Media Statistika

p-ISSN 1979 – 3693 e-ISSN 2477 – 0647
MEDIA STATISTIKA 15(2) 2022: 129-138
http://ejournal.undip.ac.id/index.php/media_statistika

SPRATAMA MODEL FOR INDONESIAN PARAPHRASE DETECTION USING BIDIRECTIONAL LONG SHORT-TERM MEMORY AND BIDIRECTIONAL GATED RECURRENT UNIT

Stanley Pratama, Titin Siswantining, Gianinna Ardaneswari
Department of Mathematics, Universitas Indonesia, Indonesia

e-mail: *titin@sci.ui.ac.id*

DOI: 10.14710/medstat.15.2.129-138

Article Info:
Received: 11 June 2022
Accepted: 5 March 2023
Available Online: 4 April 2023

Keywords:
natural language processing, natural language sentence matching, recurrent neural network.

Abstract: Paraphrasing is a way to write sentences with other words with the same intent or purpose. Automatic paraphrase detection can be done using Natural Language Sentence Matching (NLSM) which is part of Natural Language Processing (NLP). NLP is a computational technique for processing text in general, while NLSM is used specifically to find the relationship between two sentences. With the development Neural Network (NN), nowadays NLP can be done more easily by computers. Many models for detecting and paraphrasing in English have been developed compared to Indonesian, which has less training data. This study proposes SPratama Model, which models paraphrase detection for Indonesian using a Recurrent Neural Network (RNN), namely Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU). The data used is "Quora Question Pairs" taken from Kaggle and translated into Indonesian using Google Translate. The results of this study indicate that the proposed model has an accuracy of around 80% for the detection of paraphrased sentences.

1. INTRODUCTION

Paraphrasing is the re-expression of an utterance from one level or type of language to another without changing the meaning (KBBI, 2016). Crystal (1980) states that paraphrasing is a term in linguistics for the result or process of producing alternative versions of sentences or texts without changing the meaning. For humans, paraphrasing a sentence or detecting those two sentences have the same meaning is relatively easy compared to computers.

Natural Language Sentence Matching (NLSM) is part of Natural Language Processing (NLP) which focuses on finding the relationship between two sentences. For example, in identifying a paraphrased sentence, NLSM is used to determine whether two sentences are paraphrases of one of them (Yin et al., 2015). In Natural Language Inference, NLSM is used to assess whether the hypothetical sentence can be inferred from the premise sentence (Bowman et al., 2015). NLSM is also used in answering questions and seeking information by assessing the relevance between question-answer pairs and ranking all candidate answers (Wang et al., 2016a). In machine comprehension, NLSM is used to match

parts of a passage to a question and indicate the part of the passage where there is an answer to the question (Wang et al., 2016b).

With the development of neural networks, words can be represented with a vector by using embedding (Bengio et al., 2003). Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014) are utilized to process text. LSTM and GRU are designed to have memory for previous inputs so that inputs are processed in a certain order (forward or backward). LSTM utilizes memory to process text by remembering what was processed or read before. GRU is a further development of LSTM which has a simpler architecture. Although simpler, GRU was found to have similar performance to LSTM (Ravanelli et al., 2018; Su and Kuo, 2019; Chung et al., 2014; Gruber and Jockisch, 2020). Ordinary RNNs have limitations in that they can only utilize context or memory from before. In paraphrasing detection, of course, both sentences can be read back-and-forth and the result remains the same, so using context from two directions makes more sense. Bidirectional RNN (BRNN) or RNN with two directions doubles the processing of the RNN so that the input is processed in forward and backward order (Schuster and Paliwal, 1997). This allows the BRNN to view the context in both directions and get better performance than a one-way RNN (Graves and Schmidhuber, 2005). By combining BRNN with LSTM or GRU, we get bidirectional LSTM (BiLSTM) or bidirectional GRU (BiGRU).

