Scientific Journal of Informatics
Vol. 11, No. 3, Aug 2024
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

Combining Multiple Text Representations for Improved Automatic Evaluation of Indonesian Essay Answers

Moh Edi Wibowo[1*], Nur Rokhman[2], Agus Sihabuddin[3]
[1, 2, 3]Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Indonesia

Abstract.
Purpose: Essay questions serve as important examination methods to provide a more elaborative insight, compared to multiple-choices, regarding students' learning achievement. When the number of students in a class is huge, however, examinations using essay questions become hard to conduct and take a long evaluation time. Automatic essay evaluation has, therefore, become a potential approach in this situation. Various methods have been proposed, however, optimal solutions for such evaluation in the Indonesian language are less known. Furthermore, with the rapid development of machine learning approaches, in particular deep learning approaches, the investigation of such optimal solutions becomes more necessary.
Method: To address the aforementioned issue, this study proposed the investigation of text representation approaches for optimal automatic evaluation of Indonesian essay answers. The investigation compared pre-trained word embedding methods such as Word2vec, GloVe, FastText, and RoBERTa, as well as compared text encoding methods such as long short-term memories (LSTMs) and transformers. LSTMs are able to capture temporal semantics by employing state variables, while transformers are able to capture long-term dependency between parts of their input sequences. Additionally, we investigated classification-based and similarity-based training to build text encoders. We expected that these training approaches allowed encoders to extract different views of information. We compared classification results produced by different text encoders and combinations of text encoders.
Result: We evaluated various text representation approaches using the UKARA dataset. Our experiments showed that the FastText word embedding method outperformed the Word2vec, GloVe, and RoBERTa methods. The FastText method achieved an F1-score of 75.43% on validation sets, while the Word2vec, GloVe, and RoBERTa methods achieved F1-scores of 69.56%, 74.53%, and 72.87%, respectively. In addition, the experiments showed that combinations of text encoders outperformed individual encoders. The combination of the LSTM encoder, the transformer encoder, and the TF-IDF encoder obtained an F1-score of 77.22% in the best case, which is better than the best F1-scores of the individual LSTM encoders (75.35%), the best combination of transformer encoders (71.49%), and the individual TF-IDF encoder (76.69%). We observed that LSTM encoders produced better performance when they were built using classification-based training. Meanwhile, the transformer encoders obtained better performance when built using similarity-based training.
Novelty: The novelty proposed in this research is the optimal combination of text encoders specifically constructed for the evaluation of essay answers in the Indonesian language. Our experiments showed that the combination of three encoders - namely the LSTM encoder built using classification-based training, the transformer encoder built using classification-based and similarity-based training, and the TF-IDF encoder - obtained the best classification performance.

Keywords: Automatic essay evaluation, LSTM, Transformer, Text similarity, Siamese neural network
Received July 2024 / **Revised** August 2024 / **Accepted** August 2024

INTRODUCTION

Essay questions are important methods to examine student understanding of course materials [1]. Compared to multiple-choices, essay questions allow a more elaborative examination and provide a deeper insight about students' learning achievement [2]. Essay questions require students to apply logical reasoning and higher-order thinking skills to formulate answers [3]. Moreover, this approach allows students to provide opinions with respective justifications and pushes them to practice systematic writing, which is an essential life skill.

*Corresponding author.
Email addresses: mediw@ugm.ac.id (Wibowo)*, nurrokhman@ugm.ac.id (Rokhman), a_sihabudin@ugm.ac.id (Sihabudin)
DOI: 10.15294/sji.v11i3.9440

With the increasing popularity of online courses, including the massive open online courses (MOOCs), a large number of students (hundreds or more) can be enrolled in a single online class. When this happens, examinations using essay questions become hard to conduct and take a long evaluation time. While course providers may spend more funding to hire graders or examiners, the additional cost may not be feasible. Meanwhile, removing essay questions from student assessments may degrade courses' quality. Automatic essay scoring has, therefore, become a potential solution to overcome this issue [4]. An automatic system is able to quickly analyze students' answers and to decide whether the answers are correct, partially correct, or incorrect [4].

The history of automatic essay scoring has been reviewed by Burrows et al. [5], who divided the timeline into five eras: concept mapping, information extraction, corpus-based method, machine learning, and evaluation. The review emphasized the recent development of the area that focused on reproducibility, standardized corpora, and permanent evaluations e.g., those supported by evaluation forums such as SemEval Semantic Textual Similarity or STS [6]–[8].

The most direct method of automatic essay evaluation is applying grammar and semantic analysis to extract the core content of essay answers [9], [10]. The core content can then be matched to ground-truth answers to obtain the degree of correctness. This method, which directly corresponds to the human way, though, is not adaptive. It requires an explicit encoding of human knowledge and often involves extensive manual tuning. Reapplying the method from one domain to another domain is not practical. A more practical approach for the automatic evaluation is the machine learning approach. Assuming that there are a large number of answers as examples, enclosed with the respective labels, a machine learning algorithm can learn and extract patterns from the examples based on which future answers can be evaluated and marked.

The machine learning approaches can be divided into two categories: response-based and reference-based approaches [11]. The response-based approach employs the availability of a large number of pre-scored student responses to train reliable regression or classification models. Static-length features such as bag-of-words or n-grams and vector-space classifiers such as support vector machines (SVMs) have been used to build scoring systems [12]–[14]. Zedan and Al-Sultan [15], for example, combined features such as term weights, length ratios, and lexical or semantic similarities with supervised learning algorithms to produce essay scores. They also proposed the application of text augmentation, based on key-grading specific constructs, i.e., question demoting, to obtain more accurate text classifiers. Evaluations on SemEval-2013 and Mohler datasets [6], [12] have confirmed the effectiveness of the augmentation.

