Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 6, No. 1, February 2021, Pp. 11-18

Improving automatic essay scoring for Indonesian language using simpler model and richer feature

Rian Adam Rajagede
Department of Informatics, Universitas Islam Indonesia, Yogyakarta, Indonesia

Article Info
Keywords:
Automatic Essay Scoring, BERT Sentence Embedding, Neural Network

Article history:
Received: January 1, 2021
Accepted: January 13, 2021
Published: February 28, 2021

Cite:
Rajagede, R. A. (2021). Improving Automatic Essay Scoring for Indonesian Language using Simpler Model and Richer Feature. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 6(1). https://doi.org/10.22219/kinetik.v6i1.1196

*Corresponding author.
Rian Adam Rajagede
E-mail address:
rian.adam@uii.ac.id

Abstract
Automatic essay scoring is a machine learning task where we create a model that can automatically assess student essay answers. Automated essay scoring will be instrumental when the answer assessment process is on a large scale so that manual correction by humans can cause several problems. This study proposes to use a simpler model than previously published, a single hidden layer neural network but using a richer feature, BERT sentence embedding. We compare the result using Ukara dataset released for automatic essay scoring for the Indonesian language. The best model that has been published produces an F1-score of 0.821 using pre-trained fastText sentence embedding and the stacking model between the neural network and XGBoost. We use pre-trained BERT sentence embedding that extracts more information from sentences but has a smaller file size than fastText pre-trained model. Our model manages to get a higher F1-score than the previous models on the Ukara dataset, which is 0.829.

1. Introduction

In contrast to multiple-choice or other forms of objective questions, which are closed-ended, in essay-type questions, each student can provide different correct answers (open-ended question). Essay type questions can be defined as questions with freely constructed answers and can consist of one or more sentences [1]. The answer key to this type of question is usually only a guideline that helps assessors assess student answers. Therefore, the role of the assessor in this type of problem is crucial. Unfortunately, relying on humans to assess student essay answers on a large scale, for example, on a national scale, will take a very long time. Besides, the assessment process will also require many resources because it requires many competent assessors to assess student work. The combination of these two problems can lead to inconsistencies in essay assessment even though there are guidelines.

Automatic essay scoring (AES) is a machine learning task in natural language processing, where we create a model that is able to assess students' essay answers automatically. In machine learning, automatic essay scoring can be seen as a text classification or regression problem, depending on the data. The AES model will be needed when the examination is conducted on a large scale with a fairly short assessment time. However, making accurate AES is not easy. The available data for training essay scoring models is usually very limited and naturally has imbalance classes. The limited amount of data can be caused by difficulties in conducting a massive data labeling process because it requires experts to validate each data's labels. Meanwhile, the class imbalance can occur because the problemsetters wants to test students' competence by making questions that are difficult to answer correctly. Other challenges also arise when the number of questions is increasing. When the number of essay-type questions increases, then the labeling, validation, model training, and parameter tuning processes will also take a longer time.

Natural Language Processing Research Group Universitas Gadjah Mada, Indonesia, released a dataset for Indonesian language AES in 2019. The dataset was released simultaneously with the Ukara Automatic Short Answer System [2] and the Ukara Automatic Essay Scoring Challenge (Ukara 1.0 Challenge)[1] in collaboration with the Education Assessment Center, Ministry of Education and Culture of Indonesia. The Ukara dataset consists of two different essay questions in Indonesian. In the Ukara 1.0 Challenge, participants were challenged to create a model that able to automatically assess student answers based on the dataset.

The three best teams in the Ukara 1.0 Challenge used different approaches to complete the AES task on the dataset. The team that got the first position used the stacked neural network model [3], which ensemble two base

[1]https://nlp.mipa.ugm.ac.id/ukara-1-0-challenge/

models, neural network and extreme gradient boosting (XGBoost) [4] using stacking method. The feature they used is the fastText sentence embedding vector [5]. The team received the best F1-score of 0.821 for both types of questions. While the team in second position used a sequential and deep learning approach using Bidirectional Long-Short Term Memory (Bi-LSTM) [6]. The feature they used is word embedding using Word2Vec [7]. The team in the third position uses a single random forest model and logistic regression for each different question. The F1-score for both teams published on the Ukara homepage was 0.81, while the recently published paper shows that the best F1-score was 0.811 [8].