Compared to English, Indonesian has less available data and thus resulted in fewer models being made for Indonesian. Multilingual language processing models that are capable of processing various languages have been created and developed, but that is beyond the scope of this study. This study proposes SPratama model for the task of detecting Indonesian paraphrases. The model first uses an Indonesian embedding taken from https://tfhub.dev/google/nnlm-id-dim128-with-normalization/2 (Google, 2021) which is trained with “Indonesian Google News 3B Corpus” data based on from feed-forward Neural-Net Language Models (Bengio et al., 2003). Then a BiLSTM or BiGRU layer is used to process and combine the two sentences being compared into a vector. Then the vector will be processed using a fully connected layer (feed-forward neural network) and produce output that is used to make decisions.

2. LITERATURE REVIEW

2.1. Recurrent Neural Network

Recurrent Neural Network (RNN) is a neural network that has cycles in its connections. That is, the unit value of the neural network directly or indirectly depends on its own output as the next input. Simple RNN can be written in equations:

$$\boldsymbol{h_t} = \boldsymbol{g}(\boldsymbol{U h_{t-1}} + \boldsymbol{W x_t} + \boldsymbol{b}) \quad (1)$$

$$\boldsymbol{y_t} = \boldsymbol{f}(\boldsymbol{V h_t} + \boldsymbol{d}) \quad (2)$$

where $\boldsymbol{x_t}$ denotes the input $(x_1, \dots, x_T)$, $\boldsymbol{h_t}$ denotes the hidden layer $(h_1, \dots, h_T)$, $\boldsymbol{y_t}$ denotes output or also called hidden state $(y_1, \dots, y_T)$, $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ are weight matrices, $\boldsymbol{b}$ and $\boldsymbol{d}$ are constant matrices, and f and g are the activation functions.

2.2. Bidirectional Recurrent Neural Network

Bidirectional Recurrent Neural Network (BRNN) consists of two RNNs that have different "read" directions, if one reads from left to right (forward), the other will read from right to left (backward). Because there are two RNNs, there will be two hidden states (output) namely forward and backward.

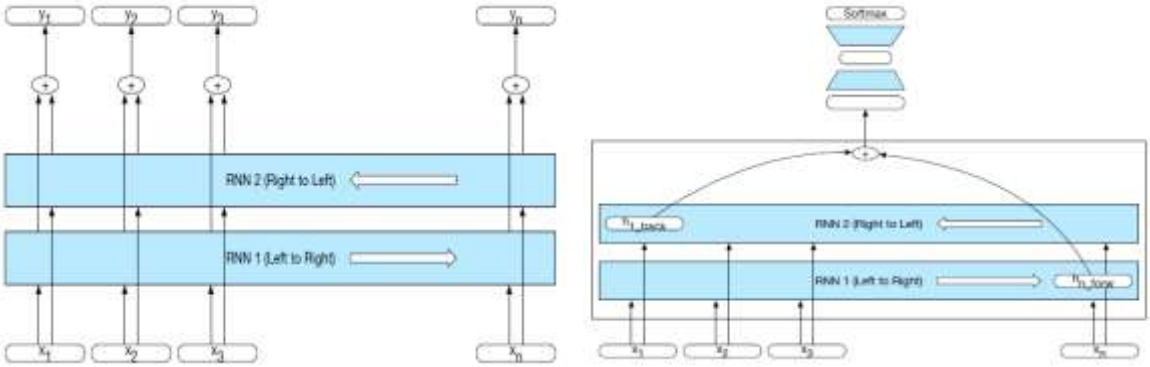
$$\boldsymbol{h_t^f = RNN_{forward}(x)} \quad (3)$$

$$\boldsymbol{h_t^b = RNN_{backward}(x)} \quad (4)$$

The two hidden states will be combined into one hidden state:

$$\boldsymbol{h_t = h_t^f \oplus h_t^b} \quad (5)$$

where $\oplus$ can be multiplication per element, addition, concatenation, or other operators between two hidden states. Figure 1 (a) illustrates a BRNN which combine its two hidden states for every step and Figure 1 (b) illustrates a BRNN which combine only its two final hidden states. This study will use the BRNN architecture illustrated by Figure 1 (b).