Reference-based approaches address the issue of scarce responses by formulating automatic scoring tasks as text similarity problems. In this case, the score of a student's response is computed by comparing the response to some reference answers. Higher syntactic or semantic similarity means a higher mark/grade for the answer. Current work on reference-based approaches is focused on vector-space regression. Pado [14] performed automatic essay evaluation by employing features such as n-grams, text similarities, dependencies, abstract semantic representations, and entailment votes. Features obtained from students' responses were matched against those obtained from reference answers. This work produced two types of corpora: language skills and content assessments, showing that the features may have different levels of effectiveness depending on corpus type. Ratna et. al [16] introduced a web-based essay grading system called SIMPLE-O that utilized latent semantic analysis (LSA) to estimate similarities between student responses and reference answers. Their experiments, which involved 40 students and 3 lecturers as graders, demonstrated that the system obtained 86% agreement with human ratings. One problem that appeared from the aforementioned methods is that they cannot directly handle inputs with variable lengths. Features employed by the methods are also "heuristically crafted", leading to the sub-optimality of their performance.

While various methods have been proposed, optimal solutions for automatic essay evaluation in the Indonesian language are less known. The Indonesian language is a standardized form of Malay that has been used as a lingua franca in the Nusantara archipelago for centuries [17]. The language is now spoken by almost 300 million people [18] and has joined the ranks of UNESCO's ten official languages alongside English, Arabic, Mandarin, Russian, French, Spanish, Hindi, Italian, and Portuguese [19]. Optimal text representations for Indonesian essay answers are, therefore, very important. With the rapid development of

machine learning approaches, particularly deep learning approaches, investigating such optimal solutions becomes more necessary.

To seek the aforementioned optimal solutions, this study compared pre-trained word embedding methods such as Word2vec, GloVe, FastText, and RoBERTa and compared text encoding methods such as long short-term memories (LSTMs) [20] and transformers [21]. LSTMs have been known to be able to capture temporal semantics by employing state variables, while transformers are able to capture long-term dependency between parts of their input sequences. We also investigated combinations of text encoding methods to determine their effectiveness in improving the accuracy of essay evaluation.

METHODS

Figure 1 shows the proposed essay evaluation method. The method takes a student's answer (text) at a time and processes this input by applying word embedding, text encoding, and classification/matching to obtain an evaluation result.

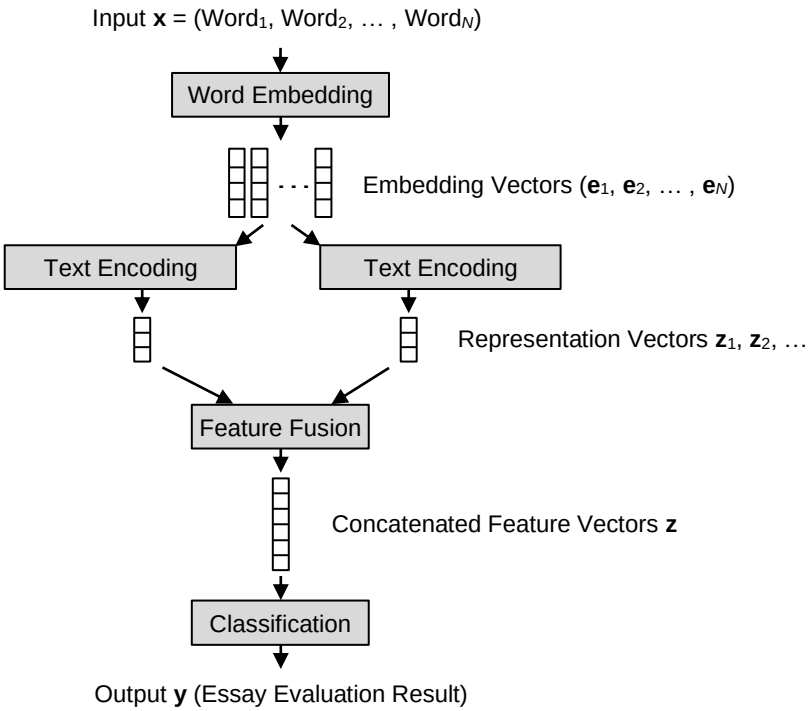


Figure 1. The proposed essay evaluation method

Pre-processing

The input of the proposed method is a text provided by a student as an answer to a particular question. To extract features from this text, some pre-processing steps are applied. The first step is replacing non-relevant characters in the text with spaces. The only characters considered relevant are a-z, A-Z, 0-9, ".", ",", "?", and "!". Following this, the text is split into words using whitespaces as delimiters. At this point, the text has become a sequence of words.

Word embedding

Before a sequence of words can be further processed, each word in the sequence has to be converted into a vector. We evaluated some word embedding methods for the conversion, namely Word2vec [22], GloVe [23], FastText [24], and RoBERTa [25], [26]. Word2vec [22] finds the vector representation of a word as the most predictive vector with regard to the surrounding words. The resulting vectors obtained from this approach are able to capture semantic meanings, allowing words with similar meanings to have similar vector representations. GloVe [23] finds representations of words by employing information from the words' neighborhoods and explicitly using the number of word co-occurrences in a particular corpus. The method thus aims to obtain representations that are representative from the global perspective. FastText

[24] works on character n-grams to obtain more fine-grained representations of texts before the method produces vector representations of words. This approach captures sub-word information, making it effective for languages with rich morphology and handling out-of-vocabulary words. RoBERTa [25] is an improved version of BERT [26], which also works on sub-word tokens. This method employs the powerful representational capability of the transformer [21] while at the same time considering the use of both left and right context (thus bidirectional). RoBERTa is trained using dynamic masking on longer sequences to improve its generalization.

