

Indonesian automatic short answer grading system

Heinrich Reagan Salim, Chintya De, Nicholas Daniel Pratamaputra, Derwin Suhartono

Department of Computer Science, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article Info

Article history:

Received Dec 27, 2021

Revised Mar 4, 2022

Accepted May 17, 2022

Keywords:

Artificial intelligence

Automatic short answer grading

BERT

Natural language processing

Ridge regression

ABSTRACT

Short answer question is one of the methods used to evaluate student cognitive abilities, including memorizing, designing, and freely expressing answers based on their thoughts. Unfortunately, grading short answers is more complicated than grading multiple choices answers. For that problem, several studies have tried to build an artificial intelligence system called automatic short answer grading (ASAG). We tried to improve the accuracy of the ASAG system at scoring student answers in Indonesian by enhancing the earlier state-of-the-art models and methods. They were the bidirectional encoder representations from transformer (BERT) with fine-tuning approach and ridge regression models utilizing advanced feature extraction. We conducted this study by doing stages of literature review, data set preparation, model development, implementation, and comparison. Using two different ASAG data sets, the best result of this study was an achievement of 0.9508 in pearson's correlation and 0.4138 in root-mean-square error (RMSE) by the BERT-based model with the fine-tuning approach. This result outperformed the results of the previous studies using the same evaluation metrics. Thus, it proved our ASAG system using the BERT model with fine-tuning approach can improve the accuracy of grading short answers.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Heinrich Reagan Salim

Department of Computer Science, School of Computer Science, Bina Nusantara University

Jl. K H. Syahdan No. 9 Kemanggisian, Palmerah, West Jakarta 11480, Indonesia

Email: heinrich.salim@binus.ac.id

1. INTRODUCTION

Educators play a significant role in forming competent young generations. In addition to educating, evaluating students is also a task of an educator. The evaluation process is crucial in measuring the students' cognitive ability to understand the learning objectives. Several grading methods, such as multiple choices, short answers, and essays, are used for the evaluation process [1]. The short answer grading method can evaluate students' cognitive abilities, including memorizing, designing, and freely expressing answers based on their thoughts [2], better than the multiple choices method. Unfortunately, grading short answers is more complicated than grading multiple choices answers manually. The enormous number of students and answer variations increase the time needed in grading these answers [3].

Figure 1 showed a large gap between the number of Indonesian educators and students. It was not surprising that educators were overwhelmed and made mistakes while carrying out the grading task, seeing this large gap. These problems prompted the development of an automatic short answer grading (ASAG) system in this study.

The machine learning model was used quite often in ASAG system studies [4], [5]. One of the studies that proved the effectiveness of using a ridge regression model for the ASAG task was [6]. The bidirectional encoder representations from transformer (BERT) language modeling architecture with a feature-based approach in the study carried out by Gaddipati *et al.* [7] achieved 0.318 in pearson's correlation and 1.057 in

root-mean-square error (RMSE). It still could not outperform the ridge regression model developed by Sultan *et al.* [6] that achieved 0.592 in pearson's correlation and 0.887 in RMSE. Therefore, this study tried to build a BERT model with a different approach according to [8], which is fine-tuning, then compared the model with the ridge regression model.

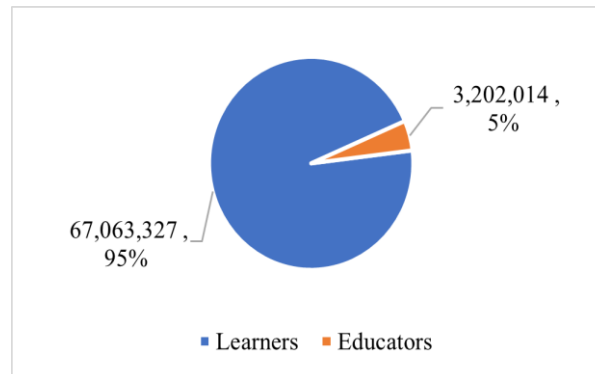


Figure 1. The statistic report on the number of students and educators in Indonesian schools and universities [10]–[14]

Just like the rarity of related studies, the Indonesian ASAG data set was also hard to obtain. One of the studies that provided an open Indonesian ASAG data set was [9]. However, some of the data set's characteristics were not suitable for our research, such as the number of responses to each question being too few and the sentences being too long. Therefore, it encouraged the design and construction of a more relevant data set for the ASAG task in this study, which has more responses to each question with shorter sentences.

2. METHOD

This study aims to develop an Indonesian ASAG system. The method consists of four stages: the data set preparation, the BERT and ridge regression models development, the analysis of each model's characteristics, and the performance comparison of these models. Parameters and configurations variations were chosen and observed for the experiment to understand their impact on the accuracy of the ASAG system. We did this series of stages to determine the best model and configuration for the Indonesian ASAG system.

2.1. Data set preparation

For the data set preparation, we used two data sets: a ready-to-use data set, which will further be referred to as the Rahutomo data set, and an own-built data set, which will further be referred to as the basic programming data set. Rahutomo data set consists of 40 pairs of questions and answers divided into four categories: politics, lifestyle, sport, and technology [9]. Labels were later converted to decimals with a scale of 0 to 5 to facilitate comparison to previous studies.

Because of the small number of responses for each question and the long sentences in the Rahutomo data set, we decided to build the basic programming data set to match the characteristics of the short answer needed in this research. Table 1 shows five basic programming questions with the reference answers gathered from various sources. Keywords and scoring rubrics were determined to score the manually collected responses. Keywords are words that become the main points of reference answers for each question and the scoring rubric is a tool to determine the score of the responses based on predefined criteria. We used the scoring rubric to produce consistent values for respondents' answers.

Table 2 showed the keywords for each question, whereas Table 3 showed the scoring rubric used by the annotators as the standard for scoring the respondent's answer. We collected a total of 1,800 responses from 360 respondents, each answering those five questions in Indonesian with a maximum length of 50 words. The respondents were undergraduate computer science students at Bina Nusantara University and IT division employees of PT Bank Central Asia Tbk using Google Forms. Three annotators graded the responses in integers with a scale of 0 to 5 and used their average score as the final score. To fix the imbalance found in the number of samples for each label on each question, we carried out data set augmentation manually according to the methods mentioned in [15], such as changing some words into their synonyms and changing the order of words in a sentence.

Table 1. Questions and reference answers in the basic programming data set

No.	Question	Reference answer	Source
1	Apa yang dimaksud dengan variabel?	Variabel adalah nama yang merujuk lokasi pada memori yang digunakan untuk menyimpan suatu nilai dengan tipe data tertentu.	[16]
2	Apa yang dimaksud dengan tipe data array?	Tipe data array adalah tipe data yang menampung banyak nilai dengan tipe data yang sama pada lokasi memori yang berurutan.	[17]
3	Apa perbedaan antara tipe data integer dan float?	Integer adalah tipe data untuk menampung bilangan bulat, sedangkan float adalah tipe data untuk menampung bilangan desimal.	[16] [17]
4	Apa yang dimaksud dengan algoritma?	Algoritma adalah urutan langkah-langkah untuk menyelesaikan sebuah masalah yang disusun secara logis dan sistematis.	[18]
5	Mengapa perlu membuat pseudocode terlebih dahulu sebelum memulai coding?	Agar lebih mudah memahami dan menggambarkan algoritma program untuk mengubahnya menjadi bahasa pemrograman tertentu.	[18]

Table 2. Questions and keywords in the basic programming data set

No.	Reference answer	Keyword
1	Variabel adalah nama yang merujuk lokasi pada memori yang digunakan untuk menyimpan suatu nilai dengan tipe data tertentu.	1. Nama 2. Lokasi 3. Menyimpan 4. Nilai 5. Tipe data tertentu
2	Tipe data array adalah tipe data yang menampung banyak nilai dengan tipe data yang sama pada lokasi memori yang berurutan.	1. Lokasi memori 2. Berurutan 3. Menampung 4. Banyak nilai 5. Tipe data sama
3	Integer adalah tipe data untuk menampung bilangan bulat, sedangkan float adalah tipe data untuk menampung bilangan desimal.	1. Menampung 2. Integer/ bilangan bulat 3. Float/ bilangan desimal
4	Algoritma adalah urutan langkah-langkah untuk menyelesaikan sebuah masalah yang disusun secara logis dan sistematis.	1. Urutan 2. Langkah-langkah 3. Menyelesaikan 4. Masalah 5. Logis
5	Agar lebih mudah memahami dan menggambarkan algoritma program untuk mengubahnya menjadi bahasa pemrograman tertentu.	1. Lebih mudah 2. Menggambarkan 3. Algoritma 4. Mengubah 5. Bahasa pemrograman

Table 3. Scoring rubric for the basic programming data set

Score	Criteria (All Except Question #3)	Criteria (Question #3 Only)
5	The respondent's answer has the same five keywords as the reference answer.	The respondent's answer has the same three keywords as the reference answer.
4	The respondent's answer has the same four keywords as the reference answer.	The respondent's answer has the second and third keywords as the reference answer.
3	The respondent's answer has the same three keywords as the reference answer.	The respondent's answer has the first and second or the second and third keywords as the reference answer.
2	The respondent's answer has the same two keywords as the reference answer.	The respondent's answer has the second or third keywords as the reference answer.
1	The respondent's answer has the same one keyword as the reference answer.	The respondent's answer has the first keywords as the reference answer.
0	The respondent's answer is empty or irrelevant to the reference answer.	The respondent's answer is empty or irrelevant to the reference answer.