(a) (b)

Figure 1. Bidirectional Recurrent Neural Network (BRNN) Architecture with
(a) Combining its Two Hidden States for Every Step and
(b) Combining Only its Two Final Hidden States

2.3. Long Short-Term Memory

Long Short-Term Memory (LSTM) is an RNN architecture. LSTM has four components, namely forget gate, input gate, memory state or cell, and output gate. The gate functions to filter the information provided and the cell functions to store information from the given input (Hochreiter and Schmidhuber, 1997).

$$\boldsymbol{f_t = \sigma\left(U_f h_{t-1} + W_f x_t + d_f\right)} \quad (6)$$

$$\boldsymbol{f_t = \sigma\left(U_f h_{t-1} + W_f x_t + d_f\right)} \quad (7)$$

$$\boldsymbol{i_t = \sigma(U_i h_{t-1} + W_i x_t + d_i)} \quad (8)$$

$$\boldsymbol{c_t = i_t \odot g_t + f_t \odot c_{t-1}} \quad (9)$$

$$\boldsymbol{o_t = \sigma(U_o h_{t-1} + W_0 x_t + d_o)} \quad (10)$$

$$\boldsymbol{h_t = o_t \odot \tanh(c_t)} \quad (11)$$

where σ is the sigmoid function, $\odot$ denotes element-wise multiplication, $\boldsymbol{f_t}$ is the forget gate, $\boldsymbol{i_t}$ is the input gate, $\boldsymbol{c_t}$ is the memory state or cell, $\boldsymbol{o_t}$ is the output gate, and $\boldsymbol{h_t}$ is the output or hidden state.

2.4. Gated Recurrent Unit

Gated Recurrent Unit (GRU) is an RNN architecture similar to an LSTM without an output gate. GRU has a cell or candidate vector and two gates, namely: reset gate and update

gate (Cho et al., 2014). The hidden state (output) is obtained by using the z value to interpolate between the previous hidden state and the candidate vector.

$$\boldsymbol{r_t} = \boldsymbol{\sigma}(\boldsymbol{U_r h_{t-1}} + \boldsymbol{W_r x_t} + \boldsymbol{d_r}) \quad (12)$$

$$\boldsymbol{z_t} = \boldsymbol{\sigma}(\boldsymbol{U_z h_{t-1}} + \boldsymbol{W_z x_t} + \boldsymbol{d_z}) \quad (13)$$

$$\widetilde{\boldsymbol{h_t}} = \tanh(\boldsymbol{U}(\boldsymbol{r_t} \odot \boldsymbol{h_{t-1}}) + \boldsymbol{W x_t} + \boldsymbol{d_h}) \quad (14)$$

$$\boldsymbol{h_t} = (\boldsymbol{1} - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}} + \boldsymbol{z_t} \odot \widetilde{\boldsymbol{h_t}} \quad (15)$$

where σ is the sigmoid function, $\odot$ denotes element-wise multiplication, $\boldsymbol{r}_t$ is the reset gate, $\boldsymbol{z}_t$ is the update, $\widetilde{\boldsymbol{h}_t}$ is the cell or candidate vector, and $\boldsymbol{h}_t$ is the output or hidden state.

Bidirectional LSTM (BiLSTM) and Bidirectional GRU (BiGRU) are achieved by combining BRNN with LSTM and GRU. BiLSTM and BiGRU consist of two LSTMs or two GRUs that run in parallel: one on the forward "read" direction and the other in the backward "read" direction.

2.5. Hash function

A hash function is any function that can be used to map data of arbitrary size to a fixed size value. A good hash function is one that satisfies (more or less) the simple uniformity assumption of hashing: each key (input) has the same probability of being hashed or mapped to one of the m slots, also called hash buckets, regardless of the hash or mapping results of other keys (Cormen *et al.*, 2009). Figure 2 illustrates a hash function.