Deep learning models are widely proposed for short answer scoring or essay scoring. Various neural network architectures have been proposed, such as Long-Short Term Memory (LSTM), Attention-based model, Convolutional Neural Network (CNN), Siamese Network, or a combination of them [9][10][11][12]. By representing the AES task as one of the problems of text classification or regression, deep learning is commonly used. It can produce better results than other traditional machine learning methods [13][14][15]. However, in the Ukara 1.0 Challenge, the deep learning approach did not get the best results. This could be due to the very small number of available data. Another approach is to use a tree-based model as is done in [2][3][16][17]. The use of tree-based models has another advantage. The model can explain the scores given as shown in [16]. In the case of Indonesian AES, besides the previously mentioned methods, one of the methods that have been used is using Latent Semantic Analysis (LSA) [18] or term frequency similarity [19]. Although many models have been proposed, unfortunately, we cannot compare some models due to the datasets' different characteristics. For example, the Ukara dataset is one of the text classification problems where student answers are assessed in a binary manner, true or false. In contrast to some of the datasets used in [17] and another Indonesian AES dataset, the Indonesian Query Answering Dataset [20], which scored between 0 and maximum point. In addition, in [16] the dataset has different answer characteristics that relatively longer than the Ukara dataset.

From these studies, we can also summarize some of the features used. Vector word embedding, sentence embedding, and Bag-of-Words (BoW) are the most commonly used. Word embedding is a method used to map a word to a vector space, while sentence embedding does the same thing but at the sentence level. In the Ukara 1.0 Challenge, the best models use fastText sentence embedding as their main feature [3]. In previous research, sentence embedding has also been used in various text classification problems with good results [21][22][23]. Apart from using fastText, another method that can be used to do sentence embedding is using Bidirectional Encoder Representations from Transformers (BERT) [24][25][26].

In this study, we propose to use a simpler model but with richer features for AES problem. We faced automatic essay scoring as a sentence-level problem and used a single hidden layer neural network with the BERT sentence embedding feature. Compared to fastText from previous research, pre-trained BERT sentence embedding size only a quarter of fastText pre-trained model size. With a simpler model, the number of parameters that need to be tuned is less so that the parameter tuning process is faster. In addition, the simpler model also opens up more possibilities for deployment models across multiple platforms. We use the Ukara dataset as a benchmark dataset to compare with the results of previous research.

2. Research Method

2.1 Datasets

This study used the Ukara dataset published in 2019 by the NLP Research Group, Gadjah Mada University, Indonesia. The dataset can be accessed on its homepage. In Ukara dataset, student answers are assessed in a binary manner, true or false. This dataset represents an essay-type national exam that is being developed by the Ministry of Education and Culture of Indonesia.

The Ukara dataset is an Indonesian language automatic essay scoring dataset which consists of two types of questions, Question A and Question B. We will be given a stimulus text, a question, answer guideline, and student answers along with the score for each question. Each dataset has also been divided into three files: trainset, validation set (named "dev" file), and test set.

In question A, students were asked to give their opinion about what people would face when changing their hometown due to natural disasters. In question B, students were asked to give their opinion about the reasons for someone who donates after watching a social condition video. Table 1 presents the statistics for the two questions.

Table 1. Ukara Dataset Summary

	Question A	Question B
Total train samples	268	305
Correct train samples	191	168
Incorrect train samples	77	137
Avg. answer length (words)	12.13	13.68

As can be seen in Table 1, there is a class imbalance in the dataset, where there are fewer wrong answers than correct answers. Besides, it can also be seen in the figure that the average answer length is relatively short, which is only around 12-13 words. Examples of answers and labels can be seen in Table 2. The part written in italics is the translation of the English text from the original Indonesian text.

