IT Journal Research and Development (ITJRD)
Vol. 9, No. 1, August 2024, pp. 14 – 26, E-ISSN: 2528-4053 |P-ISSN: 2528-4061
DOI : 10.25299/itjrd.2024.14171

Automatic Vocal Completion for Indonesian Language Based on Recurrent Neural Network

Agi Prasetiadi[1], Asti Dwi Sripamuji[2], Risa Riski Amalia[3], Julian Saputra[4], and Imada Ramadhanti[5]
Faculty of Informatics, Institut Teknologi Telkom Purwokerto, Purwokerto, Indonesia[1,2,3,4,5]
agi@ittelkom-pwt.ac.id[1], 19102006@ittelkom-pwt.ac.id[2], 19102079@ittelkom-pwt.ac.id[3],
19102008@ittelkom-pwt.ac.id[4], 19102003@ittelkom-pwt.ac.id[5]

Article Info

Article history:
Received Aug 18, 2023
Revised Nov 27, 2023
Accepted Jul 17, 2024

Keyword:
Slang
Recurrent Neural Network
Gated Recurrent Unit
Long Short-Term Memory
Bidirectional

ABSTRACT

Most Indonesian social media users under the age of 25 use various words, which are now often referred to as slang, including abbreviations in communicating. Not only causes, but this variation also poses challenges for the natural language processing of Indonesian. The previous researchers tried to improve the Recurrent Neural Network to correct errors at the character level with an accuracy of 83.76%. This study aims to normalize abbreviated words in Indonesian into complete words using a Recurrent Neural Network in the form of Bidirected Long Short-Term Memory and Gated Recurrent Unit. The dataset is built with several weight configurations from 3-Gram to 6-Gram consisting of words without vowels and complete words with vowels. Our model is the first model in the world that tries to find incomplete Indonesian words, which eventually become fully lettered sentences with an accuracy of 97.44%.

Corresponding Author:
Agi Prasetiadi
Faculty of Informatics
Institut Teknologi Telkom Purwokerto
Central Java, Indonesia, 53147
Email: agi@ittelkom-pwt.ac.id

1. INTRODUCTION

Language is a communication medium to convey ideas or thoughts from one person to another. Bahasa Indonesian is the national language standard of the Indonesian people in the daily speaking environment [1]. Along with the development of society in communication, they change the way of delivery through social media into a new language or what we usually call slang [2]. The reason for the emergence of slang on social media is the lack of enthusiasm to use Indonesian in its entirety, giving a formal impression and causing awkwardness among users under the age of 25 [3].

The use of slang has resulted in the creation of new words depending on the user, such as variations in typing [4], excessive writing of, completely new words [5], conjugation between two or more words, or even words that abbreviated [6][7]. This often creates gaps and misunderstandings in communication [8][9]. Furthermore, it poses challenges at the preprocessing stage of Indonesian text in natural language

processing [10]. The resulting words are isolated from other communities due to increased vocabulary but low references. The Term Frequency — Inverse Document Frequency score is high, meaning the word has high priority [11]. However, this is not true because it could be just as supporting words in a sentence. Word mapping can use Word2Vec [12] or FastText [13] to find out how similar it is to other words, but that will only add unnecessary vocabulary and memory during processing.

In this study, we focus on normalizing abbreviated words that occur in Indonesian, especially words without vowels, because they are very ambiguous. Because, without the abbreviated words, homograph words can create ambiguity for the reader if we do not read the context words. For example, a sentence like ”Saya punya apel” could refer to apples or a morning routine where employees gather in the field for a brief pre-work briefing. Furthermore, suppose we shorten this sentence to ”sy pny pl”. In that case, it will become even more confusing because it can mean ”Saya punya palu”, ”Saya punya apel”, ”Saya punya pala”, ”Saya punya pel”, etc.

The following are previous studies that are relevant to this research. Suppose we can see the abbreviated words as typos. In that case, there are several techniques that the previous researchers tried to improve, such as using Long Short-Term Memory (LSTM) to correct errors at the character level with an accuracy of 83.76% [14], uses Levenshtein-Damerau for typos in Indonesian with 75% accuracy [15]. Meanwhile, Recurrent Neural Network (RNN) can be used to translate language from one to another, such as translating English phrases into Indonesian with an accuracy of 88.57% [16], identifies Arabic script with a gray level Co-occurrence matrix which shows an accuracy of 78.75% [17], identifies automatically named entities in Indonesian to get the best results F-size 0.72 [18], and uses RNN for automatic sentence summary [19].

To our best knowledge, our research is the first in the world that uses RNN to change Indonesian sentences without vowels into sentences with complete words with vowels by implementing the model through several datasets and different architectures. This paper is structured as follows. Part I as an introduction. Part II discusses the design of RNN models used in this paper, which are based on LSTM and GRU, with Bidirectional variation. Section III discusses the accuracy of the algorithm model. Lastly, section IV concludes the performance of the RNN algorithm model.

2. METHOD

This research has two phases for building the model: processing the dataset and designing the architecture. The dataset is prepared in various ways that each version can fit into the designed architecture. Four primary datasets are prepared based on the Indonesian News Dataset [20], where each is modified based on its N-Gram value. After the dataset is ready to be used, the model architectures are prepared to train the dataset. Each dataset has six different models, each divided based on RNN gate cells and the number of layers. Since we had four datasets, 24 models are being tested in this research.

2.1. Dataset Preprocessing

The dataset is obtained from collecting Indonesian news with five categories labeled inside [20]. These labels are soccer, news, business, technology, and automotive. The dataset comprises 6.127 news from various Indonesian news outlets, whereas the most comprehensive article consists of 8.973 characters.

However, this Indonesian news dataset is transformed into a different dataset suitable for our research. The categories are ignored, then all the contents are joined into single long sentences. This lengthy sentence is further preprocessed into five different datasets depending on the N-gram configuration words per sentence. The configurations are 2-gram, 3-gram, 4-gram, 5-gram, and 6-gram.

N-gram is a model used to solve word ambiguity [21]. An example of the use of n-gram when there is the word ”bisa” which contains various meanings. The first meaning of ”bisa” is poison, a noun such as in the sentence of ”ular mengeluarkan bisa”. The other meaning of ”bisa” is able, an adjective word such as

in the sentence ”manusia bisa duduk”. Thus, to solve the ambiguity of the word, n-gram models are needed.