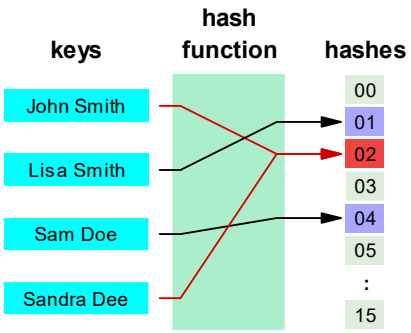


Figure 2. Illustration of a Hash Function that Maps the Keys into 16 Hash Buckets

2.6. Adam Optimizer

The name "Adam" comes from adaptive moment estimation. Adam optimizer is based on stochastic gradient descent algorithm by finding the first and second moments from the first derivative of the loss function with an adaptive learning rate. Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the step size hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

Adam algorithm has input step size; exponential decay rates; stochastic objective function; initial parameter vector; initial first moment vector; and initial second moment vector. The process is as follow: calculate gradients with respect to stochastic objective, update biased first and second raw moment estimate, compute bias-corrected first and second raw, update parameter vector. Iteration process will be stop until we get parameter values converged (Kingma and Lei Ba, 2014).

3. MATERIAL AND METHOD

3.1. Spratama Model

The SPratama model first converts sentences into vectors using an embedding layer, then the results are processed using BiLSTM or BiGRU and followed by a feed-forward neural network with one to two hidden layers with the ReLU activation function. ReLU is used because of its simple function and good performance in neural network models (Glorot et al., 2011; Krizhevsky et al., 2012; Tóth, 2013). The sigmoid activation function is used for the output layer because it is a binary classification.

The architecture of the model can be seen in Figure 3. First, the sentence will be converted into a normalized vector with 128 elements (dimensions) using an embedding layer. Then the two output vectors will be concatenated into a 2×128 matrix. The matrix will be continued to the RNN layer, then to the fully connected layer (feed-forward neural network) with the ReLU activation function, and to the output layer with the sigmoid activation function. Dropout (Srivastava et al., 2014) is used as a regularization technique to prevent overfitting in the fully connected layer.

