters selected for both models in Table 5.2. In this case, the implemented learning rate schedule technique is learning rate decay, we choose an initial learning rate, then reduce it progressively according to a scheduler. We set a learning rate is 0.001, then the learning rate decreases by 1% for every epoch above the 15th. A slower learning rate may allow the model to acquire a more optimal or even globally optimal set of weights, but it will take much longer to train the model.

Table 5.2: Model Parameter

Character Level and SentencePiece with BPE		
Parameter	**BiLSTM**	**LSTM**
Embedding Size	512	512
Epoch	80	80
Batch Size	64	64

5.4 Baseline

The use of LSTM for bilingual lexical induction has been studied in the biomedical field [2]. *Angiography:angiografie, intracranial:intracranieel, cell membrane:celmembraan, and epithelium:epitheel* are a few examples of English-Dutch translation pairs in the biomedical field. Their study uses a feed-forward neural network to perform binary classification tasks. They use a fully connected feed-forward neural network to the concatenation of source and target, which is provided as input to the network, to integrate these word-level and character-level representations.

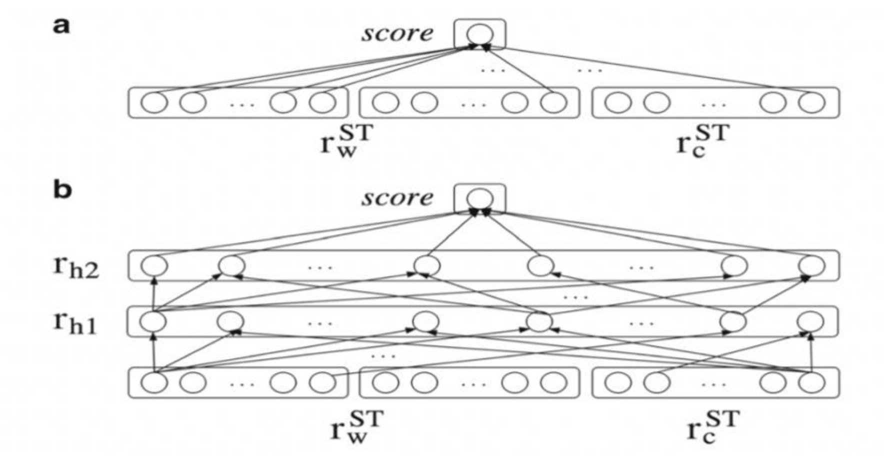


Figure 5.2: Illustrations of the classification component with feed-forward networks of different depths [2]

In the above figure they have two architecture models. Part a is the number of layers between the representation layer and the output layer (H), they set to 0, it means there are no hidden layer. r^{ST} representations the source and target in word level and character level. On the other hand, at the word level, the classifier must combine the embeddings of the source and target words in order to make an informed choice rather than merely computing a weighted sum of them, therefore in part B they use two hidden layers.

The binary classification task in their study with the input is a pair of words, and the output is a number between 0 and 1. An output value close to 1 means, they are likely each other's translations (in machine learning this is also called a positive example). Values close to 0 mean that the model thinks the input is just a pair of random words (a negative example). Here's a simple example translate between English and Dutch and the following are the source and target vocabularies:

V^{S}= [the, cat, mat, dog]

V^{T} = [de, het, kat, hond]

In theory, any combination of a word in the source vocabulary and a word in the target vocabulary would be a valid input:

$V^{S}\ x\ V^{T}$ = (the, de), (the, het), (the, kat), (the, dog), (cat, de), (cat, het), …… (dog, hond)

However, when the vocabularies are large will have many pairs to feed to the classifier. Thus in practice they use two heuristics to only select the most promising pairs. Their first heuristic is to only select pairs which have a low character-level edit distance (i.e., they pair words that 'look' the most similar) and second heuristic is to select pairs that have similar multilingual embeddings. Then, feed the union of candidates heuristics 1 and 2 to the classifier. Example:

candidates heuristic 1 = (the, het), (cat, kat), (mat, kat), (dog, de)

candidates heuristic 2 = (the, de), (cat, kat), (mat, dog), (dog, hond)

Even, they method is different to create a classifier that predicts whether a pair of words is a translation, on the other hand their study also wants to find translation. Their study utilizing LSTM network become a basis for our research.

5.5 Evaluation Result

Table 5.3 : Evaluation of SentencePiece with BPE in Indonesian-Minangkabau

Vocab Size	K-Fold Cross-Validation					
	K = 1	K = 2	K = 3	K = 4	K = 5	Average
33	79.96	76.55	78.84	**81.71**	**80.78**	79.56
35	76.11	76.89	79.42	74.31	**80.73**	77.49
40	72.12	72.88	75.23	**75.99**	71.64	73.59
50	67.12	62.15	66.97	**67.41**	64.29	65.58
80	58.73	**59.32**	53.35	54.12	56.47	56.39
100	49.36	48.24	49.46	49.70	48.78	49.10
300	34.85	34.93	30.31	35.76	36.19	34.40

This study uses two scenarios to find the optimal seq2seq model with the best performance. When comparing the character level and SentencePiece approaches with the seq2seq model, the character level seq2seq method generates a more accurate translation of word pairs.