For example, let us assume that S is the sentence that will be converted into a dataset with a 3-gram configuration where S = ”hari ini saya pergi ke pasar bersama dengan ibu dan ayah . . . ”. Since this research aims to reconstruct non-vocalized sentences into vocalized sentences, the sentence S is split into X and Y, where X is 3-gram words without vowels and Y is 3-gram words with vowels, resulting in X = ”hr n sy”, ”n sy prg”, ”sy prg k”, ”...” and Y = ”hari ini saya”, ”ini saya pergi”, ”saya pergi ke”, ”...”. The dataset example can be seen in Table 1.

To construct these datasets, we initially convert all datasets' contents to lowercase. Then, we employ regular expressions to derive dataset Y by filtering out any alphabetical characters and spaces using the regex pattern `[a-zA-Z\ ]`. From dataset Y, we then apply regular expressions once more to eliminate any vowels, retaining only consonants through the removal of the pattern `[^aeiouAEIOU]`.

Table 1. The Preprocessed Datasets

Dataset	X	Y
3-Gram	hr n sy	hari ini saya
	brsm b d	bersama ibu dan
4-Gram	hr n sy prg	hari ini saya pergi
	brsm b d yh	bersama ibu dan ayah
5-Gram	hr n sy prg k	hari ini saya pergi ke
	brsm b d yh sng	bersama ibu dan ayah siang
6-Gram	hr n sy prg k psr	hari ini saya pergi ke pasar
	brsm b d yh sng sng	bersama ibu dan ayah siang siang

Each sentence inside the dataset will be further preprocessed by splitting them into characters. The sentence will be tokenized based on its character, not the word. This is crucial because we would like the model to understand the relationship between two sentences with more creativity in filling the vocal gaps, not only based on word vocabulary. This will give a better advantage when the model sees an unseen word before so that the model can guess the new word's vocal without losing too much accuracy.

After tokenizing the sentence into the number vector, each vector will be normalized to the same length. In other words, the vector is padded with 0 until all the vectors have the same length. The length is determined as $\max \bar{S}$, the longest sentence in terms of characters within the dataset. Then, we shuffle the dataset so that the model can learn the pattern in a more distributed way.

2.2. RNN

Recurrent Neural Network is a machine learning method based on neural network learning that resembles the human brain system in capturing patterns. RNN is one of the neural network methods that can be used. Unlike other neural network methods, RNN is famous for its ability to form predictions based on previous predictions [22].

Recurrent neural network is a sequential network in which each output generated is derived from the most recent input and the previous output.This network performs poorly on data with large sequence lengths. However, the performance shown is good for sequence data with a length of 3-5 only [23]. This sequential data has a dependence on one another as input in the form of words that make up sentences or collections of sentences that make up documents to get the actual context. Classifying time series and sequential data is characteristic of the RNN. Own time series data is data that is combined according to the order of time within a certain range, while sequential data is a sample of data that is processed sequentially and each sequence relates to each other [24].

Based on architectural diagram that shows the RNN process in a position that is not open to the full network. The symbol x_t is the input at each time step. The symbol s_t is the hidden state or memory

at each time step t. The hidden state can be alluded to as the memory of a network that can store the result of calculations and the records that have been done. The following is the formula for the function $s_t = f(Ux_t + Ws_{t-1})$. The symbol o_t is the output for each step t. Here's the formula for the function $o_t = softmax(Vs_t)$.

There are two kinds of RNN cell used in this paper, LSTM cell and GRU gate. LSTM is one kind of RNN that has short term and long term memory capabilities [25]. LSTM is able to execute large datasets but requires a long time in terms of execution of a data. LSTM has three gates for reading, storing, and updating information. The gate consists of input gates, forget gates, and output gates so that LSTM is suitable for forecasting cases such as wind forecasting [26], stock price prediction, [27][28] trajectory prediction, [29][30] and character-based text generalization [31].

The LSTM training process through gate functions sequentially from forget gates, input gates, and output gates is then continued with the iterative process as many epochs as determined by adam optimization, activation using softmax and training results in the form of HDF5 or .h5 files. In the LSTM network there are forget gates, input gates, output gates and memory cell that will calculate the output value as a hidden layer for the next network [32].

Inside LSTM, there are Forget Gate (f_t), Input Gate (i_t), and Output Gate (o_t). The Forget Gate has the output and input values combined to pass the function on sigmoid activation. In this process, it is determined whether the previously obtained information will be forgotten or not. Furthermore, the information will be forwarded to the cell state or memory cell [32][33]. The Forget Gate follows the following equation, $f_t = \sigma(W_f \times [x_t + h_{t-1}] + b_f)$. In this case, f_t is forget gate, W_f is weights forget gate, h_{t-1} is output cell previously, σ is sigmoid activation function, x_t is input cell and b_f is bias forget gate.

The Input Gate has the output and input values combined, which will then be through two activation functions. The first path will pass through the activation function sigmoid to enter the input value. The other path will pass through the function activate tanh to enter a candidate memory cell value [32][33]. The Input Gate has the following equation, $i_t = \sigma(W_i \times [x_t + h_{t-1}] + b_i)$ and $\hat{C}_t = \tanh(W_c \times [x_t + h_{t-1}] + b_c)$. In this case, i_t is input gate, $\hat{C}_t$ is candidate, W_c is weights candidate, W_i is weights input, $\tanh$ is tanh activation function, b_C is bias candidate and b_i is bias input gate.

Lastly there is Output Gate. In this gate, the combined result of the previous value with the value obtained through the sigmoid activation function will produce an output value of [32][33]. This gate used the following equation, $o_t = \sigma(W_o \times [x_t + h_{t-1}] + b_o)$. In this case, o_t is output gate, b_o is bias output gate and W_o is weight output gate.

Just like the name, LSTM has two kinds of memory, the hidden state and the cell state, each state is responsible for short term and long term memory respectively. Inside cell state, there are two values that are combined. The first value is obtained from the multiplication of the value of the forget gate and the value of the cell state. The second value is obtained from the multiplication of the input gate value and the candidate memory cell value [32][33]. The following equation is used in this gate, $C_t = i_t \times C_{t-1} + i_t \times \hat{C}_t$. In this case, C_t is cell state and C_{t-1} is cell state previously.