We employed pre-trained models of the Word2vec[2], GloVe[3], FastText[4], and RoBERTa[5] trained using corpora in the Indonesian language. These pre-trained models were built from large and representative datasets such that they are able to produce more general and better representations. We used the models to embed words into vectors of certain dimensions. Specifically, the pre-trained Word2vec, GloVe, FastText, and RoBERTa models produce vectors of 300, 50, 300, and 768 dimensions, respectively. After applying word embedding, the sequence of words is converted into a sequence of vectors.

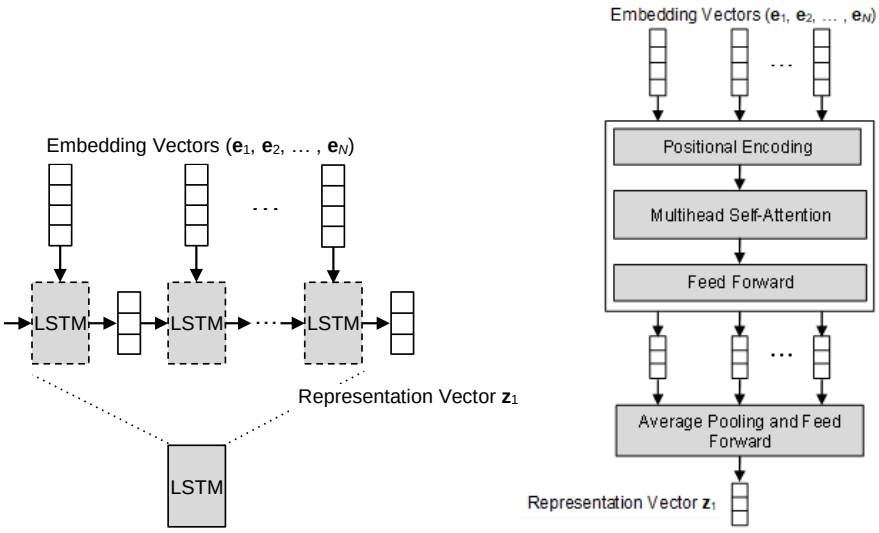


Figure 2. Text encoding using (a) the LSTM and (b) the transformer

To build both encoders, we plugged a feed-forward layer with a sigmoid activation function at the end of the encoders (Figure 3). Correct answers are given the label 1, while incorrect answers are given the label 0. We then conducted classification-based training using the mean squared error (MSE) as the loss function. Note that the MSE was computed from the values obtained from the sigmoid outputs and the targets specified in the training data. After the training finished, we unplugged the feed-forward layer and used the output of the original model as vector representations of texts. Note that we padded input sequences with zero vectors if necessary so that they had the same lengths (equals the length of the longest sequence in the dataset).

[2] https://yudiwbs.wordpress.com/2016/11/17/word2vec-untuk-bahasa-indonesia/

[3] https://yudiwbs.wordpress.com/2018/04/02/glove-untuk-wikipedia-bahasa-indonesia/

[4] https://fasttext.cc/docs/en/crawl-vectors.html

[5] https://huggingface.co/cahya/roberta-base-indonesian-522M

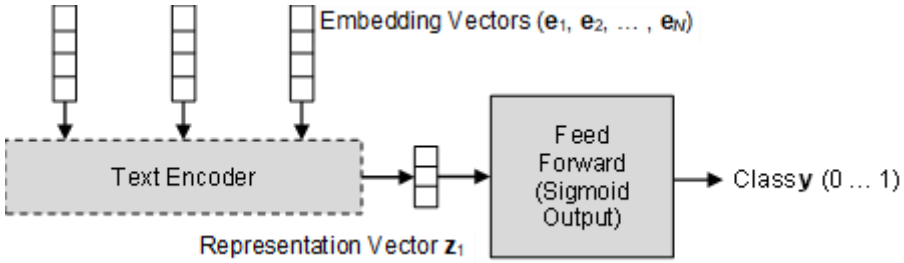


Figure 3. Classification-based training to build text encoders

Text encoding using similarity-based method

To complete our investigation, we also applied similarity-based methods to build text encoders. For this purpose, we trained the encoders using the Siamese network architecture [27], [28], as shown in Figure 4. This architecture takes two sequences as inputs, both of which are given to the same text encoder. The two outputs produced by the encoder are compared using the cosine similarity formula. The Siamese networks were built using training sets containing matched pairs and unmatched pairs. Matched pairs were given the label +1 which represents high similarities, while unmatched pairs were given the label -1 which represents low similarities. Using these labels, the training was conducted by employing the mean squared error (MSE) as the loss function.