Table 2. Examples of Student's Answer

Question	Student's answer	Label
A	mereka akan sulit beradaptasi (*they will find it difficult to adapt*)	Correct
A	karna mereka tidak mau terkena iklim yang sangat beser (*because they don't want to be exposed to a very large climate*)	Incorrect
B	Karena orang berpikir bahwa jika disumbangkan akan membuat produksi pakaian menjadi lebih beretika (*Because people think that donating will make clothing production more ethical*)	Correct
B	karana harga nya terjangkau dan pas. (*because the price is affordable and just right.*)	Incorrect

2.1.1 Data Preprocessing

In the preprocessing step we removed the selected frequent words from the dataset. We do not omit all frequent words because they may be keywords in the assessment. The word we omitted for both question types is 'yang', 'lebih', 'untuk', 'akan', 'mereka', and 'dan'.To handle class imbalance as seen in Table 1, we also experimented with creating synthetic data to upsampling data from a smaller class. We use Synthetic Minority Over-sampling Technique (SMOTE) [27] to synthesize new data. We use SMOTE to increase the number of examples in minority class, or in the Ukara dataset, it is in the correct answer class. Adding more examples of the data, if it can represent the real data can help improve the model's performance in classifying. SMOTE will upsample the data based on the feature vector after extracted from the data. For each question, we upsampled a 10% incorrect answer. Illustration of how SMOTE makes new data can be seen in Figure 1. SMOTE synthesize new data (small circle) between two original data (big circle) in vector space.

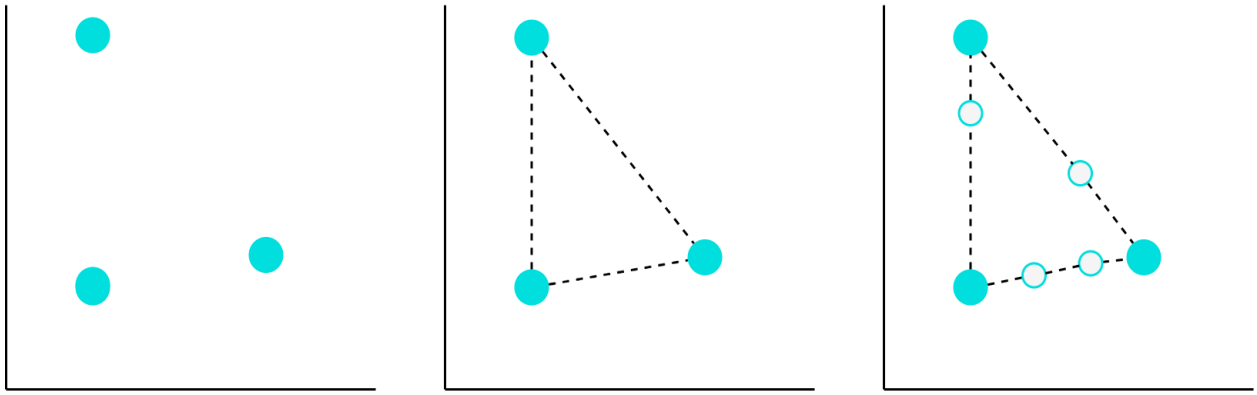

Figure 1. Illustration of How SMOTE Generates New Data (from Left to Right)

2.1.2 Dataset usage scheme

The Ukara dataset has provided three different data sets, each for training, validating, and testing purposes. The sample size for each data set is shown in Table 3. The table shows that the number of train data and validation data has similar sizes, while the test data is three times larger. In this kind of scheme, model generalization is crucial. We can also see from the table that Question B has more samples than Question A.

In Ukara 1.0 Challenge, the three sets were given in different phases. The final score on the leaderboard was taken based on the results of the test data. In this study, the three data were also used for different needs. The data usage flow can be seen in Figure 2. This scheme is used to avoid model overfitting against the validation or test data. We were also trying to imitate the data usage scheme in the Ukara 1.0 challenge. From Figure 2, it can be seen hyperparameter tuning only conducted using train set. Then we chose from each model, the best 11 parameter configurations to test using validation set. From the results obtained using validation set we selected the best 7 parameter configurations to be tested using test data.