We divided each data set into training and testing data sets with in-domain configuration. Before we divided them, we removed all the zero-scored responses because the model did not need to learn incorrect answers. The characteristics were also too diverse to learn. The Rahutomo data set was divided with a ratio of 85:15, containing 1,855 samples for the training data set and 307 samples for the testing data set. Each data set consists of an equal number of answers from each question of all categories. After removing the zero-scored responses, the basic programming data set contains a total of 1,815 samples. The data set was divided with a ratio of 80:20, containing 1,452 samples for the training data set and 363 samples for the testing data set. Each data set consists of an equal number of samples for each question.

2.2. Bidirectional encoder representations from transformer model development

We utilized eight pretrained models from three different literature to develop our ASAG system for Indonesian. The models consisted of six BERT models and two ALBERT models, considering the data set language used during their pretraining phase was Indonesian. Table 4 shows each model's abbreviation used as references in this study and their sources.

Table 4. BERT pretrained models

Score	Abbreviation	Source
indobenchmark/indobert-base-p1	Base-P1	[19]
indobenchmark/indobert-base-p2	Base-P2	
indobenchmark/indobert-large-p1	Large-P1	
indobenchmark/indobert-large-p2	Large-P2	
indobenchmark/indobert-lite-base-p2	Lite-Base-P2	
indobenchmark/indobert-lite-large-p2	Lite-Large-P2	
cahya/bert-base-indonesian -522 M	Cahya	[20]
indolem/indobert-base-uncased	IndoLEM	[21]

2.2.1. Indobenchmark/IndoBERT

This literature developed both BERT and ALBERT models called IndoBERT and IndoBERT-lite. For each model, there were base and large versions. The BERT models were developed based on [19] and the ALBERT models according to [22]. Table 5 shows the summary of the difference between each model and version [19].

Table 5. The details of Indobenchmark models

Model	#PARAMS	#Layers	#Heads	Emb. Size	Hidden Size	FFN Size	Language Type	Pre-train Emb. Type
Lite-Base	11.7M	12	12	128	768	3,072	Mono	Contextual
Base	124.5M	12	12	768	768	3,072	Mono	Contextual
Lite-Large	17.7M	24	16	128	1,024	4,096	Mono	Contextual
Large	335.2M	24	16	1,024	1,024	4,096	Mono	Contextual

2.2.2. Cahya/BERT-base-Indonesian-522M

BERT-base-Indonesian-522M was a BERT pretrained model trained using Indonesian data set sourced from a 522 MB-sized wikipedia. It was an uncased BERT model, which means it did not distinguish between uppercase and lowercase letters. WordPiece was used to generate the vocabulary, which has a total of 32,000 words [20].

2.2.3. IndoLEM/IndoBERT-base-uncased

IndoLEM was another BERT model based on Devlin *et al.* [8], but trained purely as a masked language model without the next sentence prediction objective. Koto *et al.* [21] developed this model using the huggingface library, with the default configuration for BERT-base (uncased). It has 12 hidden layers with 768 dimensions each, 12 attention heads, and 3,072 dimensions of the feed-forward hidden layer. This model used wordpiece to generate its vocabulary of 31,923 words.

After exploring the three pretrained BERT models above, we fine-tuned each model to grade Indonesian short answer questions using the corresponding data set to optimize the parameters used in each model. We modified the BERT model's architecture by adding a regression layer above the last encoder stack. The added regression layer could then proceed the classification token [CLS] into a score of respondents' answers. We carried out the fine-tuning process by retraining the modified pretrained model with the data set explained above in a supervised manner.

2.3. Ridge regression model development

The ridge regression model consisted of three feature extraction modules to produce representation used in grading predictions. They are word alignment, term weighting, and semantic similarity. Each feature returns two prediction scores: with and without question demoting (the act of removing words found in the question from the response).

2.3.1. Word alignment

The purpose of word alignment was to align word pairs from two sentences. This feature provided essential information on the connection between those two sentences. First, we did text preprocessing to both answers by removing punctuations and lowercasing the words. After that, we converted the answers in the five-

staged parsing step: tokenization, part-of-speech (POS) tagging, dependency parsing, named entity recognition (NER), and lemmatization or stemming.

We utilized the TreebankWordTokenizer from the natural language toolkit (NLTK) library to determine the character offset of each token. For the POS tagging, we trained a POS tagger model using the CRFTagger module and a manually annotated data set of Indonesian POS tagging by Kurniawan and Aji [23] containing approximately 10,000 sentences and 250,000 tokens. We used a pretrained model from the Stanza library [24], namely the Indonesian universal dependencies (UD) model for the dependency parsing. A treebank data set named UD Indonesian graduate skills development (GSD) was used to train the model. There were around 121,923 nodes samples for general dependencies. Such as *nsubj*, *obj*, *advmod*, *obl*, *case*, and *compound* are in the data set. We used the StanfordNER model that used the data set in NERIndo research for NER model training and classification. There were two alternatives to change words into their base forms. The first was stemming using PySastrawi. The other alternative was lemmatization using LemmatizerIndo-python [25].

After those were all done, the alignment process could start following the work of Sultan *et al.* [26]. Unfortunately, paraphrase database (PPDB) used in their work is not available in Indonesian. Thus, we used *cosine similarity* to determine if a word pair is similar in paraphrase. A FastText model was used to generate word embeddings used for cosine similarity. We trained the model using the first 100,000 lines of the Indonesian version of wikipedia. Later in this research, a minimum score is defined as a threshold of word pairs determined as similar in paraphrase. Thus, we did not consider word pairs with cosine similarity values below this score as paraphrases. After completing these alignment stages, the score of the word alignment feature is calculated based on the total number of aligned content words on both sentences divided by the total number of content words on both sentences.

2.3.2. Term weighting

Term weighting also played a significant role in grading short answer responses as it helped the machine determine whether a word contributed a special meaning to the given sentence. We experimented with two types of term weighting schemes: term frequency-inverse document frequency (TF-IDF) and term frequency-inverse gravity moment (TF-IGM). TF-IDF was widely used and popular in natural language processing (NLP) research. TF-IGM was the upgraded term weighting scheme used in this research. We calculated the TF weight of each term in the reference answer based on the occurrence of the word in the top 10 wikipedia articles related to the word and the search result of the links referenced by those articles. We calculated each term's IDF and IGM weights based on its occurrences in 15,000 wikipedia articles, 600 each from 25 domains of wikipedia Indonesia. They were religion, language, biography, culture, and economy. In IDF calculation, domain diversity did not affect the term weighting because it merged all articles into a single corpus. On the contrary, it affected the IGM calculation significantly [27].

There were two things to pay attention to while calculating term weight. First, we only used weights of words in the reference answer in the calculation to prevent score reduction for longer but still correct answers [12]. Second, we preprocessed the wikipedia corpus used in this process. The steps were lowercasing, punctuation removal, number removal, stopwords removal, and stemming or lemmatization. The purpose was to remove typos and errors in the corpus while doing stemming or lemmatization methods and prevent inaccuracies while calculating term weight caused by word derivatives and paraphrases.

2.3.3. Semantic similarity

The semantic similarity module tried to find word similarity between the respondent answer and the reference answer, and more importantly, compared the semantic similarity between words. A Word2Vec model was used to produce such representations. We trained this model using the Indonesian wikipedia corpus accessed on January 22nd, 2021 and then used it to train the gensim models with 300-dimension vectors. The sum of all word vectors in the sentence would produce the semantic vector of a sentence. Then, we used the cosine similarity score of both semantic vectors as a feature in the ridge regression model.