Table 5.4: Evaluation of character-level model in Indonesian-Minangkabau

Method	**K-Fold Cross-Validation**					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Ave-rage**
BiLSTM (encoder), LSTM (decoder)	**84.72**	83.7	82.67	83.6	**87.5**	83.92
LSTM (encoder & decoder)	76.79	74.56	77.82	**78.21**	75.87	76.65

According to the Table 5.2 and Table 5.3, the results demonstrate that character-level tokenization, as opposed to BPE tokenization, is more useful for translating words to words. The vocabulary size has a minimum and maximum value. The minimum number necessary for this experiment data is 33. The experiment was run seven times with different vocabulary sizes, the maximum vocabulary size used was 300. When utilizing a minimal vocabulary size in BPE, it indicates that the number of tokens is approximately the same as character level based.

However, as shown in Table 5.2, the tokenization outcomes from the source and target pairs will vary more as the vocabulary size increases, which has an impact on the BPE performance outcomes. Perhaps, because the vector length is shortened, the data is likely to be less informative, making it more difficult for the model to recognize. In general, the larger the vocabulary size, the higher the results. It is also probably because the data is word-to-word pairs translation instead of sentence to sentence.

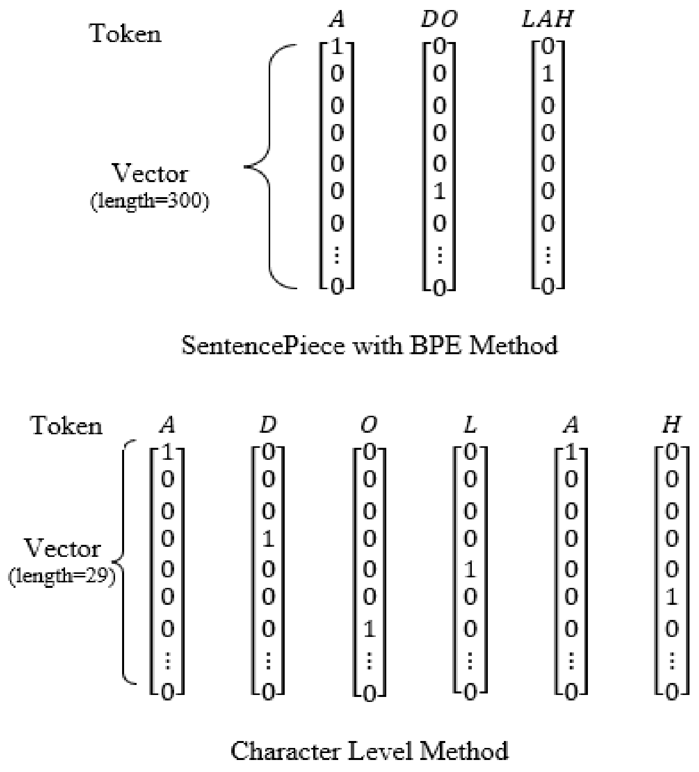


Figure 5.3 Comparison between SentencePiece with BPE and Character level method

Figure 5.3 illustrates the possibilities of why the character-based method is superior to the BPE-based method. As shown in Table 5.3, the bigger the vocabulary size, the lower the translation accuracy results. For an example of the Minangkabau word is adolah, if we use the vocabulary size=300, the number of tokens decreases, while the length of the vectors representing the tokens becomes longer because the vectors need more expression power.

The same methodology of character level, and BPE with Bi-LSTM Seq2Seq model was utilized in other experiments to create an Indonesian-Palembang, Indonesian-Malay. Unfortunately, we have a lack of training data. The secondary data is also obtained from [10] in Indonesian-Palembang there are 5098 translation pairs, then divided into 80% of training, and 20% of testing. There are

4078 translation pairs in the training data set and 1020 in the testing data set. We conducted three times of BPE experiment with vocabulary sizes 31, 33, 35, 40, 50, 80, and 100. Minimum value of vocabulary size in this case is 31. In character level based the number of tokens in Indonesia is 28, and in Palembang is 29. First, the total number of translation pairs that utilized in Indonesian-Malay is 5229, divided into 4183 training, and 1046 testing. In this Byte Pair Encoding experiment we are also using the Indonesian-Palembang scenario. In this instance, the minimum vocabulary size is 32, and the number of tokens used in character level-based is 27 for Indonesian, and 30 for Malay.

We used the optimal setting of Bi-LSTM as encoder and LSTM as decoder to generated translation pairs in the character level based experiment on Indonesian-Palembang and Indonesian-Malay. Additionally, we experimented with creating Indonesian-Minangkabau word pairs with a data size of approximately 5000 translation pairs. There are separated into 4183 training pairs and 1046 testing pairs. This experiment aims to contrast the result of Indonesian-Malay and Indonesian-Palembang translations. The Indonesian-Minangkabau translation continues to have the highest yield based on the K-fold average while using only half the amount of data.