Table 3. The Size of Train, Validation, and Test Set

	Question A	Question set B
# samples in train set	268	305
# samples in validation set	215	244
# samples in test set	855	974

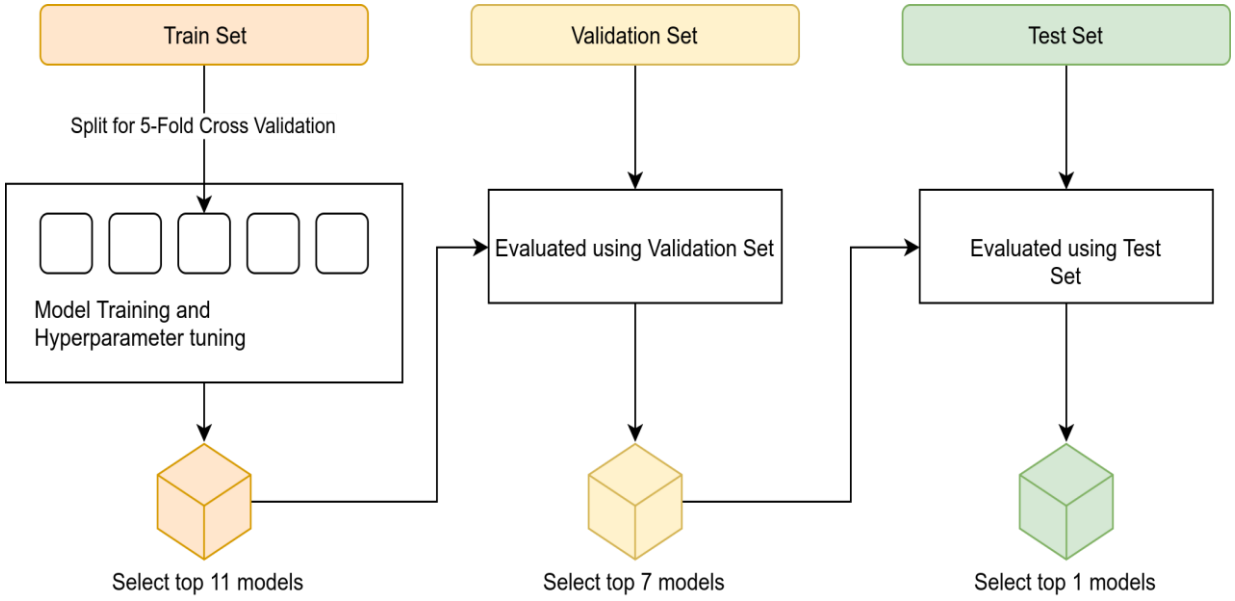

Figure 2. Data Usage Scheme

2.2 BERT Sentence Embeddings

BERT sentence embedding (SBERT) is a method for mapping a sentence to the vector space proposed in [25][26] by using Bidirectional Encoder Representations from Transformers (BERT) architecture [24]. SBERT uses the Siamese BERT network to get sentence embedding. Two sentences that have the same semantic meaning will have adjacent vector distances.

In this study, a pre-trained model of "paraphrase-xlm-r-multilingual-v1" was used, which was available on the SBERT homepage[2]. This pre-trained model is a model that has been trained in 50+ languages, including Indonesian. The model is trained by paraphrasing data so that it can see similarities in sentences. SBERT will generate a vector embedding of 768 elements for each input sentence.

Compared to the fastText used in the previous study [3], SBERT has richer feature. BERT has the advantage of paying attention to the context of a word when processing its embedding, which is called contextual embedding. A word can have different vector embedding depending on the whole sentence. In addition, the same as fastText, which has the advantage of handling Out-of-Vocabulary (OOV) words, BERT can also handle OOV even though with a different approach, namely with word piece tokenization. The file size of the pre-trained fastText model for the Indonesian language is only about 1GB, much smaller than the pre-trained fast text model for Indonesian, which reaches 4GB.

2.3 Neural Network Model

In this study we use an artificial neural network as a classifier. We chose neural network based on several previous studies that use neural network or deep learning approaches [3][9][10][11][12]. However, we avoid using complex classifiers like multi layers deep neural network in order to focus on evaluating the features used. In addition, the use of complex classifiers can make it more difficult for future deployments.

We propose to use a single hidden layer neural network model with 64 neurons and an output layer with two neurons representing the labels. We chose the single hidden layer neural network model because it is simpler than the previous stacking model [3], but it is considered capable of capturing data complexity. The number of neurons randomly selected from several trials at the beginning. We choose a small number of neurons but high performance. Each neuron uses Rectified Linear Unit (ReLU) activation function, and the optimization algorithm uses Adam optimizer [28]. The training process will use a mini-batch mode with the number of batches, and also, the number of epoch will be searched using hyperparameter optimization technique. The neural network model is implemented using PyTorch [29].