Finally, there is hidden layer which affects the value in the next process. The value of this layer is obtained from the multiplication of the output value and the value of the cell state or memory cell that has gone through the activation process in the tangent function [32][33]. We can denote the correlation with the following simple equation, $h_t = h_t \times (C_t)$. In this case, h_t is hidden layer.

Gated Recurrent Unit (GRU) is a variation of the RNN method which is designed to be a solution to the Vanishing Gradient problem that occurs in the RNN method. In the GRU architecture, there are two unit gates used, namely the Update Gate and the Reset Gate. These two unit gates are the result of the combination of the three unit gates of the reduced LSTM. Basically, these two unit gates are vectors that determine what kind of information will be passed as a result.

At the Update Gate is determined how much information or units previously stored to be carried

into the future [34]. In other words, $z_t = \sigma(x_t w_t + h_{t-1} U_z + b_z)$, where z_t as the symbol of the Update Gate with t as the time step. Then w_t is the weight symbol which will be multiplied by h_{t-1} (final memory) as the result of the previous time step and x_t is the input for the time step being processed. Then the two results will be added and entered into the sigmoid function (σ) [34].

The Reset Gate work r_t decides how much past data will be erased or overlooked. The formula for the Reset Gate is the same as the formula for the Update Gate. The difference lies in the weight of W_r, U_r, and also the use of the gate [34]. Thus, $r_t = \sigma(x_t w_r + h_{t-1} U_r + b_r)$.

In the Current Memory Content n_t formula there is a multiplication between w which is the result weight of the Reset Gate which is calculated using multiply the product of Hadamard (element-wise) with h_{t-1} and will also be multiplied by x_t. Then the result will be entered into the function tanh [34]. It can be denoted as $n_t = \tanh(w_h x_t + U_h(r_t \odot h_{t-1}) + b_n)$.

The last process is Final Memory h_t which is passed on to the next cell. The calculation $(1 - z_t)$ is multiplied by h_{t-1} or the previous final memory and then added with the result of the Update Gate z_t which has been multiplied by Current Memory Content or n_t [34]. It also can be denoted as $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot n_t$.

2.3. Layer Connection

Two crucial connection configuration needs to be addressed in our problem, Bidirectional and Time Distributed. The nature of our dataset is similar to the general language translation problem. The only key difference in our study is that the sentence is converted as the token of characters instead of the token of words. Since deciphering fully consonant texted into fully vocalized texted needs context, at least one word before and one word after, Bidirectional connection is needed to understand the context more clearly from two directions of reading, first to the last and last to the first.

A bidirectional Neural Network is one technique to combat vanishing gradient problems. Instead of letting the gradient value disappear during the training in the last cells, the network is set up so that information also flows from the last layers to the front cells. This technique is suitable for language translation where the target language has significantly different grammatical structures from the former language. Bi-RNN is the result of modifications to the RNN layer that is connected to the previous and next states. Here is the Bi-RNN formula: $h_t^{(f)} = g(W^{(j)T} + W^{(j)} h_{t-1}^{(j)} + b^{(j)})$ and $h_t^{(b)} = g(W^{(j)T} + W^{(j)} h_{t-1}^{(j)} + b^{(j)})$. In the j layer, the forward state and backward state will be calculated. Then the results of $h_t^{(f)}$ and $h_t^{(b)}$ are summed with the following formula: $h_t^{(j)} = h_t^{(f)} + h_t^{(b)}$ [35].

After that, the vector that the connection produces will be transferred into the Time Distributed Dense layer. Time Distributed is a fully connected layer that is given a challenging default class wrapper so that it will apply the fully connected layer at each time stage. The Time Distributed Layer is usually used in Recurrent Neural Networks to maintain one-to-one relations on input and output [36]. Using the Time Distributed layer will make the design and analysis for distributed optimization problems much more complex as they change over time [37]. This study uses a Time Distributed Layer followed by LSTM to capture sequential data by applying the duplicate layer to each data used. So using a time-distributed layer will not increase the complexity of a model, but the data will learn through several layers and produce a predictive result.

Lastly, we apply adam optimizer to our architecture. Adam's optimizer is a development of the momentum optimization algorithm and the Adagrad optimization algorithm, which can increase convergence speed in the early stages of training. Adam has the advantages of adaptive learning and relatively fast convergence compared to other optimizers. Therefore, the adam optimizer is one of the optimizers that is suitable in our case. Adam has methods such as dynamically adjusting the adaptive learning speed on each parameter by calculating the gradient's first and second moments of the gradient [38]. This is intended to make pa-

rameter updates more stable and not easy to enter into local optimization. In addition, adam can also record the average decay of the previous gradient, which is similar to momentum: $m_t = \beta_1 m_t - 1 + (1 - \beta_1) g_t$, $v_t = \beta_2 v_t - 1 + (1 - \beta_2) g_t^2$

Where m_t is the average value of the gradient index (mean) at time t, while v_t is the square of the gradient at time t. m_t and v_t are considered as 0s vectors when it is observed that they are biased towards zero, especially when the gradient decay rate is small. Next is to calculate the bias update, where $\hat{m}_t = m_t / 1 - \beta_1^t$ and $\hat{v}_t = v_t / 1 - \beta_2^t$. Then the parameters will be updated using the formula: $\theta_{t+1} = \theta_t - \eta / \hat{v}_t{}^{-1/2} - \varepsilon \odot \hat{m}_t$ [39][40].

2.4. Architecture

In this research, there 6 models that will be observed their performance when feeded with 4 different datasets, resulting in 24 different models. The models are divided into 2 kind of RNN cells, the LSTM based models and the GRU based models. Table 2 shows our 6 different models configuration. The value 64 represents the total count of LSTM or GRU cells employed in the model.