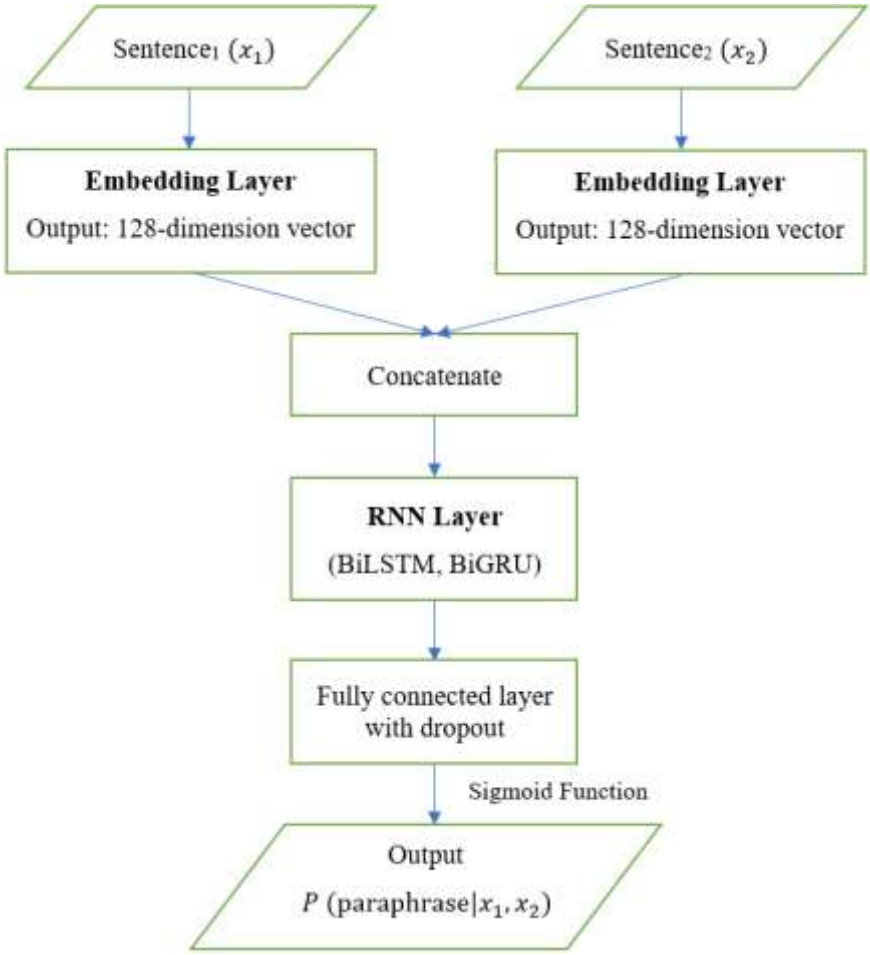


Figure 3. Architecture for SPratama Model

The embedding layer is obtained from https://tfhub.dev/google/nnlm-id-dim128-with-normalization/2 which is a pretrained model based on *Neural Probabilistic Language Model* (Bengio *et al.*, 2003) with the "Indonesian Google News 3B Corpus" dataset developed by Google. Embedding for sentences is obtained by adding the embedding vector of each word, ignoring spaces and punctuation marks, then dividing by the root of the word

count. The embedding for sentences has a weakness because as the embedding does not care about the order of the words in the sentence, but this embedding process is computationally fast and the model still able to get good results. This layer will not be retrained with the existing dataset because it will take a very long time and this dataset is not specifically designed to train the embedding model.

Words that are not in the dataset in the model training process or called Out of Vocabulary (OOV) are processed using a hash bucket. A fraction of the tokens and embedding that are rarely used (~2.5%) are replaced by hash buckets. Each hash bucket is initialized using the remaining embedding vectors that have a hash to the same bucket. In other words, OOV has its own embedding and is not unique.

There are four models built with differences in the RNN and the layers in the fully connected layer. The model uses one Bidirectional RNN layer, namely BiLSTM or BiGRU and one or two hidden layers and one output layer. The model will be named in the manner "RNN-number of fully connected layers". For example, BiLSTM-1 means the model uses BiLSTM and one fully connected layer.

Cross-entropy loss is used as a loss function and the matrices weight are estimated using the Adam optimizer (Kingma and Lei Ba, 2014). The optimal number of units and dropout rate for each layer are searched using the Hyperband algorithm (Li et al., 2017).

3.2. Experiment

The data used for this research is "Quora Question Pairs" (Chen et al., 2018) in the form of questions pairs written in English taken from the Quora website and labeled whether the two questions are duplicates or have the same meaning, the data can be accessed through the https://www.kaggle.com/c/quora-question-pairs/data (*Kaggle*, 2017) as of September 2021. This dataset has 404290 question pairs with pairs labeled with "0", not duplicates and pairs labeled with "1", duplicates. Table 1 shows two rows of the dataset. The field "id", "qid1", and "qid2" can be ignored as they are only the identification number for the questions and are not used in the model.

Table 1. "Quora Question Pairs" Dataset

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0

There are 2 blank rows in the "question2" column for this dataset, so these two rows are discarded. The dataset was translated from English to Indonesian using Google Translate. The results of the translation can be seen in Table 2.

The data is divided into three, namely 80% training, 10% validation, and 10% testing. Modeling is done with training data and validated using validation data, after that it is tested against testing data. The model is validated using a validation set and evaluated based on the loss value and accuracy. The loss value used in this study is cross-entropy and a measure of accuracy for binary classification.