[2] https://www.sbert.net/

2.3.1 Model Generalization

Due to the large number of input neurons and small amount of data, we added some additional functions to avoid overfitting. We added a dropout function [30] on the hidden layer and weight decay to avoid large weight values. Dropout will randomly deactivate neurons in a layer during the training process. The model will attempt to generalize the problem and avoid overfitting by deactivating these neurons during training. In this study, the neurons had a 40% probability of being deactivated. The weight decay method we use in this research is not L2 regularization, which adds a penalty to the cost function but rather the decoupled weight decay proposed in [31]. This is because the optimization algorithm used is an adaptive optimizer, Adam.

We also added an incremental batch size technique to avoid overfitting [32]. In this technique, the number of samples in one batch will increase after several iterations. We make this technique an option that can be used or not during training. The usage, the number of increased samples, and the frequency of addition will be determined using hyperparameter optimization.

2.3.2 Hyperparameter Optimization

The process of searching for hyperparameters can be difficult and also time-consuming. Therefore, in this study, we use the hyperparameter optimization algorithm to find parameters automatically. We use the Tree-structured Parzen Estimator (TPE) algorithm with the Optuna framework [33] to find the best hyperparameter value based on the hyperparameter search space shown in Table 4. TPE uses a bayesian approach in finding the hyperparameter value that maximizes the objective score. This study used the averaged F1-Score across every fold as the objective score for hyperparameter optimization.

Table 4. Hyperparameters Search Space

Hyperparameters	Search space
Epochs	$\{k \mid k \in \{10, 55\}\}$
Initial batch size question A	$\{k \mid 2k \in \{2, 8\}\}$
Initial batch size question B	$\{k \mid k \in \{1, 12\}\}$
batch size increase	$\{k \mid k \in \{0, 4\}\}$
frequency of increasing the batch size	$\{k \mid k \in \{2, 4\}\}$
SMOTE usage	$\{k \mid k \in \{True, False\}\}$

One of the top teams used hyperparameter optimization technique during the Ukara Challenge to find the best value in their deep learning model [8]. The framework used at that time was Hyperopt [34].

2.4 Evaluation method

Evaluation is carried out to measure the model's performance either on the train set, validation set, or test set. The final evaluation is calculated using F1-score from the test set question A and question B. The results of this final evaluation are presented in this report. F1-score is calculated using Equation 1.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{1}$$

The choice of using F1-score as evaluation metrics is to pay attention to the model's ability to handle imbalanced data classes. Besides, by using the F1-score as the evaluation metrics, we can compare the resulting model with other published models using the Ukara dataset.

3. Results and Discussion

3.1 The effect of dropout function

We compare two models with the same features, architecture, and hyperparameter search space. However, one model uses a dropout function in its hidden layer, with a 40% probability of neurons being disabled. In contrast, other models do not use a dropout function.

In accordance with the scheme in Figure 2, The best parameters for both models are searched using Optuna to obtain 100 parameter configurations. The top 11 models will be evaluated using validation data. In this experiment, we did not use test data and only used question A to compare. We averaged the F1-score validation results for the best 11 models, and the results were shown in Table 5. From the table, you can see that the dropout function helps improve the model performance.

Table 5. Performance Comparison of Dropout Usage in the Model

Models	Avg. validation score	Maximum score	Minimum score
Model with dropout	**0.9188**	**0.9221**	**0.9136**
Model without dropout	0.9096	**0.9221**	0.8889

3.2 Distance feature

As mentioned in the previous research [12], it is possible to use use the distance between the embedding of two sentences as a feature. This experiment explored the distance embedding feature, a vector calculated from the absolute difference between the answer embedding and the guideline embedding as shown in Equation 2. We modified the sentences in the guidelines so that they are more straightforward. In Equation 2, *emb()* function is a function to convert a sentence into an embedding vector.

In this experiment, we tested three models. All models use the dropout function because it has been shown to improve model performance from previous experiments. The first model is a model that uses the sentence embedding feature of the answer without using the distance feature. The second model is a model that only uses the embedding distance feature. As for the third model, we use both features so that each sample's feature length is twice as long.

$$dist(answer, guideline) = |\, emb(answer) - emb(guideline) \,| \quad (2)$$

We perform the same experimental stages as in the previous dropout function experiment. The evaluation was carried out using only validation data and was carried out on question A only. The results can be seen in Table 6. In the table, it appears that the distance vector feature is less helpful in improving model performance.