Table 2. The Models Configuration

Model	Layer(s)
A	1-layer LSTM (64)
B	2-layer LSTM (64, 64)
C	3-layer LSTM (64, 64, 64)
D	1-layer GRU (64)
E	2-layer GRU (64, 64)
F	3-layer GRU (64, 64, 64)

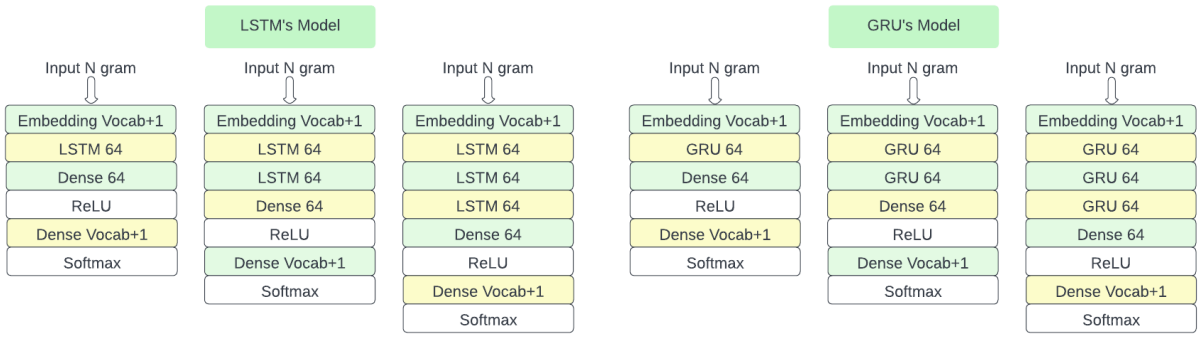
Since the dataset is consisted of series of tokenized characters, the input data is a 2-dimensional array. However since RNN received 3-dimensional data, the dataset needs to be processed with an additional layer to convert 2-dimensional input array into 3-dimensional data.

The embedding layer is the layer that receives 2-dimensional data. The first dimension deal with the number collection of the vectorized sentence, and the second dimension deal with the character's token. It works similarly with the Word2Vec [13] but in a neural network. Originally the vector-based word embedding is used to tackle the similar word in terms of meaning, projecting the word into an n-dimensional vector instead of a single value. However, this research will use this embedding layer to project characters into the 256-dimensional vector, giving them broader context and flexibility in describing ambiguous words through its vectorized characters. Figure 1 is the representation of the detailed model configuration. All models then will be run on six different datasets.

The flow of data entering the models is explained as follows. First, the X sentences, composed of only consonants, and Y sentences, composed of fully vocalized words, are tokenized based on their characters. Then, the tokens are padded into the same-length vector based on the longest tokens sequence inside the dataset. After that, the tokens are converted into vectors through the embedding layer. Then the data flows into designed RNN networks. Lastly, the data will flow into the 64-Dense layer before the predicted vocalized sentences are concluded in the last layers. The models used softmax activation on the output layer.

3. RESULTS AND DISCUSSION

All models are observed for their performance based on their training and validation accuracy. The models are trained within 30 epochs. During the training phase, the training and validation dataset is divided into 90% and 10% ratios, respectively. The experiments are done using python.



(a) (b)

Figure 1. The proposed architecture of (a) LSTM and (b) GRU based models

Since too many models are being tested, the training accuracy will be shown collectively based on their RNN cell type and the bias value. In this case, bias is defined as the difference between training accuracy and validation accuracy. Thus, the higher the accuracy, the better the model performs. At the same time, the lower the bias, the better the precision of the model gets.

3.1. 3-gram Dataset

Table 3 summarizes the overall models performance under 3-gram datasets. The accuracy column indicates the highest accuracy achieved by the model. The validation accuracy column indicates the validation accuracy of the model during the training. The average bias indicates the average difference between accuracy and validation accuracy.

Table 3. The model performance against 3-Gram dataset

Model	Accuracy	Val Accuracy	Average Bias
A: LSTM (64)	96.91%	96.72%	0.067%
B: LSTM (64, 64)	97.74%	97.42%	0.168%
C: LSTM (64, 64, 64)	98.11%	97.68%	0,203%
D: GRU (64)	96.68%	96.56%	0.06%
E: GRU (64, 64)	97.44%	97.2%	0.104%
F: GRU (64, 64, 64)	97.72%	97.47%	0.115%

It can be seen that the 3-layers LSTM (64, 64, 64) achieved 98.11% training accuracy. Despite the highest accuracy among other LSTM models, this model has the highest average bias of 0.203%. On the contrary, the 1-layer LSTM (64) model has the lowest average bias, 0.06%. However, the 1-layer LSTM (64) model has the lowest training accuracy, which is 96.91% training accuracy. It indicates that the model has better precision than other models. Based on the result, the most reasonable model is the 2-layer LSTM (64, 64) model, which has a maximum accuracy of 97.74% and an average bias of 0.168%. The performance of the LSTM models under 3-gram can be seen in Figure 2a for accuracy and in Figure 2b for bias.

The GRU-based models show similar patterns, just like the LSTM-based models. The deeper the layer, the more accurate the model becomes. First, the 3-layers GRU (64, 64, 64) achieved 97.72% training accuracy, the highest accuracy among GRU-based models. This model also has the highest average bias, 0.115%. Second, the 1-layer GRU (64) model has the lowest training accuracy, which is 96.86% training accuracy, but it has the most downward average bias, 0.06%. It also indicates that the model has better

precision than other models. Third, the most reasonable model is the 2-layers GRU (64, 64) model, which has a maximum accuracy of 97.4% and an average bias of 0.103%. The performance of the GRU models under 3-gram can be seen in Figure 2c for accuracy and in Figure 2d for bias.