The model parameters will be optimized using the Adam algorithm with a learning rate (α) of 0.001 and a mini batch of 32. Hyperparameters, namely the number of units and

dropout rates, are optimized with the Hyperband algorithm, which is set to maximize the model's accuracy against the validation set with $R = 16$ epochs and $\eta = 3$. Hyperparameter tuning and model training are carried out with early stopping, if the accuracy of the validation set decreases the training process will stop, to prevent overfitting. The number of units for BiLSTM and BiGRU is limited to three values, namely 32, 64, and 128 units. The number of units for a feed-forward neural network is limited from 32 units to 512 units in increments of 32 units. The dropout rate is limited from 0 to 0.5 in increments of 0.05.

The implementation of the program for the model was carried out using Python 3.8.8 with TensorFlow 2.3.0 library. Programming is run on a computer with the AMD Ryzen 7 5800H with Radeon Graphics @3.20 GHz, NVIDIA GeForce RTX 3060 Laptop GPU, and 16GB of RAM.

Table 2. Translated **"**Quora Question Pairs" dataset.

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	Apa panduan langkah demi langkah untuk berinvestasi di pasar saham di India?	Apa panduan langkah demi langkah untuk berinvestasi di pasar saham?	0
1	3	4	Bagaimana kisah Berlian Kohinoor (Koh-i-Noor)?	Apa yang akan terjadi jika pemerintah India mencuri kembali berlian Kohinoor (Koh-i-Noor)?	0

4. RESULTS AND DISCUSSION

There are four models built, namely BiLSTM and 1 fully connected layer (BiLSTM-1), BiLSTM and 2 fully connected layers (BiLSTM-2), BiGRU and 1 fully connected layer (BiGRU-1), and BiGRU and 2 fully connected layers (BiGRU-2). Hyperparameter tuning takes quite a long time which is around 6000 to 7000 seconds for each model. Table 3 shows the accuracy and the number of parameters of the four models built.

Table 3. Accuracy (to the Test Set) and Parameters of the Four Models

Model	Accuracy	Parameters
BiLSTM dan 1 *fully connected layer* (BiLSTM-1)	80,66%	329.127
BiLSTM dan 2 *fully connected layer* (BiLSTM-2)	80,40%	444.257
BiGRU dan 1 *fully connected layer* (BiGRU-1)	80,62%	280.705
BiGRU dan 2 *fully connected layer* (BiGRU-2)	80,73%	434.945

For BiLSTM-1, from the hyperparameter tuning, 128 units are obtained for BiLSTM, 256 units for the fully connected layer with a dropout rate of 0.45. This model has 329,127 parameters. The model is trained for 16 epochs but with earlystopping the training process stops at 10 epochs. Figure 3 (a) shows a graph of the accuracy of the model per epoch.

For BiLSTM-2, from the hyperparameter tuning, 128 units were obtained for BiLSTM, 352 and 256 units for the fully connected layer with dropout rates of 0.3 and 0.1. This model has 444,257 parameters. The model is trained for 16 epochs but with earlystopping the training process stops at 7 epochs. Figure 3 (b) shows a graph of the accuracy of the model per epoch.

For BiGRU-1, from the hyperparameter tuning, 128 units were obtained for BiGRU, 320 units for the fully connected layer with a dropout rate of 0.45. This model has 280,705 parameters. The model is trained for 16 epochs but with earlystopping the training process stops at 9 epochs. Figure 3 (c) shows a graph of the accuracy of the model per epoch.

For BiGRU-2, from the hyperparameter tuning, 128 units were obtained for BiGRU, 320 and 480 units for the fully connected layer with dropout rates of 0.2 and 0.25. This model has 434,945 parameters. The model is trained for 16 epochs but with earlystopping the training process stops at 8 epochs. Figure 3 (d) shows a graph of the accuracy of the model per epoch.