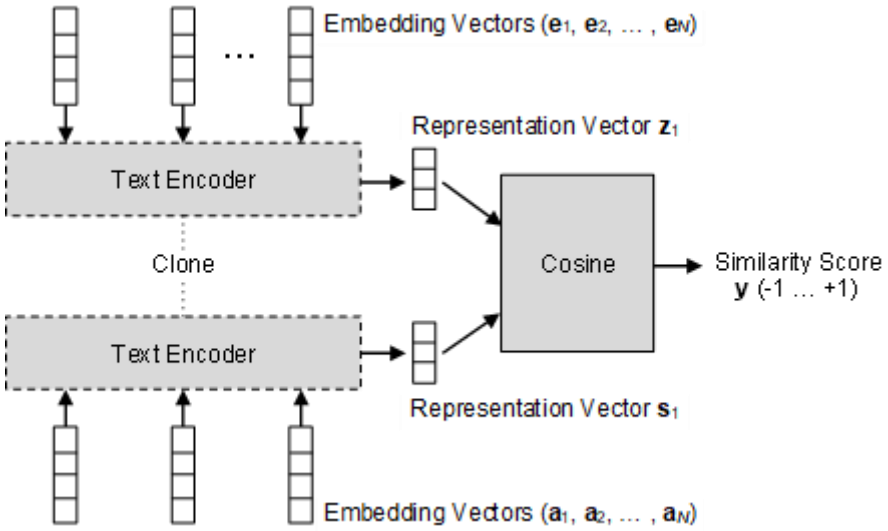


Figure 4. Similarity-based training to build text encoders

Since the original dataset used does not provide pairs for training (only provides answers with correct or incorrect labels), we needed to generate the pairs from the dataset. For a particular question, we proposed to generate the pairs as follows.

1. Initiate the set of unmatched pairs $\mathbf{P}^-$ and the set of matched pairs $\mathbf{P}^+$ to empty sets;
2. Choose randomly an answer from the training set of incorrect answers $\mathbf{X}^- = \{\mathbf{x}^-_1, \mathbf{x}^-_2, \mathbf{x}^-_3, \dots, \mathbf{x}^-_M\}$ and an answer from the training set of correct answers $\mathbf{X}^+ = \{\mathbf{x}^+_1, \mathbf{x}^+_2, \mathbf{x}^+_3, \dots, \mathbf{x}^+_N\}$ and add the pair to the set of unmatched pairs $\mathbf{P}^-$, repeat as required;
3. Extract features from each answer in the set of correct answers $\mathbf{X}^+ = \{\mathbf{x}^+_1, \mathbf{x}^+_2, \mathbf{x}^+_3, \dots, \mathbf{x}^+_N\}$ to produce a set of representation vectors $\mathbf{U}^+ = \{\mathbf{u}^+_1, \mathbf{u}^+_2, \mathbf{u}^+_3, \dots, \mathbf{u}^+_N\}$;
4. Apply the agglomerative clustering to $\mathbf{U}^+$ using cosine distance as the metric to produce clusters $\mathbf{U}^+_1$, $\mathbf{U}^+_2$, $\mathbf{U}^+_3$, … , $\mathbf{U}^+_C$ of correct answers;
5. For each cluster $\mathbf{U}^+_i$, randomly choose two answers from the cluster and add the pair to the set of matched pairs $\mathbf{P}^+$, repeat as required.

The procedure chooses any pairs of correct and incorrect answers as unmatched pairs. For the matched pairs, the procedures only choose pairs of correct answers from the same cluster. This is to make the

matched pairs close enough to each other in the feature space. Note that we called correct answers in the training set as reference answers.

During the inference, we used the trained text encoders to encode input answers. The obtained representation vectors were then "compared" to the average representation vector of reference answers to produce the final representations (Figure 5). Note that we used the Hadamard product for the "comparison" since the Hadamard product is the "ingredients" for cosine similarity.

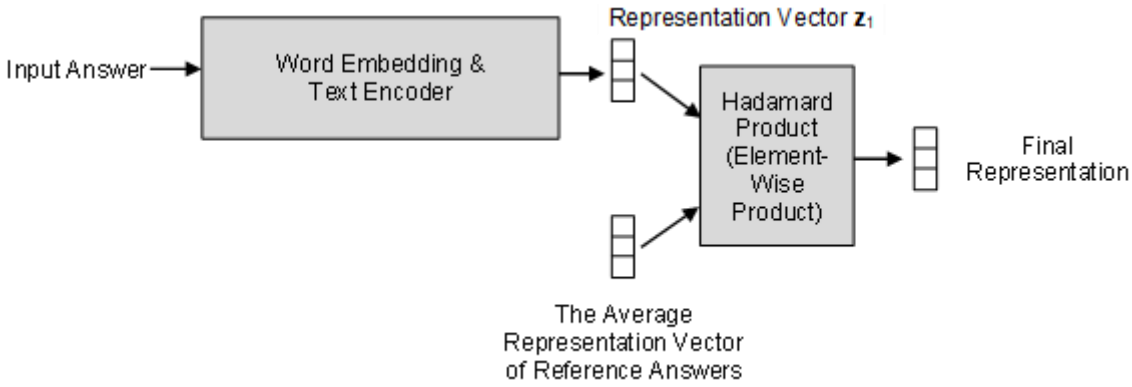


Figure 5. Extraction of final representations in similarity-based text encoders

Feature fusion and classification

As explained in the previous subsections, we have two alternatives to perform text encoding, namely the LSTM encoders and the transformer encoders. In addition to using the encoders individually, we proposed to combine multiple representation vectors obtained from multiple encoders (by concatenation). This scheme is known as feature fusion. We considered that various representations may bring complementary information and can be combined to produce improved performance. The last step of the proposed method is the classification. We used support vector machines (SVMs) [29] to classify the given essay answers into correct or incorrect.