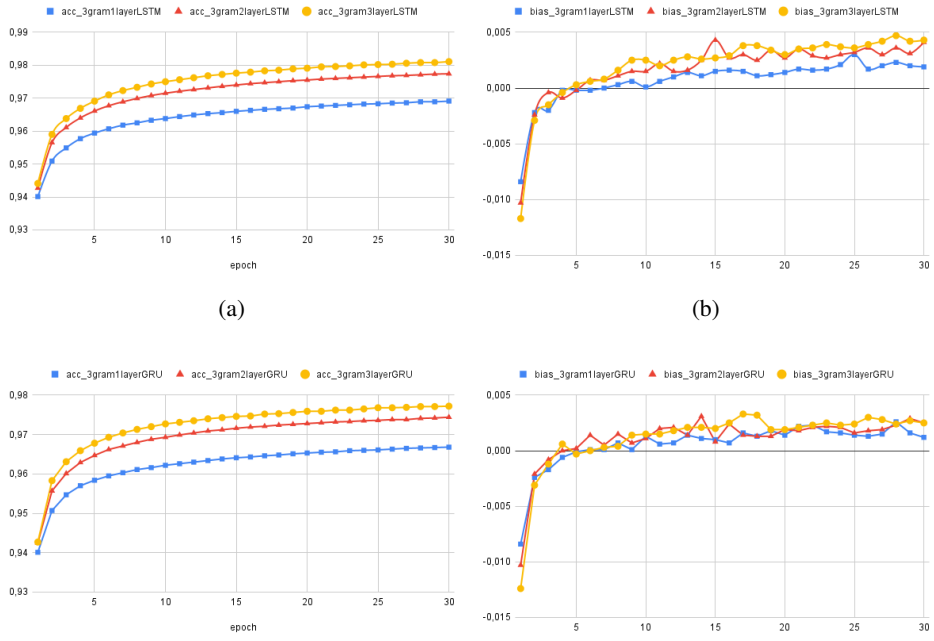


Figure 2. The overall performances for, (a) LSTM Models Accuracy, (b) LSTM Models Bias, (c) GRU Models Accuracy, (d) GRU Models Bias on 3-Gram Dataset

Thus, the highest accuracy is achieved by the 3-layer LSTM (64, 64, 64). The LSTM-based and GRU-based models show similar patterns, where the deeper the layer, the more accurate the model becomes. The GRU-based models exhibit lower average bias than the LSTM-based models.

3.2. 4-gram Dataset

The following Table 4 shows the overall performance of LSTM-based and GRU-based models in 4-gram datasets. At a glance, it can be seen that the LSTM shows similar results, just like the previous experiment. The highest accuracy is achieved by the 3-layer LSTM (64, 64, 64), which is 97.39%. But it doesn't show the same pattern as GRU-based models before.

Table 4. The model performance against 4-Gram dataset

Model	Accuracy	Val Accuracy	Average Bias
A: LSTM (64)	94.38%	94.22%	0.01%
B: LSTM (64, 64)	96.61%	96.15%	0.163%
C: LSTM (64, 64, 64)	97.39%	96.96%	0.197%
D: GRU (64)	96.61%	96.36%	0.087%
E: GRU (64, 64)	95.88%	95.66%	0.069%
F: GRU (64, 64, 64)	93.95%	93.84%	0.01%

The deeper the LSTM-based models, the better the accuracy gets. On the contrary, in this GRU-based models experiment, the deeper the layer, the worse the accuracy. But the GRU-based models also

exhibit lower average bias than the LSTM-based models. The performance of the LSTM models under 4-gram can be seen in Figure 3a for accuracy and in Figure 3b for bias. Furthermore, the performance of the GRU models under 4-gram can be seen in Figure 3c for accuracy and in Figure 3d for bias.

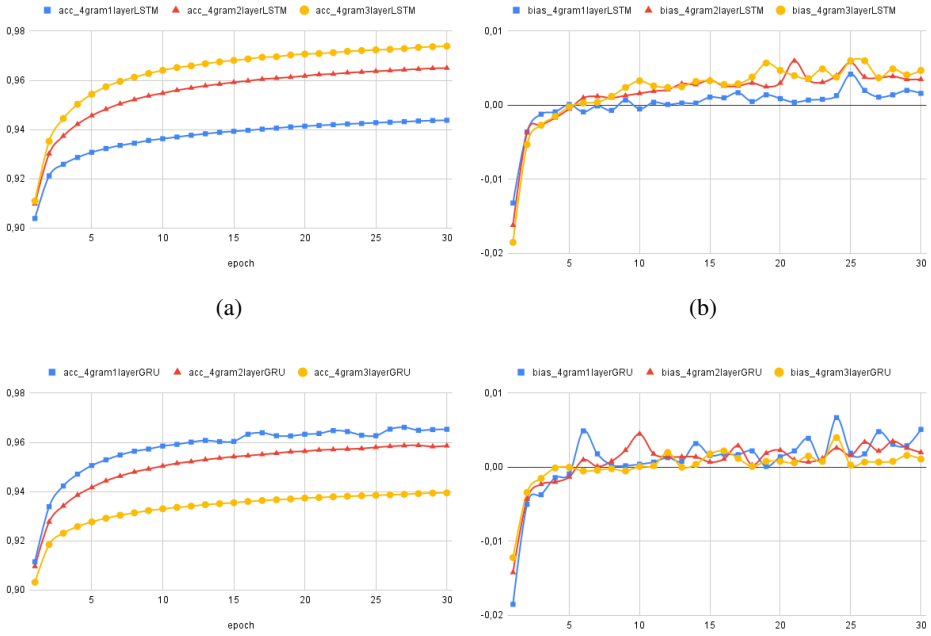


Figure 3. The overall performances for, (a) LSTM Models Accuracy, (b) LSTM Models Bias, (c) GRU Models Accuracy, (d) GRU Models Bias on 4-Gram Dataset

It is interesting to note that although the LSTM-based models show similar patterns to the previous, the GRU-based models didn't share the same pattern in terms of accuracy. Another finding is that all models trained under the 4-gram dataset obtained lowered maximum accuracy, 97.39%, than the 3-gram dataset, which has higher maximum accuracy, 98.11%. This might indicate that the Indonesian language didn't need a too long sentence to understand the context of the sentence.