After the definition of these three features, the training phase of the ridge regression model started. The model would load the training data set and pre-calculated TF-IGM or TF-IDF weights of the reference answer. Then, each sample would be feature-extracted into two scores (with and without question demoting). A grid search was done to find the alpha hyperparameter that yielded the best RMSE value on a 10-fold cross-validation scheme. The final training and testing scores were the average RMSE on the 10-fold cross-validation with the best alpha. We evaluated the testing prediction results using pearson's correlation and RMSE.

2.4. Experiment

After developing the BERT and ridge regression model, we carried deeper analyses to each model by experimenting according to the prepared scenarios. This step aimed to discover the effect of each parameter of

the corresponding model and data set used on its performance. We trained the models using google cloud server and AI server owned by the R&D Center of Bina Nusantara University.

2.4.1. Bidirectional encoder representations from transformer model

In this study, we did the training and validation process to the BERT model with a 10-fold cross-validation scheme on the training data set for the two data sets separately. Then, we tested the models using the testing data set. We used pearson's correlation and RMSE values as the evaluation matrix to analyze the model's performance in predicting scores in training and testing phases. We conducted numerous experiments with two data sets, six pretrained BERT models, two pretrained ALBERT models, and several fine-tuning parameters, such as batch size, epoch size, learning rate, and warm-up steps. We did that to study their nature, effect, and relationship with the final performance of the BERT-based ASAG system.

Table 6 shows the detail of the scenario. For simplification purposes, we referred to each parameter used here by its respective abbreviation, such as b for batch, e for epoch, lr for learning rate, and ws for warm-up steps. The models' names were also abbreviated as shown in: i) Base-P1: indobenchmark/indobert-base-p1, ii) base-P2: indobenchmark/indobert-base-p2, iii) large-P1: indobenchmark/indobert-large-p1, iv) large-P2: indobenchmark/indobert-large-p2, v) Cahya: cahya/bert-base-indonesian-522 M, vi) IndoLEM: indolem/indobert-base-uncased, vii) lite-base: indobenchmark/indobert-lite-base-p2, and viii) lite-large: indobenchmark/indobert-lite-large-p2. For the Rahutomo data set, we used only 4, 8, and 16 batch sizes across all phases and models. In addition, the maximum number of tokens for each sample was 80 for the basic programming data set and 210 for the Rahutomo data set. It was only because of the sentence length difference in the reference and respondents' answers in the two data sets.

Table 6. BERT-based model experiment scenario with the basic programming and Rahutomo data sets

No.	Model	Batch (b)				Epoch (e)		Fold (f)		Learning rate (lr)				Warm-Up step (ws)			
		4	8	16	32	4	8	5	10	1e-5	2e-5	3e-5	4e-5	0	0.1	0.2	0.3
1	Base-P1	✓				✓			✓		✓			✓			
2	Base-P2		✓			✓			✓		✓			✓			
3	Large-P1			✓		✓			✓		✓			✓			
4	Large-P2				✓ ^a	✓			✓		✓			✓			
5	Cahya			✓			✓		✓		✓			✓			
6	IndoLEM		✓					✓			✓			✓			
7		✓				✓			✓	✓				✓			
8		✓				✓			✓			✓		✓			
9		✓				✓			✓				✓	✓			
10		✓				✓			✓		✓				✓		
11		✓				✓			✓		✓					✓	
12		✓				✓			✓		✓						✓
13	Lite-Base-P2	✓				✓			✓		✓			✓			
14	&		✓			✓			✓		✓			✓			
15	Lite-Large-P2			✓		✓			✓		✓			✓			
16					✓ ^a	✓			✓		✓			✓			
17		✓				✓			✓		✓						✓

2.4.2. Ridge regression model

As for the ridge regression models, we made scenarios of utilizing four parameters: minimum cosine similarity score, paraphrase similarity score (ppdbSim), term weighting scheme, and text preprocessing modules. We obtained 48 training results: 32 scenarios using the Rahutomo data set and 16 using the basic programming data set. Tables 7 and 8 (in appendix) showed the complete scenarios.

3. RESULTS AND DISCUSSION

3.1. Bidirectional encoder representations from transformer model performance analysis

3.1.1. Batch size analysis

In general, the increase in batch size was inversely proportional to the pearson's correlation value shown in Figure 2 and directly proportional to the RMSE shown in Figure 3 when using the basic programming data set. It meant that the smaller the batch size used, the better the performance of the BERT model. When analyzed more closely, the first four models (IndoBenchmark) had similar performances, and they were also much better than the remaining two models, namely Cahya and IndoLEM. The large version of the

IndoBenchmark models, namely large-P1 and large-P2, slightly outperformed the base version of the IndoBenchmark models, namely base-P1 and base-P2. However, as the batch size increased, both large models experienced a more significant decrease when compared to the base model. The addition of a second pretraining phase to the base and large models also produced a lower performance when compared to models that only went through one pretraining phase. In addition, there are drastic and abnormal movements, both in the pearson's correlation and the RMSE of the IndoLEM models.

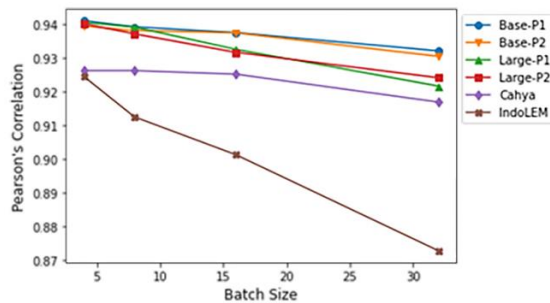


Figure 2. The effect of batch size on the average Pearson's correlation of BERT model using basic programming data set

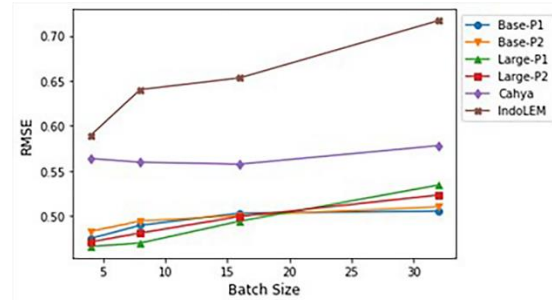


Figure 3. The effect of batch size on the average RMSE of BERT model using basic programming data set

On the other side, the performance of some models improves as the batch size increase when using the Rahutomo data set. Based on the results shown in Figures 4 and 5, the base-P2 model produced the best performance with the highest pearson's correlation value with a batch size of 16 and the lowest RMSE value with a batch size of 4.

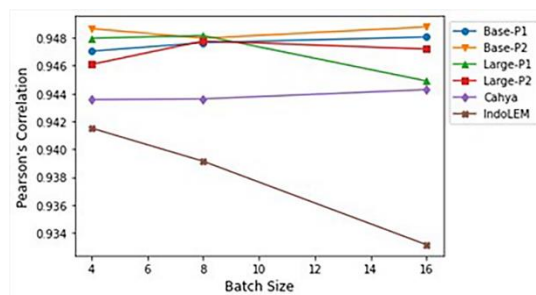


Figure 4. The effect of batch size on the average Pearson's correlation of BERT model using Rahutomo data set

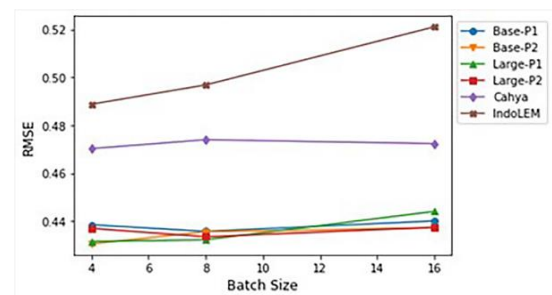


Figure 5. The effect of batch size on the average RMSE of BERT model using Rahutomo data set

3.1.2. Epoch size analysis

The increasing number of epochs in the fine-tuning process using the basic programming data set yielded various effects on the performance of each model. Based on the results in Figures 6 and 7, the number of epochs did not significantly affect the pearson's correlation and RMSE values of the base-P1, base-P2, and Cahya models. On the other hand, a notable change occurred for the large models as the number of epochs increased.

With the Rahutomo data set, the increment of the epoch size yielded a positive effect on the performances of most BERT models, except for the base-P2 and Cahya models shown in Figure 8. However, both models only experienced a slight decrease in pearson's correlation value, less than 0.003. The IndoLEM model showed quite striking results compared to the other models. It experienced the most significant change in pearson's correlation and RMSE value shown in Figure 9.