RESULTS AND DISCUSSIONS

Dataset and evaluation setup

To evaluate the validity and effectiveness of the proposed method, experiments were carried out over the UKARA dataset[6]. The UKARA dataset contains 10 essay questions, each with approximately 5000 - 6000 answers. Answers for each question are further classified into two sets, namely the sets of correct answers and incorrect answers. This grouping is performed by a lot of human graders who have received technical guidance on how to perform the grading. Table 1 shows in detail the number of answers and the grouping of the answers for the 10 questions. We regarded the set of correct answers as the positive one.

Table 1. Statistics of the UKARA dataset

Question #	# Incorrect Answers	# Correct Answers	# Total Answers
1	5045	757	5802
2	4605	1235	5840
3	5566	140	5706
4	2709	7	2716
5	5288	393	5681
6	2065	3731	5796
7	3899	1972	5871
8	3524	2298	5822
9	4714	1072	5786
10	2988	2774	5762

We used 3-fold cross-validation to conduct experiments on the UKARA dataset. For each question, stratified random sampling was employed to create training, validation, and test sets using a proportion of

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

approximately 60%, 10%, and 30%, respectively. The validation sets were used to fine-tune the proposed methods before they were applied to the test sets. By using three folds, we attempted to achieve the balance between real-world scenarios and experiment validity. When we used more folds (e.g. five or 10 folds), the number of training data, in the context of essay evaluation, became much lower than the number of test data. This did not conform to real situations of essay evaluation where the number of labeled data (i.e., the training data) is normally much less than the number of unlabeled data (test data). We also did not choose one or two folds since these numbers of folds were too low for cross-validation (and are, therefore, uncommon). When one or two folds were used, we were afraid that the obtained results were not valid and convincing enough to support research conclusions.

Model tuning and classification-based training

We built text encoders by connecting feed-forward neural networks with sigmoid outputs to the encoders and conducted classification-based training. Figure 6 shows the MSE obtained from training the LSTM model of question number 1 with 40 neurons. The graph reveals that the MSE steadily decreased as the training progressed. At the beginning of the training, the MSE was close to 0.09 (both the MSE obtained from the training data and from the validation data). After 40 epochs, the MSE of the training data dropped to around 0.04, while the MSE of the validation data dropped to around 0.065. The MSE of the validation data was, therefore, always higher than the MSE of the training data (to be expected). The training, however, did not suffer from overfitting since the MSE of the validation data did not increase until the training finished.

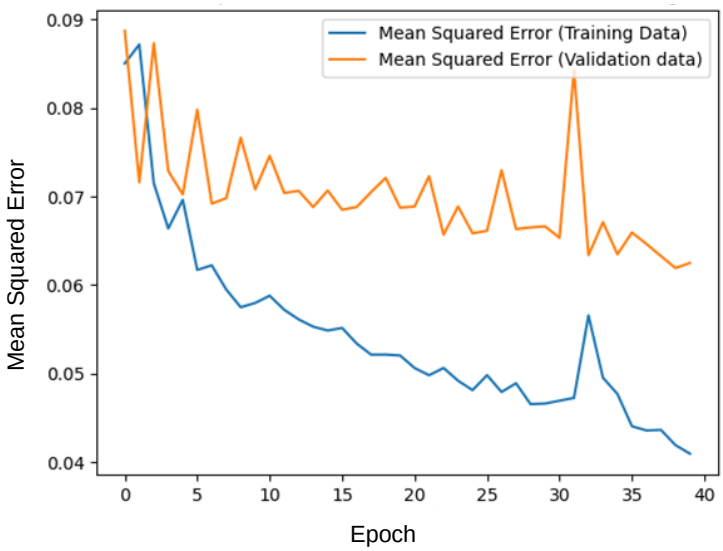


Figure 6. Mean squared error (MSE) observed during classification-based training

Table 2 shows classification results when LSTM encoders, trained using test sets, were employed to classify answers in validation sets. Bold numbers represent the best performance of a particular metric, while underlined numbers represent the best performance of a particular metric for a certain word embedding. We compared the four word embedding methods and evaluated various numbers of neurons for text encoding. Since the number of correct answers is generally less than that of incorrect answers, we valued precision, recall, and F1-score more than accuracy.

Table 2. Validation results obtained from the LSTM encoders

Word Embedding	# Neurons	Validation Result (%)			
		Accuracy	Precision	Recall	F1-Score
Word2vec	5	83.57	67.79	71.12	69.42
	10	83.38	67.16	71.68	69.35
	20	83.48	67.91	70.15	69.01
	40	83.93	69.12	70.01	69.56
GloVe	5	84.31	69.13	72.60	70.82
	10	84.52	67.16	80.16	73.09
	20	85.37	69.77	77.99	73.65

	40	85.31	68.68	80.83	74.26
FastText	5	85.29	70.31	75.98	73.03
	10	85.87	70.40	79.56	74.71
	20	**86.00**	69.92	81.78	75.39
	40	85.99	69.83	**82.00**	**75.43**
RoBERTa	5	85.62	**75.28**	67.22	71.02
	10	85.62	71.91	74.14	73.00
	20	85.35	72.48	71.12	71.79
	40	85.26	71.33	73.24	72.27

Table 2 shows that the best results were achieved by the FastText method and LSTM encoders of 20 or 40 neurons, respectively. The FastText method produced an F1-score of 75.43% in the best case, which was better than the best results produced by the Word2vec, GloVe, and RoBERTa methods. The Word2vec method produced the worst performance i.e., an F1-score of 69.56% in the best case. The GloVe and the RoBERTa methods were the second and the third performers, respectively. The GloVe method produced an F1-score of 74.26% in the best case, while the RoBERTa method produced an F1-score of 73.00% in the best case.