Table 5.5: Evaluation of Character Level Model in Indonesian-Minangkabau with Half Data Size

Method	**K-Fold Cross-Validation**					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Ave-rage**
BiLSTM (encoder), LSTM (decoder)	65.96	68.54	70.65	71.79	**72.37**	69.89
LSTM (encoder & decoder)	46.94	48.85	45.88	52.19	**57.55**	50.28

Table 5.6: Evaluation of SentencePiece with BPE in Indonesian-Palembang

Vocab Size	**K-Fold Cross-Validation**					
	K = 1	K = 2	K = 3	K = 4	K = 5	Average
31	63.82	58.92	**64.11**	61.17	63.13	62.23
33	62.54	61.27	62.05	60.39	**62.64**	61.77
35	58.13	61.66	**63.23**	60.98	62.64	61.32
40	45.60	43.3	45.60	44.26	**45.79**	44.91
50	37.45	37.05	39.80	**41.07**	39.11	38.89
80	30.98	30.29	**32.25**	29.50	29.21	30.44
100	25.88	23.33	**26.66**	25.58	25.29	25.34

Table 5.7: Evaluation of character-level model Indonesian-Palembang

Method	**K-Fold Cross-Validation**					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Average**
BiLSTM (encoder), LSTM (decoder)	**63.82**	62.45	63.23	60.29	62.84	62.52
LSTM (encoder & decoder)	44.31	41.47	**45.19**	43.43	44.41	43.76

Table 5.8: Evaluation of SentencePiece BPE in Indonesian-Malay

Vocab Size	**K-Fold Cross-Validation**					
	K = 1	K = 2	K = 3	K = 4	K = 5	Average
32	61.95	62.04	61.66	61.56	**62.23**	61.88
33	58.22	62.23	65.96	64.34	**67.59**	63.66
35	51.62	55.64	59.75	57.74	**59.84**	56.91
40	53.72	48.62	**55.0**	47.64	53.62	51.79
50	**39.16**	36.04	35.55	36.13	38.04	36.58
80	26.57	29.06	30.11	29.92	**31.73**	29.47
100	21.03	22.84	23.13	23.61	**24.76**	23.07

Table 5.9: Evaluation of character-level model Indonesian-Malay

Method	K-Fold Cross-Validation					
	K = 1	K = 2	K = 3	K = 4	K = 5	Average
BiLSTM (encoder), LSTM (decoder)	64.72	**66.15**	65.20	65.96	63.38	65.08
LSTM (encoder & decoder)	47.99	39.57	40.63	46.36	**49.80**	44.87

Results of an average precision comparison of language pairings are shown in Figures 5.4 and 5.5. Figure 5.4 depicts the comparison result using BPE, while Figure 5.5 depicts the comparison result based on character level.