Table 6. Performance Comparison of Distance Embedding Feature

Models	Avg. validation score	Maximum score	Minimum score
Only answer embedding	**0.9188**	0.9221	**0.9136**
Only distance embedding	0.9128	**0.9241**	0.9057
Answer embedding and distance embedding	0.9154	0.9197	0.9107

3.3 Comparison with previous models

The best model we proposed using SBERT and single layer neural network has the parameter configuration shown in Table 7. This parameter configuration is obtained after searching using Optuna and through a series of evaluations as illustrates in Figure 2. The results we report here are the results of the evaluation using test data.

Table 7. Best Hyperparameters Configuration

	# epochs	# init. batch size	# batch increase	# increase freq.	SMOTE
Question A	13	16	4	3	False
Question B	5	5	-	-	True

In Table 7, It appears that in the selected parameters, the model does not use SMOTE to synthesize the data in question A but uses SMOTE in question B. This difference configuration could be due to differences in the characteristics of the two questions. It is also indicated by the incremental batch size method. In question A, the best model increases the batch size three times during training with four samples for each addition, while in question B, the best model does not use the incremental batch size. If you look at the evaluation results, it can be seen in Table 8, that question B is more difficult to assess by the model.

Table 8. Performance Comparison for Each Question

	Question A	Question B
Validation F1- score	0.916	0.709
Test F1-score	0.894	0.757

Our best model by using SBERT and single layer neural network has surpassed previously published models. The table comparing our model with the previous model is shown in Table 9. We use F1-score for Bidirectional LSTM model based on [8]. Meanwhile, we took F1-score of the random forest - logistic regression model from the Ukara published scoreboard.

Table 9. Performance Comparison of Published Models

Models	F1-score
SBERT + NN (ours)	**0.829**
fastText + Stacking	0.821
Bidirectional LSTM	0.811
TFIDF + Random forest, Logistic regression	0.81

4. Conclusion

There are many challenges in making Automatic Essay Scoring models, some of which are data limitations and imbalanced classes. This study proposes using a simpler model to create a robust Automatic Essay Scoring for Indonesian language. We present the use of BERT sentence embedding and a single hidden layer neural network as classifiers for the Ukara dataset. In addition, our best models also use dropout, decoupled weight decay, incremental batch size, and SMOTE to improve model performance and reduce overfitting. The model we produce has a better F1-score than the previous models, which is 0.829. In future research, we can focus more on dealing with small and unbalanced data so that the model performance can be even better.

References

[1] B. W. Tuckman, "The Essay Test: A Look at the Advantages and Disadvantages," *NASSP Bulletin*, vol. 77, no. 555, pp. 20–26, Oct. 1993. https://doi.org/10.1177%2F019263659307755504

[2] G. B. Herwanto, Y. Sari, B. N. Prastowo, M. Riasetiawan, I. A. Bustoni, and I. Hidayatulloh, "UKARA: A fast and simple automatic short answer scoring system for Bahasa Indonesia," in *Proceeding Book of 1stInternational Conference on Educational Assessment and Policy*, 2018, vol. 2, pp. 48–53. https://doi.org/10.26499/iceap.v2i1.95

[3] R. A. Rajagede and R. P. Hastuti, "Stacking Neural Network Models for Automatic Short Answer Scoring," *arXiv preprint arXiv:2010.11092*, 2020.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794. https://doi.org/10.1145/2939672.2939785

[5] É. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning Word Vectors for 157 Languages," presented at the the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 07, 2018.

[6] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997. https://doi.org/10.1109/78.650093

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[8] A. A. Septiandri, Y. A. Winatmoko, and I. F. Putra, "Knowing Right from Wrong: Should We Use More Complex Models for Automatic Short-Answer Scoring in Bahasa Indonesia?," in *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, 2020, pp. 1–7. http://dx.doi.org/10.18653/v1/2020.sustainlp-1.1

[9] K. Taghipour and H. T. Ng, "A neural approach to automated essay scoring," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 1882–1891. https://doi.org/10.18653/v1/d16-1193

[10] B. Riordan, A. Horbach, A. Cahill, T. Zesch, and C. Lee, "Investigating neural architectures for short answer scoring," in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 2017, pp. 159–168. http://dx.doi.org/10.18653/v1/W17-5017