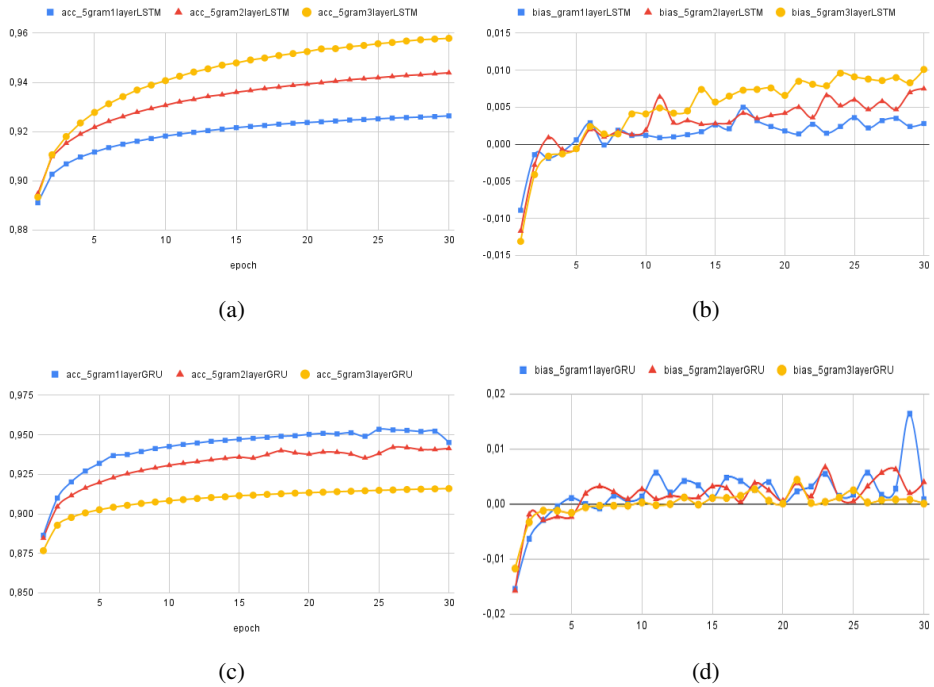
3.3. 5-gram Dataset

Table 5 shows the overall performance of LSTM-based and GRU-based models under a 5-gram dataset. The LSTM-based models again show the same pattern, just like the previous patterns. The deeper the layer, the more accurate the model becomes. The GRU-based models also exhibit the same pattern as the 4-gram dataset-based model. The lower the layer, the more accurate the model becomes. However, these GRU-based models did not share the same pattern as those trained with the 3-gram database models.

Table 5. The model performance against 5-Gram dataset

Model	Accuracy	Val Accuracy	Average Bias
A: LSTM (64)	92.64%	92.37%	0.141%
B: LSTM (64, 64)	94.39%	93.84%	0.286%
C: LSTM (64, 64, 64)	95.79%	94.93%	0.476%
D: GRU (64)	95.35%	95.18%	0.176%
E: GRU (64, 64)	94.21%	93.89%	0.129%
F: GRU (64, 64, 64)	91.6%	91.59%	-0.003%

There is also one consistency that can be found here. The higher n-gram, the lower the accuracy it gets. The highest accuracy can be obtained by these models with a 5-gram dataset is 95.79%, which is still lower than the previous a 4-gram dataset-based models. It can be seen the performance of the LSTM models under 5-gram in Figure 4a for accuracy and Figure 4b for bias. Similarly, the performance of the GRU models under 6-gram is illustrated in Figure 4c for accuracy and Figure 4d for bias.



(a) (b)

(c) (d)

Figure 4. The overall performances for, (a) LSTM Models Accuracy, (b) LSTM Models Bias, (c) GRU Models Accuracy, (d) GRU Models Bias on 5-Gram Dataset

Another consistency is that the GRU-based models have a lower average bias than the LSTM-based models. It seems like GRU-based models have better precision tendencies compared to LSTM-based models. Nevertheless, still, LSTM-based models across previous models have better accuracy.

3.4. 6-gram Dataset

Lastly, the 6-gram dataset is used to train the models. Interestingly, it is also LSTM that has the same pattern as the previous models. The LSTM model, which has the deepest model, scores the highest accuracy among others. It also shows that the deeper the LSTM layers, the better the performance. However, the GRU-based models show different accuracy results. Table 6 shows the overall performance of LSTM-based and GRU-based models under a 6-gram dataset.

Table 6. The model performance against 6-Gram dataset

Model	Accuracy	Val Accuracy	Average Bias
LSTM (64)	90.11%	89.93%	0.069%
LSTM (64, 64)	92.5%	91.75%	0.365%
LSTM (64, 64, 64)	93.21%	92.68%	0.379%
GRU (64)	89.45%	89.36%	0.004%
GRU (64, 64)	92.89%	92.79%	0.097%
GRU (64, 64, 64)	91.6%	91.59%	-0,003%

The 2-layers GRU (64, 64) model achieved the highest accuracy among other GRU models, reaching 92.89%, and the 1-layers GRU (64) model achieved the lowest, reaching 89.45%. The GRU models did not have the same consistency as the LSTM models. Suppose all the GRU models are being grouped all together. The smallest layer achieves a model with the highest accuracy, and the deepest layer achieves a model with the lowest accuracy. Besides these, the GRU-based models still have the lowest average bias. The Figure 5a and Figure 5b illustrated the LSTM models' performance under 6-gram, shows accuracy and bias, respectively. Similarly, accuracy and bias for the GRU models' performance in the 6-gram are illustrated in Figure 5c and Figure 5d.

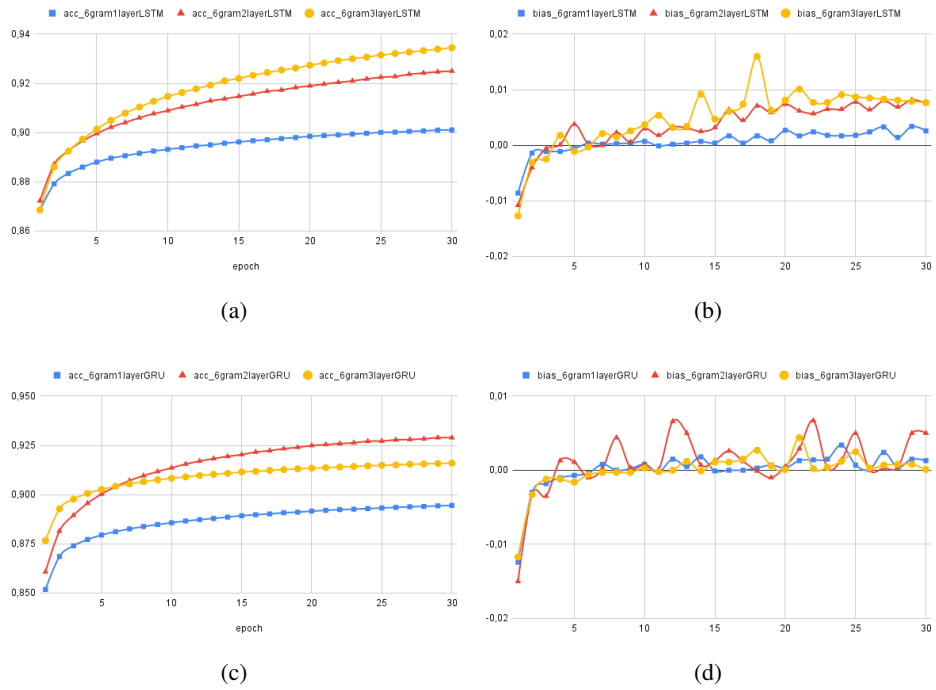


Figure 5. The overall performances for, (a) LSTM Models Accuracy, (b) LSTM Models Bias, (c) GRU Models Accuracy, (d) GRU Models Bias on 6-Gram Dataset