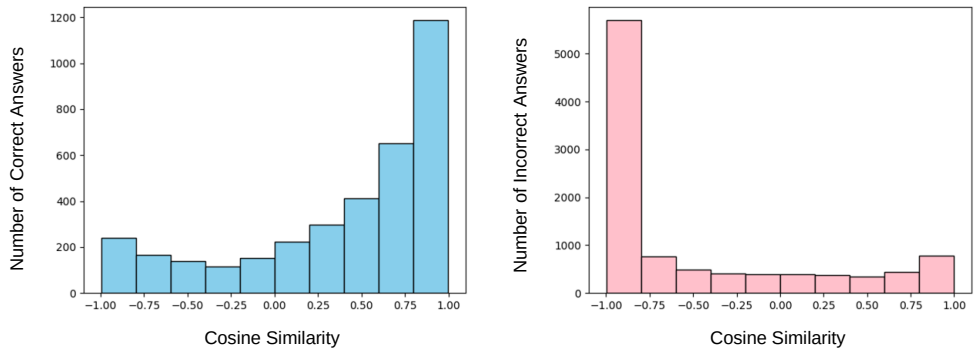
Table 3 shows classification results when transformer encoders were employed for text encoding. Similar to the previous results, Table 3 shows that the best results were achieved by the FastText method and text encoders of 20 or 40 neurons, respectively. The FastText method produced an F1-score of 75.14% in the best case. The Word2vec method still produced the worst performance i.e., an F1-score of 69.01% in the best case. The GloVe and the RoBERTa methods produced F1-scores of 74.53% and 72.87% in the best cases, respectively. We used these results to direct further experiments. In the subsequent experiments, we employed the FastText method for word embedding and text encoders of 40 neurons for text encoding.

Table 3. Validation results obtained from the transformer encoders

Word Embedding	# Neurons	Validation Result (%)			
		Accuracy	Precision	Recall	F1-Score
Word2vec	5	81.26	70.34	49.36	58.01
	10	82.65	74.24	51.79	61.01
	20	83.76	69.26	68.47	68.86
	40	84.16	70.85	67.27	69.01
GloVe	5	85.76	73.14	72.24	72.68
	10	86.15	**74.43**	71.88	73.13
	20	86.45	73.48	75.61	74.53
	40	85.97	72.82	74.16	73.49
FastText	5	79.94	67.14	46.01	54.60
	10	81.59	71.14	50.14	58.82
	20	86.20	72.05	**77.40**	74.63
	40	**86.71**	73.76	76.56	**75.14**
RoBERTa	5	78.34	67.81	33.15	44.53
	10	80.76	68.05	50.17	57.76
	20	84.30	68.48	74.33	71.29
	40	85.55	71.77	74.00	72.87

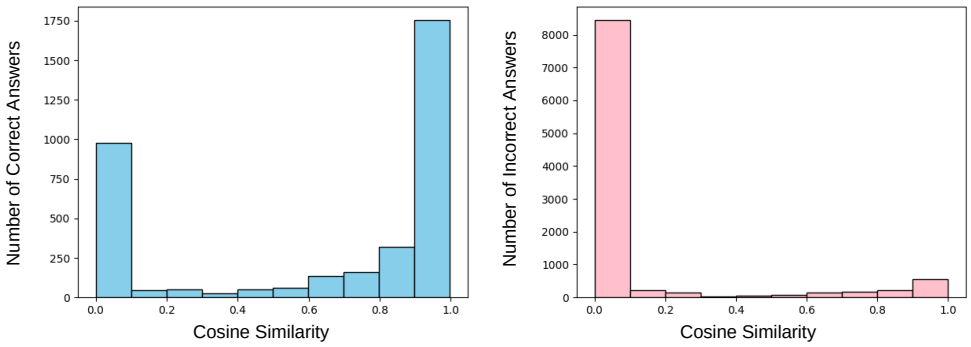
Similarity-based training

We built text encoders using similarity-based training by employing the Siamese network architecture. The encoders were trained to produce low similarities when given pairs of correct answers and reference answers and high similarities when given pairs of incorrect answers and reference answers. Figure 7 shows distributions of cosine similarities of correct answers (a) and incorrect answers (b) when the answers were encoded using LSTM encoders and were compared to the average representation vectors of reference answers. Figure 8 shows similar distributions obtained from transformer encoders. Both figures show that most of the correct answers had high cosine similarities, while most of the incorrect answers had low cosine similarities. These results thus suggested that the similarity-based training has worked as expected.



(a) (b)

Figure 7. Distributions of cosine similarities of (a) correct answers and (b) incorrect answers, with respect to the average representation vectors of reference answers, obtained from LSTM encoders



(a) (b)

Figure 8. Distributions of cosine similarities of (a) correct answers and (b) incorrect answers, with respect to the average representation vectors of reference answers, obtained from transformer encoders

Method comparison

As described previously, we have alternatives for text encoders, namely LSTM encoders and transformer encoders; both can be trained using a classification-based approach (denoted as CB) or a similarity-based approach (denoted as SB). We have, therefore four alternatives of text-encoders: LSTM constructed using classification-based training (LSTM-CB), transformer constructed using classification-based training (transformer-CB), LSTM constructed using similarity-based training (LSTM-SB), and transformer constructed using similarity-based training (transformer-SB). These different alternatives of encoders produce different representation vectors of texts.