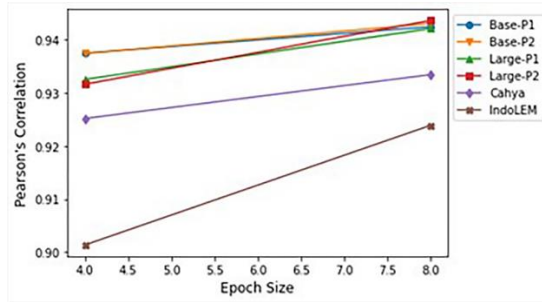


Figure 6. The effect of epoch size on the average Pearson's correlation of BERT model using basic programming data set

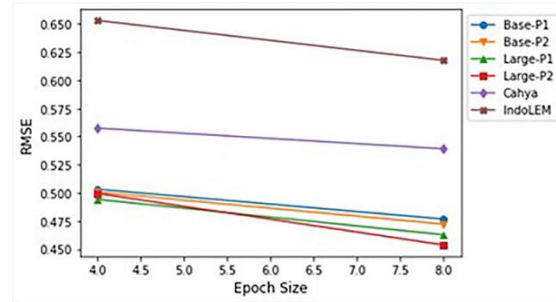


Figure 7. The effect of epoch size on the average RMSE of BERT model using basic programming data set

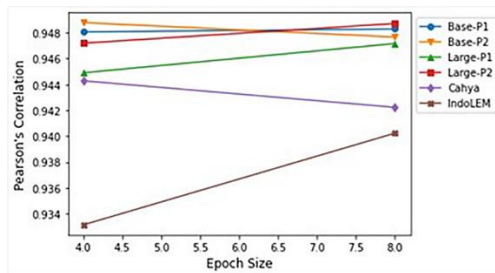


Figure 8. The effect of epoch size on the average Pearson's correlation of BERT model using Rahutomo data set

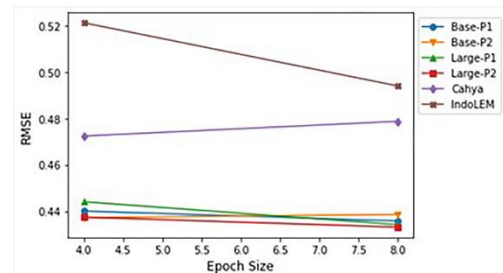


Figure 9. The effect of epoch size on the average RMSE of BERT model using Rahutomo data set

3.1.3. Learning rate analysis

The effect of learning rate had quite an impact on the performance of several BERT models using the basic programming data set. In general, the BERT models obtained impressive results starting from the $1r1e-5$ until $1r2e-5$. However, most models experience a decreasing performance with $1r2e-5$ until $1r4e-5$ in Pearson's correlation and RMSE values shown in Figures 10 and 11.

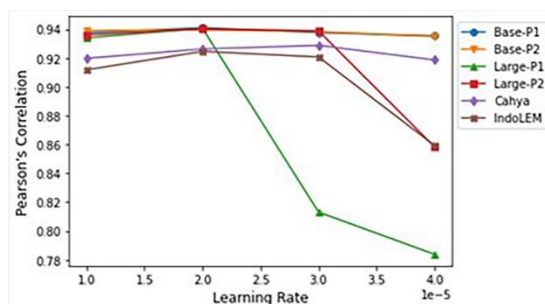


Figure 10. The effect of learning rate on the average Pearson's correlation of BERT model using basic programming data set

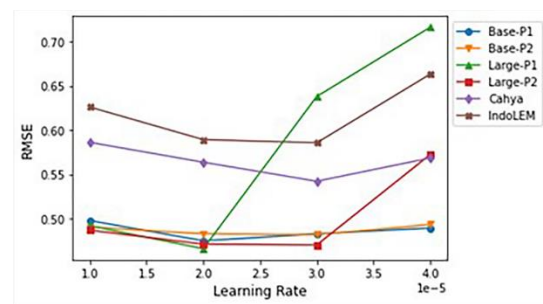


Figure 11. The effect of learning rate on the average RMSE of BERT model using basic programming data set

The experiment using the Rahutomo data set resulting in the base-P1, base-P2, Cahya, and IndoLEM models produced stable values. The four models could compete at any learning rate. On the contrary, there were contrasting differences in the Pearson's correlation and RMSE values in Figures 12 and 13 of the large-P1 and large-P2 models. Their performances decreased drastically as the learning rate increased to $4e-5$. Changes in learning rate affected the large-P1 and large-P2 models significantly, but not for the other four models.

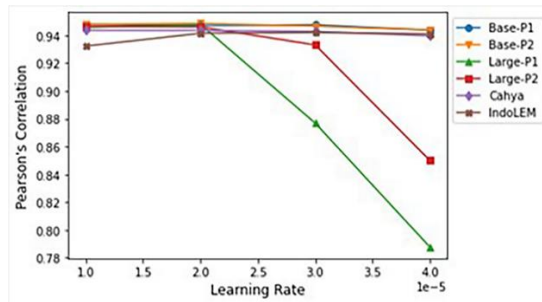


Figure 12. The effect of learning rate on the average Pearson's correlation of BERT model using Rahutomo data set

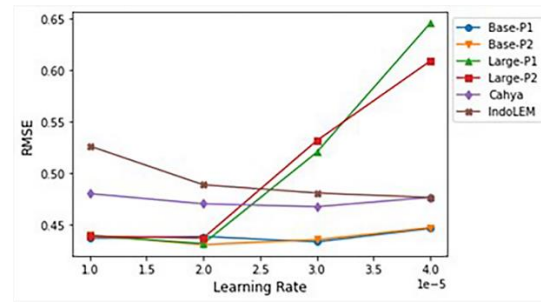


Figure 13. The effect of learning rate on the average RMSE of BERT model using Rahutomo data set

3.1.4. Warm-up step analysis

Most BERT models benefited from using warm-up steps of 30% (ws 0,3) on the basic programming data set. Warm-up steps of 10% (ws0,1) and 20% (ws0,2) reduced the models' performances slightly. On the other hand, the Cahya and IndoLEM models yielded different results in Figures 14 and 15.

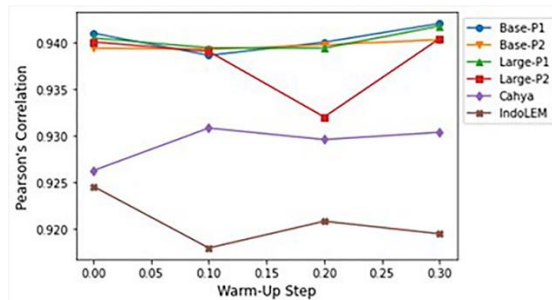


Figure 14. The effect of warm-up step on the average Pearson's correlation of BERT model using basic programming data set

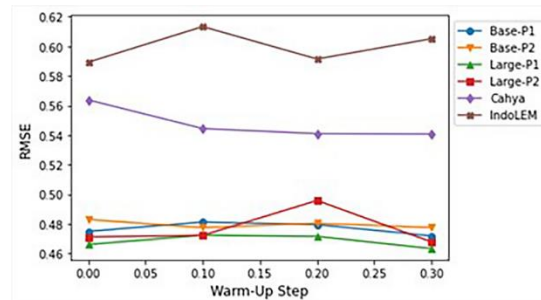


Figure 15. The effect of warm-up step on the average RMSE of BERT model using basic programming data set

Meanwhile, changes in the number of warm-up steps in the six BERT models gave quite varied results on the Rahutomo data set compared to the influence of other parameters. In general, the performance trend in Figures 16 and 17 was decreasing, but it was still on a tiny scale so that the effect of the warm-up steps was not too significant.