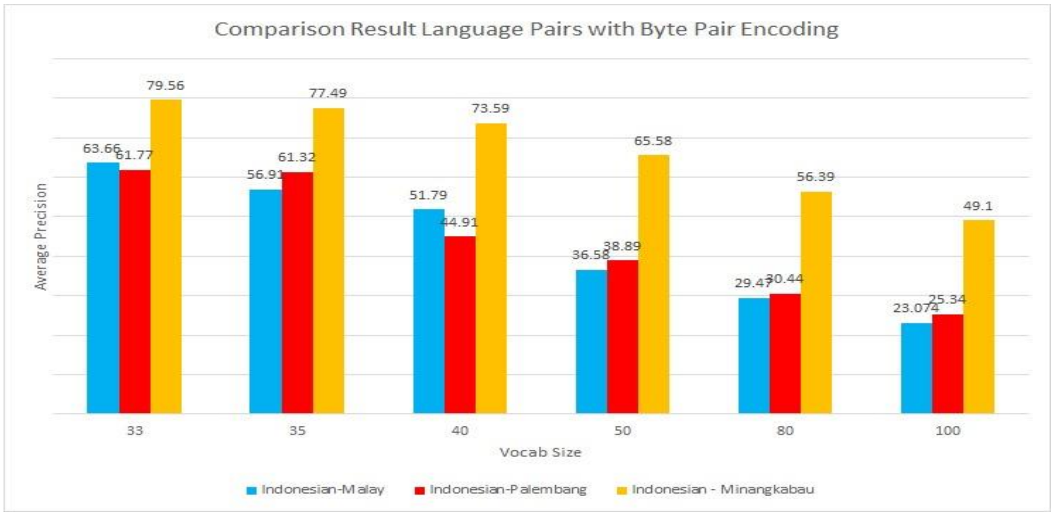


Figure 5.4: Comparison Language Pairs Result by BPE

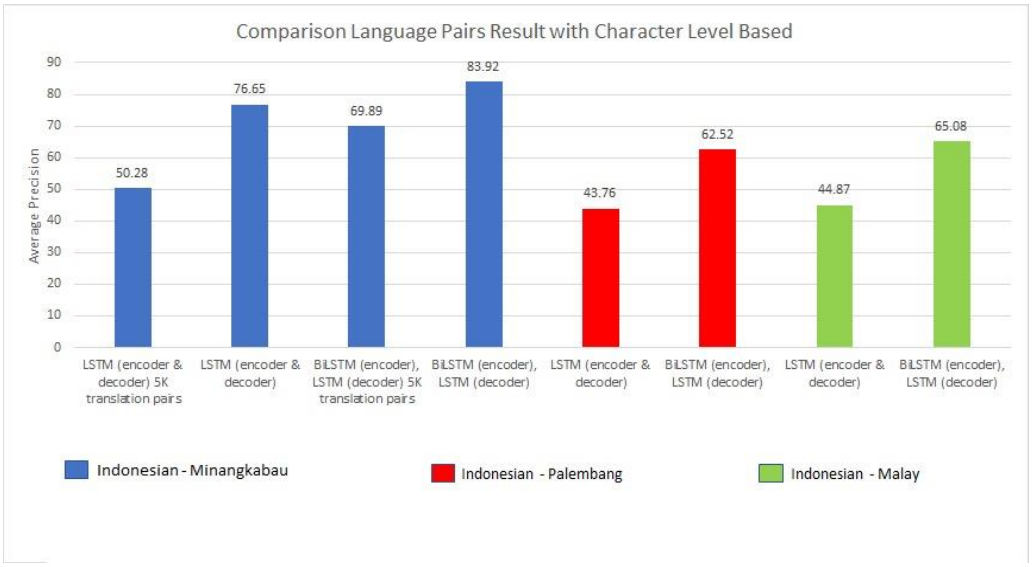


Figure 5.5: Comparison Language Pairs Result by Character level Based

In additional experiment, we also used the encoder settings from Indonesian-Minangkabau with roughly ten thousand translation pairings as initial values for other encoder in different language pairs (Indonesia-Palembang and Indonesia-Minangkabau). When define an input sequence , add the model that has been generated from Indonesian-Minangkabau character level Bi-LSTM Seq2Seq experiment.

```
model_minang=keras.models.load_model(currwd_minang+f"seq2seq_{i}.h5")
encoder_inputs = model_minang.input
```

Table 5.9: Evaluation of character-level model Indonesian-Palembang using Initial Encoder

Method	K-Fold Cross-Validation					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Averag e**
BiLSTM (encoder), LSTM (decoder)	**64.41**	63.03	63.13	61.56	62.94	63.01

Table 5.9.1 : Evaluation of character-level model Indonesian-Malay using Initial Encoder

Method	K-Fold Cross-Validation					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Averag e**
BiLSTM (encoder), LSTM (decoder)	61.66	63.00	63.67	64.53	**65.77**	63.72

Figure 5.6 compares the average precision results between Indonesian-Palembang and Indonesian-Malay using the intial encoder model from Indonesian-Minangkabau. In both language pairs, we are using half the data size. The total number of translation pairs is 5089 for Indonesian-Palembang and 5229 for Indonesian-Malay. According on the encoder-reuse model's performance, reusing the existing encoder that has been trained using several language translations pairs (high data size) seems to be beneficial.