We compared the classification performance of the four aforementioned encoders and their combinations. These encoders and the combinations can be regarded as the methods "proposed" in this paper and are written in bold in Table 4. Table 4 shows classification results when encoders constructed using training sets were used to evaluate answers in validation and test sets. The bold number represents the best F1-score, while underlined numbers represent the best F1-score of a particular class of text encoding. Note that for combinations of multiple encoders, we concatenated the outputs of the encoders into a single long vector.

Table 4. Classification results produced by various alternatives of text encoders

Text Encoder	Classification Result (%)			
	Accuracy	Precision	Recall	F1-Score
LSTM-CB	86.17	70.79	80.53	75.35
Transformer-CB	87.03	76.43	73.16	74.76
LSTM-CB + Transformer-CB	87.32	77.26	73.23	75.19
LSTM-SB	83.22	67.87	68.50	68.18
Transformer-SB	84.94	73.98	65.75	69.62
LSTM-SB + Transformer-SB	85.50	73.83	69.30	71.49
LSTM-CB + LSTM-SB + Transformer-SB	86.74	73.42	77.56	75.44
Average FastText Vector [30]	85.72	78.78	62.38	69.63
TF-IDF [31]	88.44	81.43	72.48	76.69
TF-IDF + LSTM-CB + LSTM-SB + Transformer-SB	87.94	76.54	77.92	**77.22**

*CB means trained using a classification-based approach, SB means trained using a similarity-based approach

We compared text encoders trained using the classification-based approach, as shown in the first three rows of Table 4. The best results were achieved by the individual LSTM encoder (achieved an F1-score of 75.35%), outperforming the individual transformer encoder (achieved an F1-score of 74.76%) and the combination of the two (achieved an F1-score of 75.19%).

We also compared text encoders trained using the similarity-based approach, as shown in the second three rows of Table 4. In this case, the best results were achieved by combining the LSTM encoder and the transformer encoder (achieved an F1-score of 71.49%). The individual transformer encoder came second with an F1-score of 69.62%, and the individual LSTM encoder came third with an F1-score of 68.18%. From the results presented in Table 4, we may infer that LSTM encoders are better suited to classification-based training, while transformer encoders are better suited to similarity-based training. In other words, recurrence and attention mechanisms have their own uniqueness and work differently in various situations.

After we obtained results from the classification-based approach and the similarity-based approach, we combined the best encoders obtained from the two approaches, as shown in the seventh row of Table 4. It turned out that combining the two encoders produced better performance than using the encoders individually. The combined model achieved an F1-score of 75.44%. The combined model was slightly better than the best individual encoder built using the classification-based approach and was clearly better than the best individual encoder built using the similarity-based approach. These results suggested that classification-based and similarity-based approaches have advantages that might complement each other.

Moreover, we compared our text encoders to existing methods found in previous research [30, 31]. Rajagede and Hastuti [30] proposed a method that encoded texts by taking the average of FastText's word embedding vectors. This method represents a basic text encoding approach that calculates some basic aggregate operations over a collection of vectors. Meanwhile, Sari et al. [31] proposed a method that encoded texts using a more classical approach, namely the bag of words approach or the TF-IDF method. Unlike recent methods, the TF-IDF method simply ignores the order of words in input texts and puts more emphasis on the appearance of distinguished keywords. The TF-IDF method was originally developed for information retrieval rather than semantic understanding. Table 4 shows that our combined encoders clearly performed better than the method used in [30]. However, when compared to the TF-IDF method (the bag of words approach) used in [31], our combined encoders performed worse. The TF-IDF approach achieved an F1-score of 76.69%, while our combined encoders achieved an F1-score of 75.44%. These results revealed that although the TF-IDF approach is simple, it is able to extract useful information from Indonesian essay answers effectively. Furthermore, these results suggested that the appearance of certain keywords becomes a strong indicator of the correctness of the answers.

Motivated by this observation, as the last examination of this research, we combined the TF-IDF approach and the best combination of LSTM and transformer encoders. It turned out that combining the aforementioned approaches increased further classification performance. The combined approaches achieved an F1-score of 77.22%, the best performance obtained from the overall comparison. Therefore, these results suggested that the combined different encoders captured complementary information that can

be used to improve the automatic evaluation of Indonesian essay answers. We may also conclude that classical and recent approaches have their own uniqueness that can be employed together to achieve better performance.

CONCLUSION

In this research, we proposed to evaluate and combine some text encoders, including LSTM encoders and transformer encoders, to extract text representations from essay answers in the Indonesian language. Some methods were used to build the encoders, including classification-based training and similarity-based training. We also evaluated some word embedding methods such as Word2vec, GloVe, FastText, and RoBERTa. We compared various alternatives of the proposed text encoders and compared them with some existing methods, such as average FastText vectors and TF-IDF. Our experiments have shown that FastText has become the best word embedding method. The experiments have also shown that the best text encoding has been performed by the combination of three encoders: the LSTM encoder, which was built using classification-based training; the transformer encoder, which was built using classification-based and similarity-based training; and the TF-IDF encoder. The combined encoders were able to achieve an F1-score of 77.22% when evaluated on the UKARA dataset.

REFERENCES

[1] S. Latifi, O. Noroozi, and E. Talaee, "Worked example or scripting? Fostering students' online argumentative peer feedback, essay writing and learning," *Interact. Learn. Environ.*, vol. 31, no. 2, pp. 655–669, Feb. 2023, doi: 10.1080/10494820.2020.1799032.