[11] F. Dong, Y. Zhang, and J. Yang, "Attention-based recurrent convolutional neural network for automatic essay scoring," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 153–162. https://doi.org/10.18653/v1/k17-1017

[12] G. Liang, B.-W. On, D. Jeong, H.-C. Kim, and G. S. Choi, "Automated Essay Scoring: A Siamese Bidirectional LSTM Neural Network Architecture," *Symmetry*, vol. 10, no. 12, Art. no. 12, Dec. 2018. https://doi.org/10.3390/sym10120682

[13] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751. https://doi.org/10.3115/v1/d14-1181

[14] A. Hassan and A. Mahmood, "Deep learning for sentence classification," in *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2017, pp. 1–5. https://doi.org/10.1109/lisat.2017.8001979

[15] A. F. Hidayatullah, S. Cahyaningtyas, and R. D. Pamungkas, "Attention-based CNN-BiLSTM for Dialect Identification on Javanese Text," *KINETIK*, pp. 317–324, Nov. 2020. https://doi.org/10.22219/kinetik.v5i4.1121

[16] Y. Kumar, S. Aggarwal, D. Mahata, R. R. Shah, P. Kumaraguru, and R. Zimmermann, "Get IT Scored Using AutoSAS—An Automated System for Scoring Short Answers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 9662–9669. https://doi.org/10.1609/aaai.v33i01.33019662

[17] M. A. Sultan, C. Salazar, and T. Sumner, "Fast and easy short answer grading with high accuracy," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1070–1075. http://dx.doi.org/10.18653/v1/N16-1123

[18] A. A. P. Ratna, B. Budiardjo, and D. Hartanto, "SIMPLE: System Automatic Essay Assessment for Indonesian Language Subject Examination," *Makara Journal of Technology*, vol. 11, no. 1, p. 2, 2007. https://doi.org/10.7454/mst.v11i1.435

[19] T. A. Roshinta and F. Rahutomo, "Analisis Aspek-Aspek Ujian Esai Daring Berbahasa Indonesia," *Prosiding Sentrinov (Seminar Nasional Terapan Riset Inovatif)*, vol. 2, no. 1, Art. no. 1, Oct. 2016.

[20] F. Rahutomo and T. Roshinta, "Indonesian Query Answering Dataset for Online Essay Test System," vol. 1, Aug. 2018. http://dx.doi.org/10.17632/6gp8m72s9p.1

[21] H. R. Acharya, A. D. Bhat, K. Avinash, and R. Srinath, "LegoNet - classification and extractive summarization of Indian legal judgments with Capsule Networks and Sentence Embeddings," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 2, pp. 2037–2046, Aug. 2020. https://doi.org/10.3233/JIFS-179870

[22] J. YU and J. JIANG, "Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification," *EMNLP 2016: Proceedings of the Conference on Empirical Methods in Natural Language Processing: Austin, Texas, November 1-5*, pp. 236–246, Nov. 2016. http://dx.doi.org/10.18653/v1/D16-1023

[23] V. Indurthi, B. Syed, M. Shrivastava, N. Chakravartula, M. Gupta, and V. Varma, "FERMI at SemEval-2019 Task 5: Using Sentence embeddings to Identify Hate Speech Against Immigrants and Women in Twitter," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minneapolis, Minnesota, USA, 2019, pp. 70–74. http://dx.doi.org/10.18653/v1/S19-2009

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[25] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3973–3983. http://dx.doi.org/10.18653/v1/D19-1410

[26] N. Reimers and I. Gurevych, "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020, pp. arXiv-2004.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002. https://doi.org/10.1613/jair.953

[28] D. P. Kingma, "Adam: A Method for Stochastic Optimization.," presented at the 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, San Diego, CA, USA, 2015.

[29] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8026–8037.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, Art. no. 1, 2014.

[31] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[32] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't Decay the Learning Rate, Increase the Batch Size," presented at the 6th International Conference on Learning Representations, {ICLR} 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

[33] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA, Jul. 2019, pp. 2623–2631. https://doi.org/10.1145/3292500.3330701

[34] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*, 2013, vol. 13, pp. 20. https://doi.org/10.25080/Majora-8b375195-003