This 6-gram dataset is the most extended dataset to train. Nevertheless, this dataset did not increase the accuracy of the models. Instead, the lower 3-gram dataset with the deeper layers model gives the highest accuracy among others. It might indicate that the nature of the Indonesian language itself is not too complicated to guess with few words besides the context word.

4. CONCLUSION

This study successfully carried out the normalization of words without vowels in Indonesian into complete sentences. The 3-layer LSTM with (64, 64, 64) filter configuration achieved the highest accuracy when trained with the 3-gram dataset, reaching 98.11% accuracy. Although GRU-based models achieved lower accuracy than their LSTM-based counterparts, the GRU-based models have a lower average bias. The result suggests that the ambiguity from the non-vocalized text in the Indonesian language can be solved easily by only seeing the context from one previous and one next word.

This research shows promosing technique to improves Indonesian text dataset preprocessing, where the model can be used to normalized slang or shortened words inside the dataset, enhancing dataset quality for tasks like sentiment analysis, clustering, or translation. Despite being limited to the Indonesian News

Dataset, future work could broaden the model's capabilities by including diverse datasets from sources like children's books or scientific literature. This research also opens to more promising applications in diverse fields, such as text suggestion from abbreviation, typo correction, and lossy text compression in various languages, where the text can be compressed by partially deleting the characters intentionally.

REFERENCES

[1] A. Lutfiatuna, A. Novitasarib, and A. Helfiyanac, "Bahasa alay pada chating di medsos remaja millenial (Bahasa Alay vs Remaja Millenial)," *Pros. SENASBASA* vol. Vol 2, no. 3, pp. 34–41, 2018, doi: 10.22219/.v2i2.2241.

[2] L. O. M. S. Raditya, "Penggunaan bahasa gaul (Bahasa Alay) di twitter," *BASINDO J. Kaji. Bahasa, Sastra Indones. dan Pembelajarannya*, vol. 5, no. 1, pp. 117–123, 2021.

[3] D. Rani Gustiasari, "Pengaruh perkembangan zaman terhadap pergeseran tata Bahasa Indonesia; studi kasus pada pengguna Instagram tahun 2018," *J. Renaiss.*, vol. 3, no. 2, pp. 433, 2018, doi: 10.53878/jr.v3i2.86.

[4] D. S. Maylawati, W. B. Zulfikar, C. Slamet, M. A. Ramdhani, and Y. A. Gerhana, "An improved of Stemming Algorithm for mining Indonesian text with slang on social media," in *2018 6th International Conference on Cyber and IT Service Management (CITSM)*, Parapat, Indonesia, 2018, pp. 1-6, doi: 10.1109/CITSM.2018.8674054.

[5] L. Wu, F. Morstatter, and H. Liu, "SlangSD: building, expanding and using a sentiment dictionary of slang words for short-text sentiment classification," *Lang Resources & Evaluation*, vol. 52, pp. 839–852, 2018, doi: 10.1007/s10579-018-9416-0.

[6] A. R. Pal and D. Saha, "Detection of slang words in e-data using semi-supervised learning," *International Journal of Artificial Intelligence & Applications*, vol. 4, no. 5, pp. 49–61, 2013, doi: 10.5121/ijaia.2013.4504.

[7] L. Saputra and L. Marlina, "An analysis of slang words used by Instagram account Plesbol," *English Language and Literature.*, vol. 8, no. 3, 2019, doi: 10.24036/ell.v8i3.105802.

[8] K. Karlgren and R. Ramberg, "The use of design patterns in overcoming misunderstandings in collaborative interaction design," *CoDesign*, vol. 8, no. 4, pp. 231–246, 2012, doi: 10.1080/15710882.2012.734829.

[9] D. Scheer, C. Benighaus, L. Benighaus, O. Renn, S. Gold, B. Röder, and GF. Böl, "The distinction between risk and hazard: understanding and use in stakeholder communication," *Risk Analysis*, vol. 34, no. 7, pp. 1270–1285, 2014, doi: 10.1111/risa.12169.

[10] A. A. S. Gunawan, P. R. Mulyono, and W. Budiharto, "Indonesian question answering system for solving arithmetic word problems on intelligent humanoid robot," *Procedia Computer Science*, vol. 135, pp. 719–726, 2018, doi: 10.1016/j.procs.2018.08.213.

[11] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE transactions on knowledge and data Engineering*, vol. 28, no. 2, pp. 496–509, 2016, doi: 10.1109/TKDE.2015.2489653.

[12] A. Anikin, A. Katyshev, M. Denisov, V. Smirnov, and D. Litovkin, "Using online update of distributional semantics models for decision-making support for concepts extraction in the domain ontology learning task," in *IOP Conference Series: Materials Science and Engineering*, vol. 483, no. 1, 2019, doi: 10.1088/1757-899X/483/1/012073.

[13] S. Thavareesan and S. Mahesan, "Sentiment lexicon expansion using Word2vec and fastText for sentiment prediction in Tamil texts," in *2020 Moratuwa engineering research conference (MERCon)*, 2020, pp. 272–276, doi: 10.1109/MERCon50084.2020.9185369.

[14] D. Zaky and A. Romadhony, "An LSTM-based spell checker for Indonesian text," in *2019 international conference of advanced informatics: concepts, theory and applications (ICAICTA)*, 2019, pp. 1-6, doi: 10.1109/ICAICTA.2019.8904218.