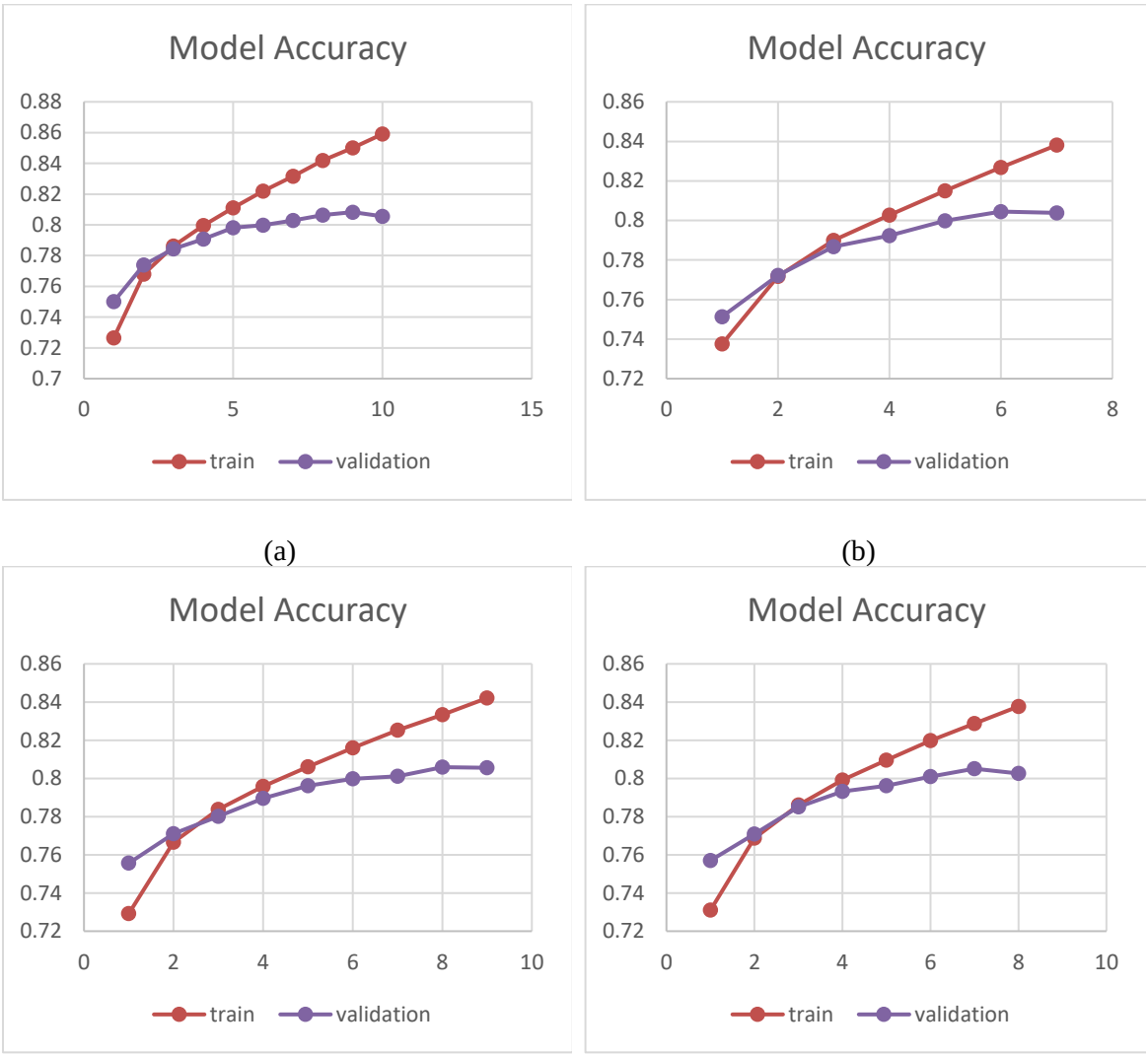


Figure 4. Accuracy of the Models per Epoch
(a) BiLSTM-1, (b) BiLSTM-2, (c) BiGRU-1, and (d) BiGRU-2

The four models have similar accuracy to each other. From the graph of the accuracy of the four models seen in the second or third epoch, the accuracy of the training set will exceed the accuracy of the validation set. It can also be seen that the increase in accuracy in validation is slower for per epoch after the second or third epoch compared to accuracy in training. The model with BiGRU and 1 fully connected layer (BiGRU-1) is the best model for this problem compared to the other three models because this model has the fewest parameters.