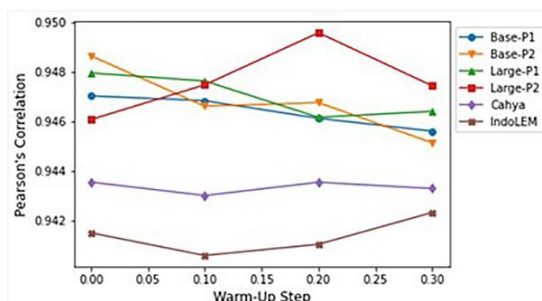


Figure 16. The effect of warm-up step on the average Pearson's correlation of BERT model using Rahutomo data set

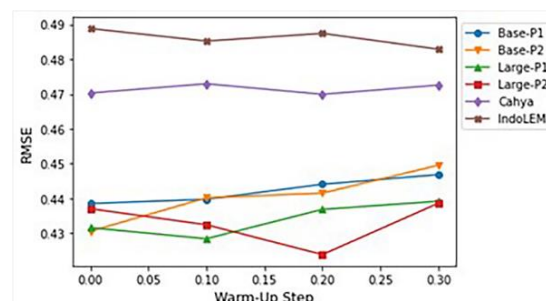


Figure 17. The effect of warm-up step on the average RMSE of BERT model using Rahutomo data set

3.2. ALBERT model performance analysis

3.2.1. Batch size analysis

In testing the batch size parameter with the basic programming data set, the ALBERT models had similar characteristics to the BERT models because the ALBERT models' performances trend shown in Figures 18 and 19 decreased with increasing batch size. From the achieved peak performance, the lite-base-P2 model managed to outperform the lite-large-P2 model with a batch size of 4. However, from the performance stability point-of-view, the lite-base-P2 model experienced more decrease in performance, and even ending with lower performance than the lite-large-P2 model with a batch size of 32.

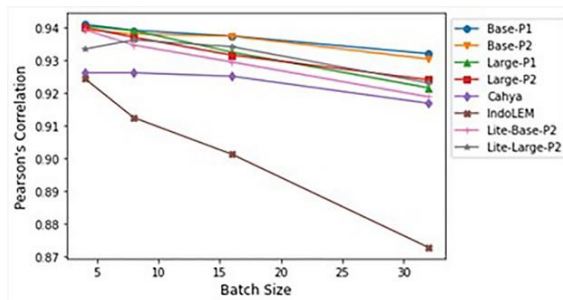


Figure 18. The effect of batch size on the average Pearson's correlation of ALBERT model using basic programming data set

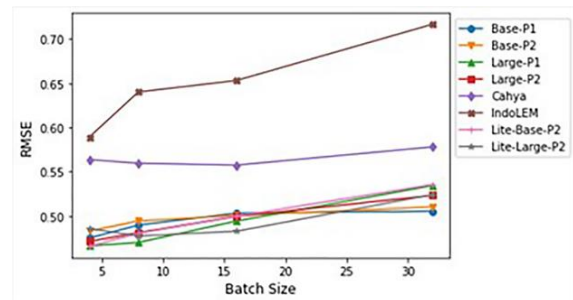


Figure 19. The effect of batch size on the average RMSE of ALBERT model using basic programming data set

The performance trends of the two ALBERT models on the Rahutomo data set were also still in line with the BERT model, which decreases with increasing batch size, as shown in Figures 20 and 21. However, the lite-base-P2 model achieved its best performance and was the best model compared to other models with a batch size of 8.

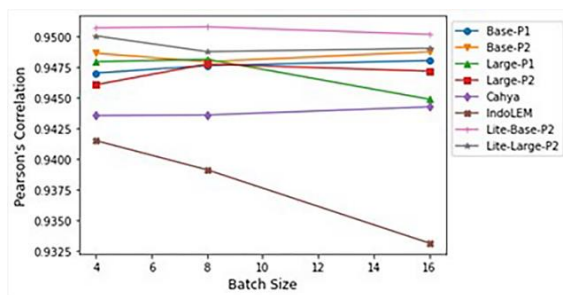


Figure 20. The effect of batch size on the average Pearson's correlation of ALBERT model using Rahutomo data set

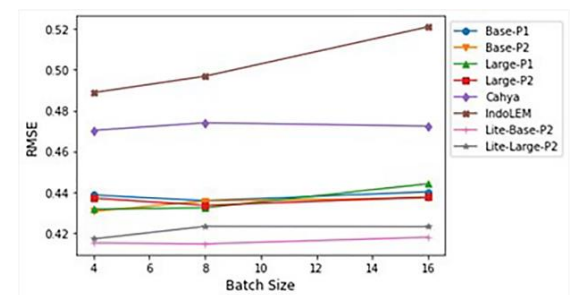


Figure 21. The effect of batch size on the average RMSE of ALBERT model using Rahutomo data set

3.2.2. Warm-up step analysis

The warm-up step had quite a different effect on the lite-base-P2 and lite-large-P2 models on the basic programming data set. As seen from Figures 22 and 23, the lite-base-P2 model tended to produce a decreasing performance while the lite-large-P2 model had an increasing performance as the number of warm-up steps increased.

Based on the results of using the Rahutomo data set in Figure 24, the change in warm-up step insignificantly affects the pearson's correlation value of the ALBERT model. The lite-large-P2 model only experienced a slight decrease, and the lite-base-P2 model was even seemed unchanged. Figure 25 also showed comparable results.

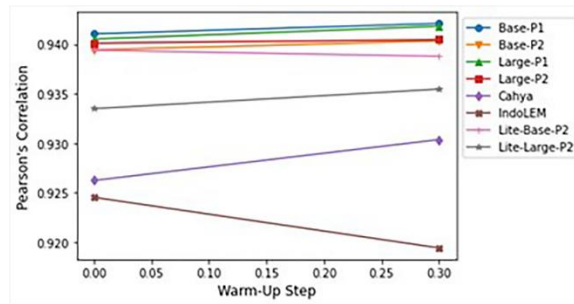


Figure 22. The effect of warm-up step on the average Pearson's correlation of ALBERT model using basic programming data set

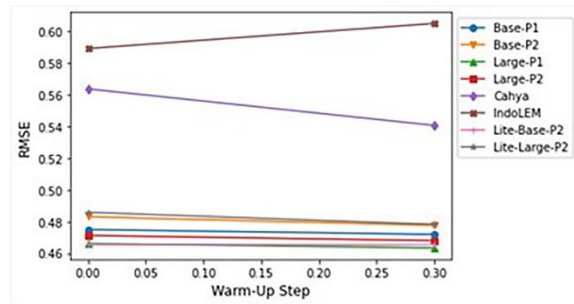


Figure 23. The effect of warm-up step on the average RMSE of ALBERT model using basic programming data set

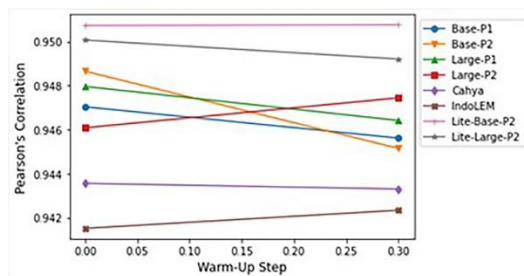


Figure 24. The effect of warm-up step on the average Pearson's correlation of ALBERT model using Rahutomo data set

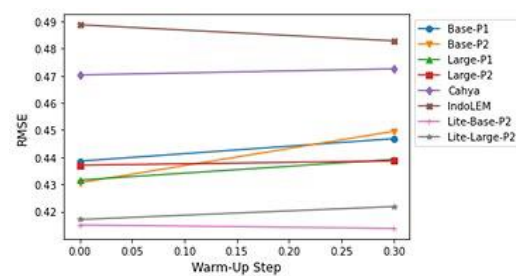


Figure 25. The effect of warm-up step on the average RMSE of ALBERT model using Rahutomo data set

3.3. Ridge regression model performance analysis

3.3.1. Minimum cosine similarity score

We find that using the seven minimum cosine similarity scores (0.625, 0.65, 0.675, 0.7, 0.75, 0.8, and 0.9) show no significant impact on the training's RMSE score while using the basic programming data set. The range of RMSE values obtained during this training is 0.015, with the minimum cosine similarity score of 0.65, bringing the best RMSE score. The increase of minimum cosine similarity scores used on the training process shows fluctuating RMSE scores in Figure 26. From Figure 27, the best result in the minimum cosine similarity score experiment using the Rahutomo data set was obtained with training using cosine similarity values of 0.7, with the RMSE value of 0.561. The increase of the minimum cosine similarity score used on training corresponds with the decrease in the RMSE value because it will increase the selectivity of word pairs to be deemed as paraphrases, thus increasing alignment accuracy.