[2] R. M. Kaipa, "Multiple choice questions and essay questions in curriculum," *J. Appl. Res. High. Educ.*, vol. 13, no. 1, pp. 16–32, Apr. 2020, doi: 10.1108/JARHE-01-2020-0011.

[3] J. E. Barnett and A. L. Francis, "Using higher order thinking questions to foster critical thinking: a classroom study," *Educ. Psychol.*, vol. 32, no. 2, pp. 201–211, Mar. 2012, doi: 10.1080/01443410.2011.638619.

[4] D. Ramesh and S. K. Sanampudi, "An automated essay scoring systems: a systematic literature review," *Artif. Intell. Rev.*, vol. 55, no. 3, pp. 2495–2527, Mar. 2022, doi: 10.1007/s10462-021-10068-2.

[5] S. Burrows, I. Gurevych, and B. Stein, "The Eras and Trends of Automatic Short Answer Grading," *Int. J. Artif. Intell. Educ.*, vol. 25, no. 1, pp. 60–117, Mar. 2015, doi: 10.1007/s40593-014-0026-8.

[6] E. Agirre, D. M. Cer, M. T. Diab, A. Gonzalez-Agirre, and W. Guo, "SEM 2013 shared task: Semantic Textual Similarity," in *International Workshop on Semantic Evaluation*, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:10241043

[7] E. Agirre *et al.*, "SemEval-2014 Task 10: Multilingual Semantic Textual Similarity," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 81–91. doi: 10.3115/v1/S14-2010.

[8] E. Agirre *et al.*, "SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2015, pp. 252–263. doi: 10.18653/v1/S15-2045.

[9] K. Zupanc and Z. Bosnić, "Automated essay evaluation with semantic analysis," *Knowledge-Based Syst.*, vol. 120, pp. 118–132, Mar. 2017, doi: 10.1016/j.knosys.2017.01.006.

[10] V. V. Sayyaparaju, R. B. Pichukala, V. Tummalapalli, and A. Dayal, "Grammar Expert an Automated Essay Scoring Application," in *Recent Advances in Computer Based Systems, Processes and Applications*, CRC Press, 2020, pp. 177–184. doi: 10.1201/9781003043980-21.

[11] K. Sakaguchi, M. Heilman, and N. Madnani, "Effective Feature Integration for Automated Short Answer Scoring," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2015, pp. 1049–1054. doi: 10.3115/v1/N15-1111.

[12] M. Mohler, R. Bunescu, and R. Mihalcea, *Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments*. 2011.

[13] A. Magooda, M. Zahran, M. Rashwan, H. Raafat, and M. Fayek, *Vector Based Techniques for Short Answer Grading*. 2016.

[14] U. Pado, "Get semantic with me! The usefulness of different feature types for short-answer grading," in *Proceedings of the 26th International Conference on Computational Linguistics:*

Technical Papers COLING, 2016, pp. 2186–2195.
[15] H. Zedan and S. Al-Sultan, "The specification and design of secure context-aware workflows," *Expert Syst. Appl.*, vol. 86, pp. 367–384, Nov. 2017, doi: 10.1016/j.eswa.2017.05.078.
[16] A. A. Putri and D. Purnamasari, "SIMPLE-O, the Essay Grading System for Indonesian Language Using LSA Method with Multi-Level Keywords," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:8824249
[17] M. S. Yusop, "Reviving Malay as a lingua franca," *The Jakarta Post*. https://www.thejakartapost.com/news/2013/09/18/m-sharifudin-yusop-reviving-malay-a-lingua-franca.html (accessed Jul. 12, 2024).
[18] Antara, "Indonesian language spoken by 300 million people globally," *Antara*. https://en.antaranews.com/news/263455/indonesian-language-spoken-by-300-million-people-globally-agency (accessed Jul. 12, 2024).
[19] UMS, "The recognition of Bahasa Indonesia as UNESCO language, Available:," *UMS*. https://www.ums.ac.id/en/news/global-pulse/the-recognition-of-bahasa-indonesia-as-unesco-language (accessed Jul. 12, 2024).
[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[21] A. Vaswani *et al.*, "Attention Is All You Need," Jun. 2017.
[22] T. Mikolov, K. Chen, G. . Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Proc. Work. ICLR*, vol. 2013, Jan. 2013.
[23] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
[24] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, Dec. 2017, doi: 10.1162/tacl_a_00051.
[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.04805
[26] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019, doi: https://doi.org/10.48550/arXiv.1907.11692.
[27] G. R. Koch, "Siamese Neural Networks for One-Shot Image Recognition," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:13874643
[28] J. BROMLEY *et al.*, "SIGNATURE VERIFICATION USING A 'SIAMESE' TIME DELAY NEURAL NETWORK," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 07, no. 04, pp. 669–688, Aug. 1993, doi: 10.1142/S0218001493000339.
[29] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, New York, NY, USA: ACM, Jul. 1992, pp. 144–152. doi: 10.1145/130385.130401.
[30] R. A. Rajagede and R. P. Hastuti, "Stacking Neural Network Models for Automatic Short Answer Scoring," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1077, no. 1, p. 012013, Feb. 2021, doi: 10.1088/1757-899X/1077/1/012013.
[31] Y. Sari *et al.*, "A PLATFORM VIEWW OF AUTIMATIC SHORT ANSWER SCORING SYSTEM," in *ICEAP Proceeding 2019*, Kementerian Pendidikan dan Kebudayaan, Dec. 2019, pp. 264–270. doi: 10.26499/iceap.v0i0.228.