Since this dataset is a relatively new dataset, we compare our model to the models developed on the English dataset. First, "Siamese CNN" and "Siamese LSTM" model designed according to the architecture in Wang et al., 2016c. Second, "L.D.C" model

proposed in Wang et al., 2016d. Finally, "Bilateral Multi Perspective Matching Model (BiMPM)" proposed in Wang et al., 2017. The comparison is shown in Table 4.

Table 4. Accuracy of the Compared Models

Model	Accuracy
Siamese CNN (English)	79,60%
BiGRU-1	**80,62%**
Siamese LSTM (English)	82,58%
L.D.C (English)	85,55%
BiMPM (English)	88,17%

While BiGRU-1 model does not perform as the best, it is relatively simple and holds quite well against the other models. This shows that our simple model is still good for paraphrase detection.

5. CONCLUSION

The proposed SPratama architecture for Indonesian paraphrase detection tasks using BiLSTM and BiGRU performs quite well. There are four models that are built with this architecture namely BiLSTM and 1 fully connected layer (BiLSTM-1), BiLSTM and 2 fully connected layers (BiLSTM-2), BiGRU and 1 fully connected layer. (BiGRU-1), and BiGRU and 2 fully connected layers (BiGRU-2). The four models have similar accuracy to each other and are quite good, which is around 80%.

REFERENCES

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research, 3*:1137-1155.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A Large Annotated Corpus for Learning Natural Language Inference. *arXiv preprint arXiv:1508.05326.*

Chen, Z., Zhang, H., Zhang, X., & Zhao, L. (2018). Quora Question Pairs. *URL https://www. kaggle.com/c/quora-question-pairs*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT press.

Crystal, David. (1980). *A First Dictionary of Linguistics and Phonetics.* Boulder, Colorado: Westview Press.

Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315-323). JMLR Workshop and Conference Proceedings.

Google. (2021). *nnlm-id-dim128-with-normalization* - Token Based Text Embedding Trained on Indonesian Google News 3B corpus. Tensorflow hub. Retrieved

December 20, 2021, from https://tfhub.dev/google/nnlm-id-dim128-with-normalization/2

Graves, A., & Schmidhuber, J. (2005). Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural networks*, *18*(5-6), 602-610.

Gruber, N., & Jockisch, A. (2020). Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification Of Text?. *Frontiers in Artificial Intelligence*, *3*, 40.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580.*

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, *9*(8), 1735-1780.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980.*

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *The Journal of Machine Learning Research*, *18*(1), 6765-6816.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673-2681.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Su, Y., & Kuo, C. C. J. (2019). On Extended Long Short-Term Memory and Dependent Bidirectional Recurrent Neural Network. *Neurocomputing*, *356*, 151-161.

Tóth, L. (2013, May). Phone Recognition with Deep Sparse Rectifier Neural Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6985-6989). IEEE.

Wang, Z., Mi, H., & Ittycheriah, A. (2016a). Sentence Similarity Learning by Lexical Decomposition And Composition. *arXiv preprint arXiv:1602.07019*.

Wang, Z., Mi, H., Hamza, W., & Florian, R. (2016b). Multi-Perspective Context Matching for Machine Comprehension. *arXiv preprint arXiv:1612.04211*.

Wang, Z., Mi, H., & Ittycheriah, A. (2016c). Semi-Supervised Clustering for Short Text Via Deep Representation Learning. *arXiv preprint arXiv:1602.06797*.

Wang, Z., Mi, H., & Ittycheriah, A. (2016d). Sentence Similarity Learning by Lexical Decomposition and Composition. *arXiv preprint arXiv:1602.07019*.

Wang, Z., Hamza, W., & Florian, R. (2017). Bilateral Multi-Perspective Matching for Natural Language Sentences. *arXiv preprint arXiv:1702.03814.*

Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2015). Abcnn: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *arXiv preprint arXiv:1512.05193.*