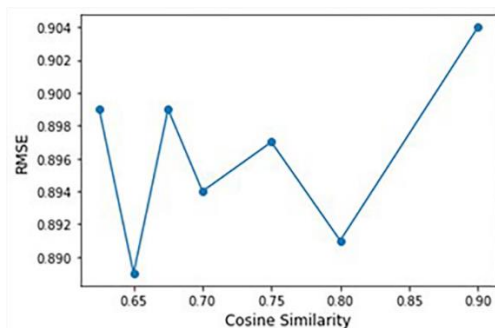


Figure 26. The effect of minimum cosine similarity score on the average RMSE of ridge regression model using basic programming data set

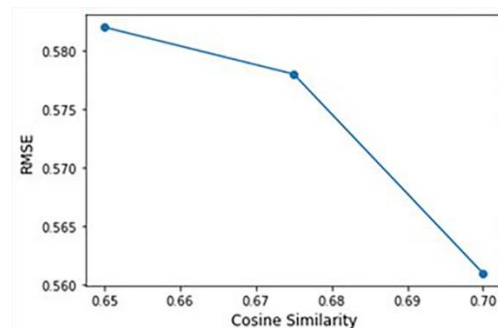


Figure 27. The effect of minimum cosine similarity score on the average RMSE of ridge regression model using Rahutomo data set

3.3.2. Paraphrase similarity score

We experimented with two ppdbSim values: a constant value of 0.9 for every word pair with a cosine similarity value of more than or equal to the minimum cosine similarity score and an inconstant ppdbSim value equals to the cosine similarity value of the word pair. Table 9 shows the results of using those ppdbSim values with the basic programming data set. We found that training the model with the cosine similarity ppdbSim value yields no better RMSE value in both term-weightings than the worst result obtained using the ppdbSim value of 0.9. One of the factors contributing to such outcomes is the noise present while contextually aligning words. Such noises appeared while training using the cosine similarity ppdbSim value. As there are no minimum value thresholds, many word pairs with low cosine similarity values will still have a wordSim value larger than 0, and thus generating disrupting noises that will be aligned one way or another.

While using the Rahutomo data set, using cosine similarity value as ppdbSim shows a better RMSE value than using a constant value of 0.9 as the ppdbSim value for every word pair with a minimum cosine similarity score threshold. The difference is 0.023 for both term weighting schemes. Table 10 shows the results. The use of cosine similarity as the paraphrase similarity also helps to pair the words with a cosine similarity value below the minimum threshold set.

Table 9. The effect of paraphrase similarity score (ppdbSim) on the average RMSE of ridge regression model using basic programming data set

Term weighting scheme	ppdbSim	Minimum cosine similarity	RMSE
TF-IGM	0.9	0.65	0.889
		0.9	0.904
TF-IDF	Cosine similarity	-	0.904
	0.9	0.675	0.889
		0.9	0.902
	Cosine similarity	-	0.903

Table 10. The effect of paraphrase similarity score (ppdbSim) on the average RMSE of ridge regression model using Rahutomo data set

Term weighting scheme	ppdbSim	Minimum cosine similarity	RMSE
TF-IGM	0.9	0.675	0.582
		0.7	0.561
		-	0.538
TF-IDF	0.9	0.65	0.581
		0.7	0.562
		-	0.539

3.3.3. Term weighting

The performance of the TF-IDF term weighting scheme yields better results than the TF-IGM while trained using the basic programming data set. Nevertheless, the difference is not significant. The maximum difference between them is just 0.009, with an experiment configuration of minimum cosine similarity score of 0.675, ppdbSim of 0.9, and stemming modules. Figure 28 shows the line graphs of the results. The usage of the TF-IDF term-weighting scheme on the Rahutomo data set yields better results compared to TF-IGM term-weighting scheme, like the results obtained on the basic programming data set in Figure 29.

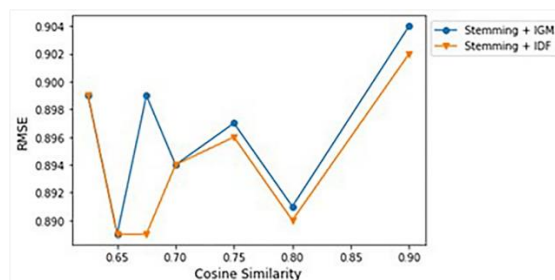


Figure 28. The effect of term weighting scheme on the average RMSE of ridge regression model using basic programming data set

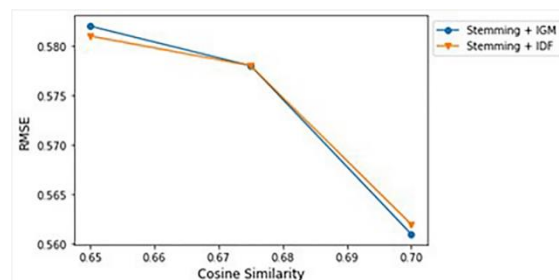


Figure 29. The effect of term weighting scheme on the average RMSE of ridge regression model using Rahutomo data set

3.3.4. Text preprocessing

The lemmatization module decreases the RMSE value of the training process on the basic programming data set. The ridge regression model achieves the most significant decrease in the RMSE value with a minimum cosine similarity score of 0.9 and ppdbSim value of 0.9. The module produces a 0.021 decrease in RMSE value while using the TF-IGM weighting scheme and 0.02 while using the TF-IDF weighting scheme. We also obtained a similar result from the model trained using cosine similarity value as the ppdbSim value, a 0.021 decrease in RMSE value with either TF-IDF or TF-IGM weighting scheme (see Table 11 and Figure 30).

Table 11. The effect of text preprocessing module on the average RMSE of ridge regression model using basic programming data set

Text preprocessing module	Term weighting scheme	RMSE
Stemming	IGM	0.904
	IDF	0.903
Lemmatization	IGM	0.883
	IDF	0.882

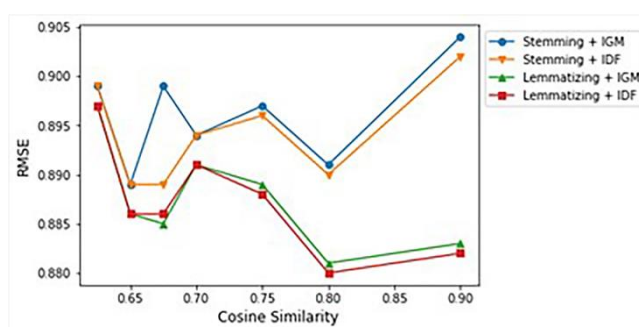


Figure 30. The effect of text preprocessing module on the average RMSE of ridge regression model using basic programming data set

Contrary to the results above, the lemmatization module used while training using the Rahutomo data set shows inferior results than the stemming module (See Table 12 and Figure 31). It is because both data sets' characteristics are different. There are more foreign words in the Rahutomo data set than the basic programming data set.

Table 12. The effect of text preprocessing module on the average RMSE of ridge regression model using Rahutomo data set

Text preprocessing module	Term weighting scheme	RMSE
Stemming	IGM	0.538
	IDF	0.539
Lemmatization	IGM	0.562
	IDF	0.562

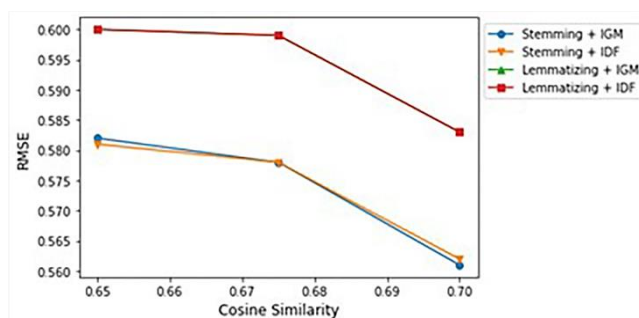


Figure 31. The effect of text preprocessing module on the average RMSE of ridge regression model using Rahutomo data set

3.4. Comparison

We compared the best results from both models to find the best model with its configuration. Table 13 shows that the BERT models' performances outperform the ridge regression with a large gap. Three main factors that contribute to those results are the transfer learning concept of BERT, the feature extraction process, and the preprocessing steps of each model.

Table 13. Comparison of the best results of BERT and ridge regression models

Score	Basic programming data set				Rahutomo data set			
	BERT (Large-P2)		Ridge regression		ALBERT (Lite-Base-P2)		Ridge regression	
	Train	Test	Train	Test	Train	Test	Train	Test
Pearson's ρ	0.9437	0.9530	-	0.7790	0.9508	0.9483	-	0.9164
RMSE	0.4537	0.4294	0.88	0.8752	0.4138	0.4277	0.538	0.536

Transfer learning ability is beneficial to the BERT model. The BERT model effectively gains linguistic patterns and vocabulary through the pretraining phase using large data sets on diverse topics. Manual feature formulation and dependency on the feature extraction modules are some drawbacks of the ridge regression model. The model needs the feature extraction modules to generate representation or characteristics of texts so the model can understand and produce predictions based on them. However, every language has its own grammatical rules, thus making the modules language-sensitive. The ridge regression model needs some preprocessing steps before training. These steps include stemming or lemmatization and stopwords removal. Moreover, in several cases, stopwords or affixes possess significant semantical meaning to the words in the sentence.