[15] P. Santoso, P. Yuliawati, R. Shalahuddin, and A. P. Wibawa, "Damerau Levenshtein distance for Indonesian spelling correction," *J. Inform.*, vol. 13, no. 2, pp. 11-15. 11, 2019, doi: 10.26555/jifo.v13i2.a15698.

[16] W. Octoviani, M. Fachrurrozi, N. Yusliani, M. Febriady, and A. Firdaus, "English-Indonesian phrase translation using Recurrent Neural Network and adj technique," in *Journal of Physics: Conference Series*, 2019, vol. 1196, no. 1, doi: 10.1088/1742-6596/1196/1/012007.

[17] A. S. Kholimi and F. Nazihullah, "Identifikasi tulisan Arab dengan menggunakan GLCM dan RNN," in *Prosiding SENTRA (Seminar Teknologi dan Rekayasa)*, 2019, no. 4, pp. 39–43, doi: 10.22219/sentra.v0i4.2323.

[18] Y. Wibisono and M. L. Khodra, "Pengenalan entitas bernama otomatis untuk Bahasa Indonesia dengan pendekatan pembelajaran mesin," in *Semin. Tah. Linguist. 2018*, 2018, pp. 1–5.

[19] K. Yoko, V. C. Mawardi, and J. Hendryli, "Sistem peringkas otomatis abstraktif dengan menggunakan Recurrent Neural Network," *Computatio: Journal of Computer Science and Information Systems*, vol. 2, no. 1, p. 65, 2018, doi: 10.24912/computatio.v2i1.1481.

[20] A. Chandra, "Indonesia news dataset," Indonesia, 2020, [Online]. Available: https://github.com/andreaschandra/indonesian-news.

[21] U. Singh, V. Goyal, and A. Rani, "Disambiguating hindi words using n-gram smoothing models," *International Journal of Engineering Sciences*, vol. 10, pp. 26–29, 2014.

[22] H. K. Poon, W. S. Yap, Y. K. Tee, W. K. Lee, and B. M. Goi, "Hierarchical Gated Recurrent Neural Network with

adversarial and virtual adversarial training on text classification," *Neural Networks*, vol. 119, pp. 299–312, 2019, doi: 10.1016/j.neunet.2019.08.017.

[23] A. Kumar and R. Rastogi Nee Khemchandani, "Self-Attention enhanced Recurrent Neural Networks for sentence classification," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 905–911, doi: 10.1109/SSCI.2018.8628865.

[24] E. A. Nismi Mol and M. B. Santosh Kumar, "Study on impact of RNN, CNN and HAN in text classification," in *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, 2020, pp. 94–102, doi: 10.1109/ACCTHPA49271.2020.9213231.

[25] L. Wiranda and M. Sadikin, "Penerapan Long Short-Term Memory pada data time series untuk memprediksi penjualan produk Pt. Metiska Farma," *Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI*, vol. 8, no. 3, pp. 184–196, 2019, doi: 10.23887/janapati.v8i3.19139.

[26] K. Moharm, M. Eltahan, and E. Elsaadany, "Wind speed forecast using LSTM and Bi-LSTM algorithms over gabal el-zayt wind farm," in *2020 International Conference on Smart Grids and Energy Systems (SGES)*, 2020, pp. 922–927, doi: 10.1109/SGES51519.2020.00169.

[27] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep Learning-Based stock price prediction using LSTM and Bi-Directional LSTM model," in *2020 2nd novel intelligent and leading emerging sciences conference (NILES)*, 2020, pp. 87–92, doi: 10.1109/NILES50944.2020.9257950.

[28] S. Li, Q. Wang, X. Liu, and J. Chen, "Low cost LSTM implementation based on stochastic computing for channel state information prediction," in*2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2019, pp. 231–234, doi: 10.1109/APCCAS.2018.8605569.

[29] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1186–1194, doi: 10.1109/WACV.2018.00135.

[30] S. Dai, L. Li, and Z. Li, "Modeling vehicle interactions via modified LSTM models for trajectory prediction," *IEEE Access*, vol. 7, pp. 38287–38296, 2019, doi: 10.1109/ACCESS.2019.2907000.

[31] S. Chakraborty, J. Banik, S. Addhya, and D. Chatterjee, "Study of dependency on number of LSTM units for character based text generation models," in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2020, pp. 1-5, doi: 10.1109/ICCSEA49143.2020.9132839.

[32] F. Miedema, "Sentiment analysis with Long Short-Term Memory networks," *Vrije Universiteit Amsterdam*, vol. 1, pp. 1–17, 2018.

[33] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of Recurrent Neural Networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no.7, pp. 1235–1270, 2019, doi: 10.1162/neco_a_01199.

[34] J. Patihullah and E. Winarko, "Hate speech detection for Indonesia tweets using word embedding and Gated Recurrent Unit," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 13, no. 1, pp. 43-52, 2019, doi: 10.22146/ijccs.40125.

[35] C. Ronran and S. Lee, "Effect of Character and Word Features in Bidirectional LSTM-CRF for NER," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Busan, Korea (South), 2020, pp. 613-616, doi: 10.1109/BigComp48618.2020.00132.

[36] D. Yolanda, K. Gunadi, and E. Setyati, "Pengenalan slfabet bahasa isyarat yangan secara real- time dengan menggunakan metode Convolutional Neural Network dan Recurrent Neural Network," *Jurnal Infra*, vol. 8, no. 1, pp. 203–208, 2020.

[37] S. Sun, Y. Zhang, P. Lin, W. Ren, and J. A. Farrell, "Distributed time-varying optimization with state-dependent gains: algorithms and experiments," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 1, pp. 416–425, 2021, doi: 10.1109/TCST.2021.3058845.

[38] X. Liu, "Research on the forecast of coal price based on LSTM with improved Adam optimizer," in *Journal of physics: conference series*, 2021, vol. 1941, no. 1, p. 012069, doi: 10.1088/1742-6596/1941/1/012069.

[39] S. Jiang and Y. Chen, "Hand gesture recognition by using 3DCNN and LSTM with adam optimizer", in *Pacific Rim Conference on Multimedia*, 2017, pp. 743-753.

[40] S. Bock, J. Goppold, and M. Weiß, "An improvement of the convergence proof of the ADAM-Optimizer," *arXiv preprint arXiv:1804.10587*, 2018, doi: 10.48550/arXiv.1804.10587.