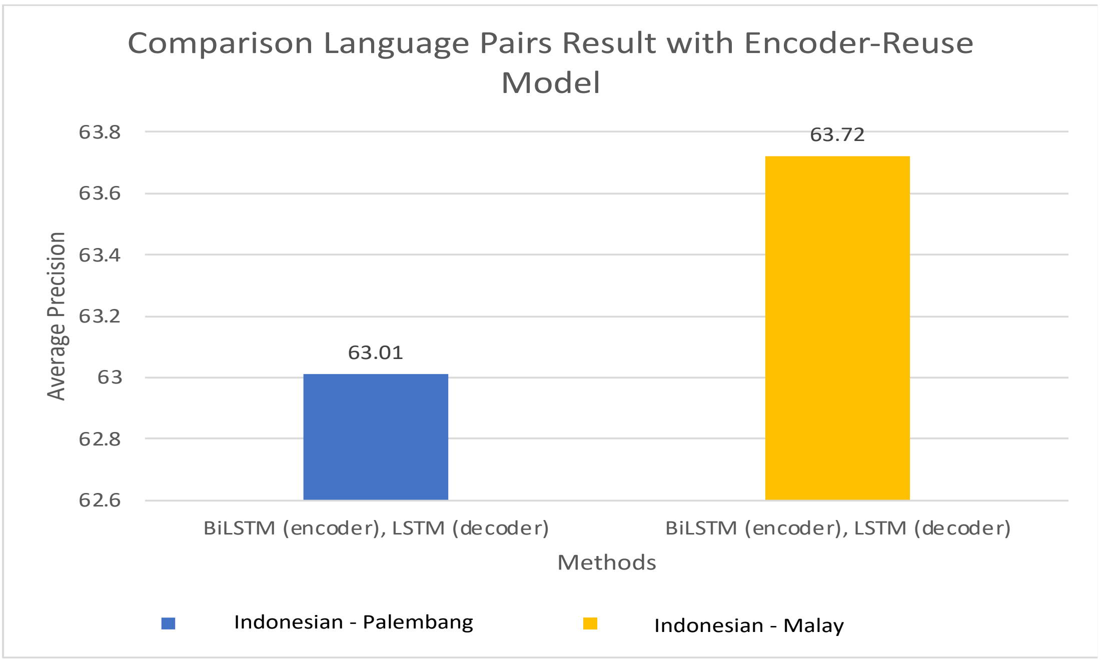


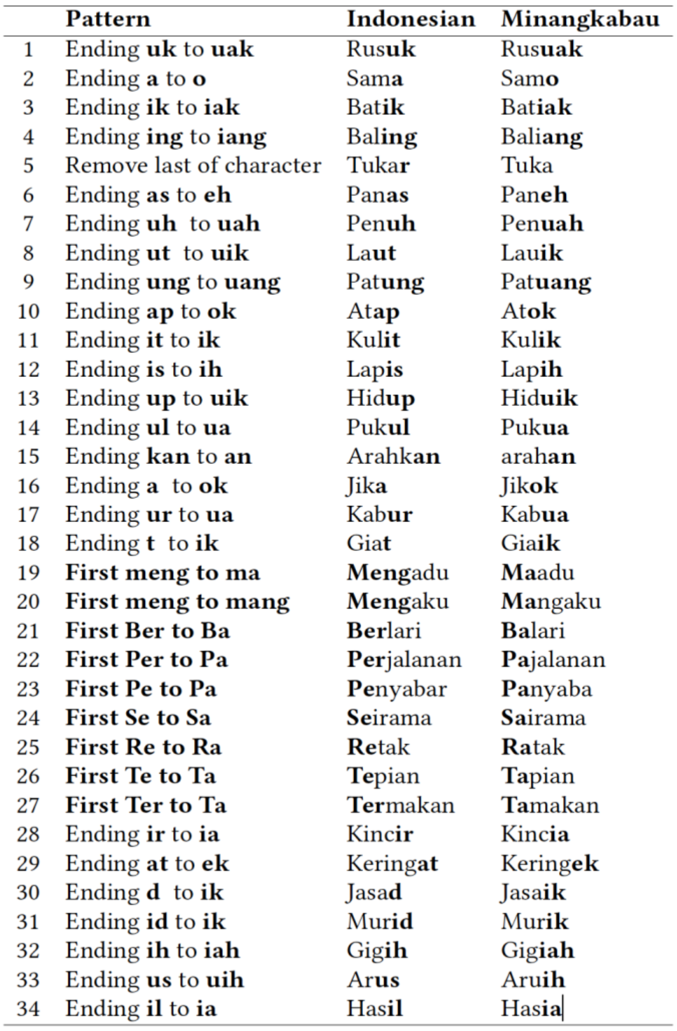
Figure 5.6: Comparison Language Pairs Result with Encoder

5.6 Pattern-based Precision

Simple rule-based performance evaluation has been done to compare with neural network performance result (Character level Bi-LSTM), whether the neural network-based performs efficiently or not. The rule-based approach is not our proposed. For comparing our model, use that as the baseline. Minang speakers provided the patterns used in the Minangkabau and Indonesian languages. Step to generate translation using rule-based are as follows:

1. Remove the similar word translation pairs from the 2056 translation pairings. The remaining data, after the similar translation pairs are removed, is 1262.
2. Determine the rules manually by regular expression.
3. Use the rules that have been determined and apply to all source words and replaces rule matches with a string.

Below is the figure of patterns in the Indonesian-Minangkabau language:



	Pattern	Indonesian	Minangkabau
1	Ending **uk** to **uak**	Rus**uk**	Rus**uak**
2	Ending **a** to **o**	Sam**a**	Sam**o**
3	Ending **ik** to **iak**	Bat**ik**	Bat**iak**
4	Ending **ing** to **iang**	Bal**ing**	Bal**iang**
5	Remove last of character	Tuka**r**	Tuka
6	Ending **as** to **eh**	Pan**as**	Pan**eh**
7	Ending **uh** to **uah**	Pen**uh**	Pen**uah**
8	Ending **ut** to **uik**	La**ut**	La**uik**
9	Ending **ung** to **uang**	Pat**ung**	Pat**uang**
10	Ending **ap** to **ok**	At**ap**	At**ok**
11	Ending **it** to **ik**	Kul**it**	Kul**ik**
12	Ending **is** to **ih**	Lap**is**	Lap**ih**
13	Ending **up** to **uik**	Hid**up**	Hid**uik**
14	Ending **ul** to **ua**	Puk**ul**	Puk**ua**
15	Ending **kan** to **an**	Arah**kan**	arah**an**
16	Ending **a** to **ok**	Jik**a**	Jik**ok**
17	Ending **ur** to **ua**	Kab**ur**	Kab**ua**
18	Ending **t** to **ik**	Gia**t**	Gia**ik**
19	**First meng to ma**	**Meng**adu	**Ma**adu
20	**First meng to mang**	**Meng**aku	**Ma**ngaku
21	**First Ber to Ba**	**Ber**lari	**Ba**lari
22	**First Per to Pa**	**Per**jalanan	**Pa**jalanan
23	**First Pe to Pa**	**Pe**nyabar	**Pa**nyaba
24	**First Se to Sa**	**Se**irama	**Sa**irama
25	**First Re to Ra**	**Re**tak	**Ra**tak
26	**First Te to Ta**	**Te**pian	**Ta**pian
27	**First Ter to Ta**	**Ter**makan	**Ta**makan
28	Ending **ir** to **ia**	Kinc**ir**	Kinc**ia**
29	Ending **at** to **ek**	Kering**at**	Kering**ek**
30	Ending **d** to **ik**	Jasa**d**	Jasa**ik**
31	Ending **id** to **ik**	Mur**id**	Mur**ik**
32	Ending **ih** to **iah**	Gig**ih**	Gig**iah**
33	Ending **us** to **uih**	Ar**us**	Ar**uih**
34	Ending **il** to **ia**	Has**il**	Has**ia**

Figure 5.9.1: List of Patterns in Indonesian and Minangkabau

There are 34 patterns, except based on the above pattern number 19-20 and 21-27 in regular expression are converted into a group by changing all first characters before “e” to “a” and all first characters before "er" to “a”. The group can be created using the regular expressions (’([aiueo]*) e’) and (’([aiueo]*) er’). Also converted into a group by changing all first character before “eng” to “a” and all characters before “eng” to “ang” in regular expression becomes ([^aiueo]*)eng ang and ([^aiueo]*)eng a.