3.5. Evaluation

We conducted objective and subjective evaluations for the Indonesian ASAG system. We compared its performance to the human correlation score shown in Table 14 and earlier studies shown in Table 15 in the objective evaluation phase. Based on the results shown in Table 14, the pearson's correlation obtained by both human annotators and the Indonesian ASAG shows an insignificant difference. Therefore, the ASAG has the potential to assist humans in grading short answers.

Table 14. Comparison of the best results of Indonesian ASAG system and human annotators

Data set	Annotator	Pearson's ρ	RMSE
Basic programming	Human	0.9565	0.4193
	Indonesian ASAG system (Large-P2)	0.9437	0.4537
Rahutomo	Human	0.9853	0.2454
	Indonesian ASAG system (Lite-Base-P2)	0.9508	0.4138

Table 15. Comparison of the best results of Indonesian ASAG system and earlier studies

Research source	Model	Pearson's ρ		RMSE	
		Train	Test	Train	Test
Indonesian ASAG system	large-p2	0.9437	0.953	0.4537	0.4294
	lite-base-p2	0.9508	0.9483	0.4138	0.4277
[28]	bert-base-multilingual-cased	-	-	-	1.893
	bert-base-multilingual-uncased	-	-	-	1.787
[6]	Ridge Regression ^a	-	0.592	-	0.887
	Ridge Regression	-	0.63	-	0.85
[29]	SVMRank ^a	-	0.518	-	0.998
	SVR ^a	-	0.464	-	0.978
[7]	ELMO	-	0.485	-	0.978
	BERT	-	0.318	-	1.057
	GPT	-	0.248	-	1.082
	GPT-2	-	0.311	-	1.065

Table 14 also shows that the performance of the best model of BERT can match the best pearson's correlation score of the human annotator on training using the basic programming data set. We can conclude it by seeing the almost-identical pearson's correlation and RMSE obtained by the large-P2 with a fine-tuning configuration of b16, e8, lr2e-5, and ws0. However, the same does not apply to training using the Rahutomo data set, where the difference between the performance of the human annotator and the best ALBERT model configuration, which is the lite-base-P2 with a fine-tuning configuration of b4, e4, lr2e-5, and ws0.3, is quite

significant. This result also proves that characteristics differences between two data sets affect the performance of the ASAG system. The analyzed data set characteristics are the number of foreign words, number of topics, question, and answer composition, the maximum length of a sentence, and the scoring mechanism.

Another factor as evidenced by these results is that the grading and labeling in each data set can significantly affect the training of the BERT model. The human correlation score shows the agreement and consistency level of each annotator. However, a high correlation score does not always result in an accurate grading. It can lead to confusion on BERT model training. Such grading inaccuracy is caused by how human makes decisions. The scoring rubric counters such humane factors, increasing objectivity while grading. It leads to a superior consistency and correlation score, as seen in Table 14.

Results obtained from the Indonesian ASAG system in Table 15 prove that fine-tuning approach yields better results than the feature-based approaches [13], [28]. The BERT and ALBERT model outperforms machine learning models, such as the ridge regression model implemented by Sultan *et al.* [6] and SVMRank implemented by Mohler *et al.* [29]. It shows how the transfer learning of BERT and ALBERT models improves their performance during the fine-tuning phase. It also shows that the increasing number of features in the machine learning model cannot enable the model to outperform neural network architectures such as BERT and ALBERT models.

Last, we did a subjective evaluation by interviewing five lecturers with four to eight years of experience lecturing 50 to 200 students at the time of this research. We find that the number of answers and the lack of time are the main difficulties of grading as a lecturer. The benefits of using the Indonesian ASAG system are: i) speeding up the annotator's grading task, ii) maintaining the objectivity of the grading task, and iii) enabling the annotator to reuse the system to grade used questions. All agreed that this Indonesian ASAG system performed well on grading short answers with an average score of eight out of ten and can be helpful as an alternative solution to their problems. They also showed interest in using this system in the future.

4. CONCLUSION

This study proves that the best BERT-based model can outperform the best ridge regression model in the Indonesian ASAG task. The best model of ALBERT (lite-base-P2) achieves 0.9508 in pearson's correlation and 0.4138 in RMSE, whereas the ridge regression model achieves only 0.538 in RMSE. This BERT-based model also can surpass the performance of other models in previous studies. Fine-tuning is one of the main factors that affect the performance of this model. This study finds that the best fine-tuning configuration for this model is a batch size of 4, epoch size of 4, a learning rate of $2e-5$, and 0.3 warm-up steps. It also confirms the possibility of improving the current accuracy of AI systems in grading short answer questions so the education world can thrive further, especially in Indonesia.

ACKNOWLEDGEMENTS

The authors would like to express our gratitude to Bina Nusantara University for providing educational and other relevant resources for such research to be conducted.

APPENDIX

Table 7. Ridge regression model experiment scenario with the basic programming data sets

Table 7: Ridge Regression model experiment scenario with the basic programming data sets													
No.	Minimum cosine similarity score							ppdbSim	Term weighting		Text preprocessing		
	0.625	0.65	0.675	0.7	0.75	0.8	0.9	Cosine similarity	0.9	TF-IGM	TF-IDF	Stemming	Lemmatization
1	✓								✓	✓		✓	
2		✓							✓	✓		✓	
3			✓						✓	✓		✓	
4				✓					✓	✓		✓	
5					✓				✓	✓		✓	
6						✓			✓	✓		✓	
7							✓		✓	✓		✓	
8								✓		✓		✓	
9	✓								✓		✓	✓	
10		✓							✓		✓	✓	
11			✓						✓		✓	✓	
12				✓					✓		✓	✓	

Table 7. Ridge regression model experiment scenario with the basic programming data sets (continue)

No.	Minimum cosine similarity score							ppdbSim Cosine similarity	0.9	Term weighting		Text preprocessing	
	0.625	0.65	0.675	0.7	0.75	0.8	0.9			TF- IGM	TF- IDF	Stemming	Lemmatization
13					✓				✓		✓	✓	
14						✓			✓		✓	✓	
15							✓		✓		✓	✓	
16								✓			✓	✓	
17	✓								✓	✓			✓
18		✓							✓	✓			✓
19			✓						✓	✓			✓
20				✓					✓	✓			✓
21					✓				✓	✓			✓
22						✓			✓	✓			✓
23							✓		✓	✓			✓
24								✓		✓			✓
25	✓								✓		✓		✓
26		✓							✓		✓		✓
27			✓						✓		✓		✓
28				✓					✓		✓		✓
29					✓				✓		✓		✓
30						✓			✓		✓		✓
31							✓		✓		✓		✓
32								✓			✓		✓

Table 8. Ridge regression model experiment scenario with the Rahutomo data sets

No.	Minimum cosine similarity score							ppdbSim Cosine similarity	0.9	Term weighting		Text preprocessing	
	0.625	0.65	0.675	0.7	0.75	0.8	0.9			TF- IGM	TF- IDF	Stemming	Lemmatization
1		✓							✓	✓		✓	
2			✓						✓	✓		✓	
3				✓					✓	✓		✓	
4								✓		✓		✓	
5		✓							✓		✓	✓	
6			✓						✓		✓	✓	
7				✓					✓		✓	✓	
8								✓			✓	✓	
9		✓							✓	✓			✓
10			✓						✓	✓			✓
11				✓					✓	✓			✓
12								✓		✓			✓
13		✓							✓		✓		✓
14			✓						✓		✓		✓
15				✓					✓		✓		✓
16								✓			✓		✓




REFERENCES

- [1] B. Winarji, "Education and technical training for teaching and learning activities for tutors" (in Indonesia: Pendidikan dan pelatihan teknis kegiatan belajar mengajar bagi pamong belajar)," *Kementerian Pendidikan dan Kebudayaan 1 ed.*, Depok, Jawa, 2016, pp. 18–36.
- [2] U. Hasanah, T. Astuti, R. Wahyudi, Z. Rifai, and R. A. Pambudi, "An Experimental Study of Text Preprocessing Techniques for Automatic Short Answer Grading in Indonesian," *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, 2018, pp. 230–234, doi: 10.1109/ICITISEE.2018.8720957.
- [3] S. Burrows and D. D'Souza, "Management of teaching in a complex setting," *Conference: Proceedings of the Second Melbourne Computing Education Conventicle*, Melbourne, December 2005.
- [4] S. Burrows, I. Gurevych, and B. Stein, "The eras and trends of automatic short answer grading," *International Journal of Artificial Intelligence in Education*, vol. 25, no. 1, pp. 60–117, March 2015, doi: 10.1007/s40593-014-0026-8.
- [5] G. B. Herwanto, Y. Sari, B. N. Prastowo, M. Riasetiawan, I. A. Bustoni, and I. Hidayatullo, "UKARA: A fast and simple automatic short answer scoring system for Indonesian," in *Proceeding Book of 1st International Conference on Educational Assessment and Policy*, vol. 2, pp. 48–53, 2018.
- [6] 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*, June 2016, pp. 1070–1075.
- [7] S. K. Gaddipati, D. Nair, and P. G. Plöger, "Comparative evaluation of pretrained transfer learning models on automatic short answer grading," *arXiv:2009.01303v1*, pp. 1–8, September 2020, doi: 10.48550/arXiv.2009.01303.




- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018, doi: 10.48550/arXiv.1810.04805.
- [9] F. Rahutomo, *et al.*, "Open problems in Indonesian automatic essay scoring system," *Int. J. Eng. Technol.*, vol. 7, no. 4.44, pp. 156–160, 2018.
- [10] Kementerian Pendidikan dan Kebudayaan, "Primary school statistics 2019/2020" (in Indonesia: *Statistik persekolahan SD 2019/2020*), 1st ed., Tangerang Selatan, Banten: Pusdatin Kemendikbud, 2020, p. 1.
- [11] Kementerian Pendidikan dan Kebudayaan, "Middle School Statistics 2019/2020" (in Indonesia: *Statistik persekolahan SMP 2019/2020*), 1st ed., Tangerang Selatan, Banten: Pusdatin Kemendikbud, 2020, p. 1.
- [12] Kementerian Pendidikan dan Kebudayaan, "High school statistics 2019/2020" (in Indonesia: *Statistik persekolahan SMA 2019/2020*), 1st ed., Tangerang Selatan, Banten: Pusdatin Kemendikbud, 2020, p. 1.
- [13] Kementerian Pendidikan dan Kebudayaan, "Vocational School Statistics 2019/2020" (in Indonesia: *Statistik persekolahan SMK 2019/2020*), 1st ed., Tangerang Selatan, Banten: Pusdatin Kemendikbud, 2020, p. 1.
- [14] F. Herdiyanto, Kementerian Riset, Teknologi, dan Pendidikan Tinggi, "Higher Education Statistics" (in Indonesia: *Statistik Pendidikan Tinggi*), 4th ed., Ed., Jakarta Pusat, DKI Jakarta: Pusat Data dan Informasi IPTEK DIKTI, 2018, pp. 1-256.
- [15] J. Wei and K. Zou, "Eda: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks," *arXiv preprint arXiv:1901.11196*, 2019, doi: 10.48550/arXiv.1901.11196.
- [16] Y. W. Syafiudin, I. F. Rozi, M. Mentari, and Y. A. Lestari, "Basics of Programming: Basics of Programming" (in Indonesia: *Dasar Pemrograman: Dasar Pemrograman*), vol. 1, pp. 27–31, 2018.
- [17] T. H. F. Harumy, A. P. Windarto, and I. Sulistianingsih, "Learn basic algorithms and C++ programming," (in Indonesia: *Belajar dasar algoritma dan pemrograman C++*), Yogyakarta: Deepublish, 2016, pp. 32–137.
- [18] Liswati and M. Sahal, "Basic programming for SMK/MAK class X" (in Indonesia: *Pemrograman dasar untuk SMK/MAK kelas X*), T. Grasindo, Ed., Jakarta, DKI Jakarta: PT Gramedia Widiasarana Indonesia, 2018, pp. 3–18.
- [19] B. Wilie, *et al.*, "IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding," *arXiv preprint arXiv:2009.05387*, 2020, doi: 10.48550/arXiv.2009.05387.
- [20] C. Wirawan, "BERT-base-indonesian-522M," 23 Juni 2020. [Online]: Available: <https://huggingface.co/cahya/bert-base-indonesian-522M>. [Accessed 25 Februari 2021].
- [21] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP," *arXiv preprint arXiv:2011.00677*, 2020, doi: 10.48550/arXiv.2011.00677.
- [22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2020, doi: 10.48550/arXiv.1909.11942.
- [23] K. Kurniawan and A. F. Aji, "Toward a Standardized and More Accurate Indonesian Part-of-Speech Tagging," *2018 International Conference on Asian Language Processing (IALP)*, 2018, pp. 303–307, doi: 10.1109/IALP.2018.8629236.
- [24] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020, doi: 10.48550/arXiv.2003.07082.
- [25] I. P. A. Karasugi, "agikarasugi/lemmatizerIndo-python," 24 Juni 2019. [Online]. Available: <https://github.com/agikarasugi/lemmatizerIndo-python>. [Accessed 18 Maret 2021].
- [26] M. A. Sultan, S. Bethard, and T. Sumner, "Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 219–230, 2014, doi: 10.1162/tacl_a_00178.
- [27] K. Chen, Z. Zhang, J. Long, and H. Zhang, "Turning from TF-IGM to TF-IGM for term weighting in text classification," *Expert Systems with Applications*, vol. 66, pp. 245–260, December 2016, doi: 10.1016/j.eswa.2016.09.009.
- [28] M. H. Haidir and A. Purwarianti, "Short answer grading using contextual word embedding and linear regression," *Jurnal Linguistik Komputasional*, vol. 3, no. 2, pp. 54–61, September 2020, doi: 10.26418/jlk.v3i2.38.
- [29] M. Mohler, R. Bunesco, and R. Mihalcea, "Learning to grade short answer questions using semantic similarity measures and dependency graph alignments," *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, June 2011, pp. 752–762.

BIOGRAPHIES OF AUTHORS







Heinrich Reagan Salim    was born in Bandar Lampung, Lampung, Indonesia in 1999. He received the S. Kom degree in computer science from Bina Nusantara University, West Jakarta, DKI Jakarta, Indonesia in 2021. In 2019, he started pursuing his career as an IT Trainee at PT Bank Central Asia Tbk, DKI Jakarta, Indonesia. From April 2020 until now, he has been working as a full-time IT specialist in the webserver administration. His research interests include natural language processing, language modeling (BERT and ELMo), depression detection systems, and automatic short answer grading. He can be contacted at email: heinrich.salim@binus.ac.id.







Chintya De    was born in West Jakarta, DKI Jakarta, Indonesia, in 1999. She received the S.Kom degree in computer science from Bina Nusantara University, West Jakarta, DKI Jakarta, Indonesia, in 2021. From May to August 2017, she was a Song Integration Collaborator Intern at PT Chorus Digital Indonesia. Since 2019, she has been an IT Trainee at PT Bank Central Asia Tbk, DKI Jakarta, Indonesia. She has been working there as a full-time IT Specialist in the mainframe application team from April 2020 until now. Her research interests include natural language processing, automatic short answer grading, and BERT. She can be contacted at email: chintya.de@binus.ac.id.



Nicholas Daniel Pratamaputra     was born in Jakarta, Indonesia in 2000. He received the S. Kom degree in computer science from Bina Nusantara University, West Jakarta, DKI Jakarta, Indonesia in 2021. In 2019, he started pursuing his career as an IT Trainee at PT Bank Central Asia Tbk, DKI Jakarta, Indonesia. Since April 2020, he has been working as a full-time IT Specialist in the database administration. His research interests include natural language processing, automatic short answer grading, and machine learning. He can be contacted at email: nicholas.pratamaputra@binus.ac.id.



Derwin Suhartono     is faculty member and the head of the Computer Science Department in Bina Nusantara University, Indonesia. He got his Ph.D. degree in computer science from Universitas Indonesia in 2018. His research fields are natural language processing. Recently, he has been continuously researching argumentation mining and personality recognition. He actively involves in the Indonesia Association of Computational Linguistics (INACL), a national scientific association in Indonesia. He has professional memberships in ACM, INSTICC, and IACT. He also takes the role of reviewer in several international conferences and journals. He can be contacted at email: dsuhartono@binus.edu.