Table 5.9.2: The Number of Comparisons Between Neural Network Approach and Rule-Based

Comparison The Possibilities of Both Method		
Neural Network Approach (True)	Rule Based (True)	374
Neural Network Approach (True)	Rule Based (False)	643
Neural Network Approach (False)	Rule Based (True)	59
Neural Network Approach (False)	Rule Based (False)	186
Total Translation Pairs		1262

The table above shows the comparison result of simple rule based and neural network based. We compare it by examining some of the potential outcomes from data testing. According to the Table 5.9.2 the outcome of accurate translation in terms of the target language is true, meanwhile false is a translation inaccurate that is in the target language. The performance according to each pattern by comparing with the simple rule-based system is described in Table 5.9.3.

Table 5.9.3. Compare performance with Simple-Rule Based

Pattern	**Match Rule Based**	**Match Neural Network**	**Total of All Words**	**Precision Rule Based**	**Precision Neural Network**
First all character before eng to ang	19	72	89	0.21	0.81
First all character before eng to all character before a	0	0	1	0	0

Pattern	**Match Rule Based**	**Match Neural Network**	**Total of All Words**	**Precision Rule Based**	**Precision Neural Network**
First all character before er to befor a	94	242	291	0.32	0.83
First all character before e to before a	184	422	517	0.36	0.82
Ending uk to uak	11	14	15	0.73	0.93
Ending a to o	98	203	259	0.38	0.78
Ending ik to iak	3	11	11	0.27	1
Ending ing to iang	15	20	22	0.68	0.91
Remove last of character	13	50	63	0.21	0.79
Ending as to eh	6	15	18	0.33	0.83
Ending uh to uah	10	11	13	0.77	0.85
Ending uk to uik	6	9	10	0.6	0.9
Ending ung to uang	17	24	24	0.71	1
Ending ap to ok	6	10	13	0.46	0.77
Ending at to aik	22	54	69	0.32	0.78
Ending at to ek	0	0	0	0	0
Ending ir to ia	0	0	0	0	0
Ending ur to ua	0	0	0	0	0
Ending kan to an	30	139	164	0.18	0.85

Pattern	**Match Rule Based**	**Match Neural Network**	**Total of All Words**	**Precision Rule Based**	**Precision Neural Network**
Ending it to ik	4	5	6	0.67	0.83
Ending it to iak	0	0	0	0	0
Ending is to ih	0	11	12	0	0.92
Ending ul to ua	4	3	4	1	0.75
Ending d to ik	1	5	6	0.17	0.83
Ending id to ik	0	0	0	0	0
Ending ih to iah	6	20	22	0.27	0.91
Ending us to uih	4	12	16	0.25	0.75
Ending il to ia	3	4	5	0.6	0.8
Total				0.34	0.66

Based on table 5.9.2 compared to simple rule-based performance, neural network approach is superior since it allows for the generation of output translations with a variety of patterns. Simple rule based could not distinguish the patterns according to the position of index rule that has been determined. Example the words of *darat* to *darek* , even though we define the rule = ending at to ek but based on index rule, the rule "ending at to aik "is in index position before rule at to ek. Therefore, the result in that rule is not found.

Using half data size, and even with a total data set of 2.000-Indonesia-Minangkabau word pairings, comparison studies between Rule-based as baseline and Neural Network models were also conducted. The steps are the same as in the 10.000 translation pair experiment. The only difference on 5000 translation pair data is this step, which remove the similar word translation pairs from the 1046 translation pairings. The remaining data, after the similar translation pairs are removed, is 641. Below is the table result of comparisons between Neural Network and Rule-based using half data size translation pairs.

Table 5.9.4: Neural Network Approach and Rule-Based 5000 Translation Pairs

Comparison The Possibilities of Both Method		
Neural Network Approach (True)	Rule Based (True)	179
Neural Network Approach (True)	Rule Based (False)	267
Neural Network Approach (False)	Rule Based (True)	53
Neural Network Approach (False)	Rule Based (False)	142
Total Translation Pairs		641

Based on Table 5.9.4 Neural Network approach still outperforms the Rule-based with 267 pairs of words that are correctly translated. The performance according to each pattern is shown in Table 5.9.5.

Table 5.9.5: Compare performance with Simple-Rule Based

	Match RB	**Match NN**	**Total**	**Precision RB**	**Precision NN**
First all character before eng to ang	13	16	25	0.52	0.64
First all character before eng to all character before a	0	0	0	0	0
First all character before er to befor a	18	36	56	0.32	0.64
First all character before e to before a	60	88	137	0.44	0.64

Ending uk to uak	2	3	3	0.67	1
Ending a to o	23	43	66	0.35	0.65
Ending ik to iak	1	3	4	0.25	0.75
Ending ing to iang	4	5	6	0.67	0.83
Remove last of character	2	8	11	0.18	0.73
Ending as to eh	2	4	7	0.29	0.57
Ending uh to uah	4	5	5	0.8	1
Ending uk to uik	0	0	1	0	0
Ending ung to uang	5	3	6	0.83	0.5
Ending ap to ok	0	1	1	0	1
Ending at to aik	1	6	6	0.17	1
Ending at to ek	0	0	0	0	0
Ending ir to ia	0	0	0	0	0
Ending ur to ua	0	0	0	0	0
Ending kan to an	14	23	35	0.4	0.66
Ending it to ik	1	3	3	0.33	1
Ending it to iak	0	0	0	0	0
Ending is to ih	0	3	4	0	0.75
Ending ul to ua	1	2	3	0.33	0.67
Ending d to ik	0	0	0	0	0
Ending id to ik	0	0	0	0	0
Ending ih to iah	4	6	8	0.5	0.75

Ending us to uih	2	2	4	0.5	0.5
Ending il to ia	0	1	2	0	0.5

For the index rule sequence created based on the same regular expression as the experiment with around 10.000 translation pairs. There are fewer patterns that do not have translation pairs because of the amount of the data used.

We experimented with a smaller sample size of 2569 Indonesian-Minangkabau translation pairings. They are divided into 2055 training data and 514 testing data. The average outcome utilizing character level Bi-LSTM and K Fold Validation is shown below.

Table 5.9.6: Result of Character Level Model in Indonesian-Minangkabau 2.000 Data Size

Method	**K-Fold Cross-Validation**					
	K = 1	**K = 2**	**K = 3**	**K = 4**	**K = 5**	**Average**
BiLSTM (encoder), LSTM (decoder)	55.44	57.78	64.20	64.00	**66.34**	61.55

The first step to compare with Rule-based is remove the similar word translation pairs from the 514 translation pairings. After similar translation pairings are removed, 312 pairings are left. Then the next step is similar to the experiment above. Comparison results of simple rule-based and neural network based is shown in table 5.9.7 and the performance according to each pattern by comparing with the simple rule-based system is described in Table 5.9.8.

Table 5.9.7: Neural Network Approach and Rule-Based 2.000 Translation Pairs

Comparison The Possibilities of Both Method		
Neural Network Approach (True)	Rule Based (True)	87
Neural Network Approach (True)	Rule Based (False)	107
Neural Network Approach (False)	Rule Based (True)	36
Neural Network Approach (False)	Rule Based (False)	82
Total Translation Pairs		312

Table 5.9.8: Neural Network Approach and Rule-Based 2.000 Translation Pairs

	Match RB	**Match NN**	**Total**	**Precision RB**	**Precision NN**
First all character before eng to ang	13	16	25	0.52	0.64
First all character before eng to all character before a	0	0	0	0	0
First all character before er to befor a	18	36	56	0.32	0.64
First all character before e to before a	60	88	137	0.44	0.64
Ending uk to uak	2	3	3	0.67	1
Ending a to o	23	43	66	0.35	0.65
Ending ik to iak	1	3	4	0.25	0.75
Ending ing to iang	4	5	6	0.67	0.83

Remove last of character	2	8	11	0.18	0.73
Ending as to eh	2	4	7	0.29	0.57
Ending uh to uah	4	5	5	0.8	1
Ending uk to uik	0	0	1	0	0
Ending ung to uang	5	3	6	0.83	0.5
Ending ap to ok	0	1	1	0	1
Ending at to aik	1	6	6	0.17	1
Ending at to ek	0	0	0	0	0
Ending ir to ia	0	0	0	0	0
Ending ur to ua	0	0	0	0	0
Ending kan to an	14	23	35	0.4	0.66
Ending it to ik	1	3	3	0.33	1
Ending it to iak	0	0	0	0	0
Ending is to ih	0	3	4	0	0.75
Ending ul to ua	1	2	3	0.33	0.67
Ending d to ik	0	0	0	0	0
Ending id to ik	0	0	0	0	0
Ending ih to iah	4	6	8	0.5	0.75
Ending us to uih	2	2	4	0.5	0.5
Ending il to ia	0	1	2	0	0.5

According to all experiments, the neural network approach result outperforms the simple rule-based. As is well known that the rule-based approach is unable to distinguish the position of the index rule. Additionally, we do not use a specific strategy in this experiment to identify where the index rule created by regular expression should be placed. Therefore, that it also has an impact on rule-based performance. This experiment section is to show well the proposed model can reproduce the well-known pattern compared to the simple rules based as baseline.

Chapter 6 Conclusion

In case to translate word to word pairs, it can be argued that the neural network approach utilizing a sequence-to-sequence model is better able to extract Indonesian-Minangkabau language patterns with a distinct number of tokens based on character basis. Character level seq2seq method (Bi-LSTM as encoder and LSTM as decoder) outperforms SentencePiece byte pair encoding (vocab size of 33) according to a comparison of the two methods utilized, which has an average precision of 79.56% compared to 83.92% for character level seq2seq method. Lack of training data was one of the obstacles in the experiment to create an Indonesian-Palembang dictionary, the best outcomes were obtained from the Character level seq2seq method (Bi-LSTM as encoder and LSTM as decoder) with an average precision of 62.52%. The best setting (character level embedding with the Bi-LSTM as encoder and LSTM as decoder) shows a good result for four other Indonesian ethnic languages even with about half size of input dictionaries (Palembang, Malay) with the average precision of 65.2% and 65.08%, respectively. The performance of the encoder-reuse model it seems useful to reuse the exiting encoder trained by different language translation pairs. The result of average precision in Indonesian-Palembang, and Indonesian-Minangkabau using encoder-reuse model is 63.01% and 63.72%. In the future, we would like to apply this method to other Indonesian ethnic languages that might not only demand the same pattern.

Additionally, compared to simple rule-based performance, neural network methodology is superior since it allows for the generation of output translations with a variety of patterns.

Acknowledgments

Firstly, I would like to give my acknowledgement to my supervisor, Prof Yohei Murakami for giving me a chance to study under his guidance. This research would have not been possible without his guidance and support. I am really grateful to study under such an inspiring person. Without him, I would never reached this far. My thanks go the members of Social Intelligent Laboratory, especially Ampere, Nishimura , Motozawa, Te, Li , and the others for their assistance and support in helping me to comprehend the research done by the member labs. I'm thankful to Allah SWT for everything, because without him, I couldn't have completed my study. I thank my family, for the love, supports, and prayers. I would never reached this far without all of their supports.

I would like to thank all of my Indonesian friends in Japan, especially Fira and Suryadi, as well as Ema, Aisyah, Teh Chami, Andri, Opi, Abi, Marcel, and all of the dorm students, for all of their support. Even though they do not stay in Japan, I also want to thank my Indonesian friends Rani, Farah, Ruen, Shofi, Pi, Haqi, Ghufron, Uul, Nisa, Zakia, Milah, Dima, Ana, Venia, and Dinda for their support, especially thanks to Ali in helping me to do this research. Last but not least, I would like to thank myself for struggling and surviving on campus, and for Japan’s life.

References

[1] A.R. Baidalina, and S.A. Boranbayev and. 2021. PROGRAMMING DATA STRUCTURE ALGORITHMS IN PYTHON. 73, 1 (mar 2021), 134–141.

[2] Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2018. A deep learning approach to bilingual lexicon induction in the biomedical domain. BMC Bioinformatics 19, 1 (jul 2018).

[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Computation 9, 8 (nov 1997), 1735–1780.

[4] Fajri Koto and Ikhwan Koto. 2020. Towards Computational Linguistics in Minangkabau Language: Studies on Sentiment Analysis and Machine Translation. In Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation. 138–148.

[5] Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics.

[6] Yohei Murakami. 2019. Indonesia Language Sphere: an ecosystem for dictionary development for low-resource languages. Journal of Physics: Conference Series 1192 (mar 2019), 012001.

[7] Arbi Haza Nasution, Yohei Murakami, and Toru Ishida. 2016. Constraint-based bilingual lexicon induction for closely related languages. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). 3291–3298.

[8] Arbi Haza Nasution, Yohei Murakami, and Toru Ishida. 2017. A generalized constraint approach to bilingual dictionary induction for low-resource language families. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 17, 2 (2017), 1–29.

[9] Arbi Haza Nasution, Yohei Murakami, and Toru Ishida. 2017. Plan optimization for creating bilingual dictionaries of low resource languages. In 2017 International Conference on Culture and Computing (Culture and Computing). IEEE, 35–41.

[10] Arbi Haza Nasution, Yohei Murakami, and Toru Ishida. 2019. Generating similarity cluster of Indonesian languages with semi-supervised clustering. International Journal of Electrical and Computer Engineering (IJECE) 9, 1 (feb 2019), 531.

[11] Arbi Haza Nasution, Yohei Murakami, and Toru Ishida. 2021. Plan Optimization to Bilingual Dictionary Induction for Low Resource Language Families. Transactions on Asian and Low-Resource Language Information Processing 20, 2 (2021), 1–28.
[12] Scott Paauw. 2009. One Land, One Nation, One Language: An Analysis of Indonesia's National Language Policy. University of Rochester working papers in the language sciences 5, 1 (2009).
[13] M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45, 11 (1997), 2673–2681.
[14] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of RareWords with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics.
[15] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. Advances in neural information processing systems 27 (2014).
[16] Intan Nurma Yulita, Mohamad Ivan Fanany, and Aniati Murni Arymuthy. 2017. Bi-directional Long Short-Term Memory using Quantized data of Deep Belief Networks for Sleep Stage Classification. Procedia Computer Science 116 (2017), 530–538.
[17] Haijun Zhang, Jingxuan Li, Yuzhu Ji, and Heng Yue. 2016. A character-level sequence-to-sequence method for subtitle learning. In 2016 -IEEE 14th International Conference on Industrial Informatics (INDIN). IEEE.
[18] Kartika Resiandi, Yohei Murakami, and Arbi Haza Nasution. 2022. A Neural Network Approach to Create Minangkabau-Indonesia Bilingual Dictionary. Proceedings of SIGUL2022@LREC2022. Pages 122–128.
[19] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben- gio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
[20] Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2016. Neural Machine Translation with Recurrent Attention Modeling. arXiv preprint arXiv:1607.05.108.
[21] Y.M. Wazery, Marwa E. Saleh, Abdullah Alharbi, and Abdelmgeid A. Ali. Abstractive Arabic Text Summarization Based on Deep Learning. Hindawi. Computational Intelligence and Neuroscience. Volume 2022. Article ID 1566890.