IJECBE (2025), **3**, **3**, 579–597
Received (13 January 2025) / Revised (22 May 2025)
Accepted (8 June 2025) / Published (30 September 2025)
https://doi.org/10.62146/ijecbe.v3i3.100
https://ijecbe.ui.ac.id
ISSN 3026-5258

International Journal of Electrical, Computer and Biomedical Engineering

RESEARCH ARTICLE

DEVELOPMENT AND OPTIMIZATION OF SIMPLE-O: AN AUTOMATED ESSAY SCORING SYSTEM FOR JAPANESE LANGUAGE BASED ON BERT, BILSTM, AND BIGRU

Muhammad Aidan Daffa Junaidi and Anak Agung Putri Ratna*

Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Indonesia
*Corresponding author. Email: ratna@eng.ui.ac.id

Abstract

This study aims to develop an Automatic Essay Scoring System (SIMPLE-O) for Japanese short essays, consisting of five essay questions. SIMPLE-O is designed to enhance scoring accuracy by leveraging deep learning models such as BERT, BiLSTM, and BiGRU. The research evaluates deep-level score predictions for each question, rather than only considering the total score across the five questions, to provide more reliable and accurate assessments. SIMPLE-O compares student responses with three predefined answer keys using two similarity measurement methods, Cosine Similarity and Manhattan Distance. The study employs two datasets developed through data augmentation techniques applied to lecturer and student responses. The system is implemented using Python, and its performance is evaluated through analyses of various architectures based on specified hyperparameters. The best results were achieved using a BERT-BiLSTM architecture with the Cosine Similarity method, configured with a batch size of 8, 256 hidden state units, a learning rate of 0.00001, and 100 epochs. The evaluation demonstrated that this approach achieved a Mean Absolute Percentage Error (MAPE) of 7.230% and an average score difference of 5.689. This research highlights the potential of SIMPLE-O for automated scoring of Japanese essays, offering improved accuracy, reliability, and deeper analytical insights.

Keywords: BERT, BiLSTM, BiGRU, Manhattan Distance, Cosine Similarity

1. Introduction

The exponential growth of Artificial Intelligence (AI) has significantly impacted various domains, with education being one of the most prominent beneficiaries. One particularly compelling application within education is Automatic Essay Scoring (AES), a technology designed to evaluate essay-based answers with accuracy and objectivity comparable to human graders. AES systems aim to enhance efficiency, reduce human bias, and ensure consistency in scoring, making them highly valuable in modern educational settings [1] [2]. Since the advent of AES, starting with Project Essay Grader by Ellis Page in 1966, these systems have undergone transformative advancements, shifting from simple feature-based methods to sophisticated machine learning (ML) and deep learning (DL) approaches. Innovations like BERT, GloVe, and fastText have demonstrated remarkable accuracy in capturing the semantic and syntactic nuances of text, marking a new era in AES technology [3] [4].

While AES systems for English-language essays have shown considerable success, the development of such systems for Japanese poses unique challenges. Japanese is characterized by its complex grammar, reliance on context, and the coexistence of multiple writing systems, including kanji, hiragana, and katakana. These features introduce significant linguistic challenges, making traditional approaches less effective [5]. Early attempts, such as those utilizing Latent Semantic Analysis (LSA), laid the groundwork for automated scoring in Japanese but fell short of capturing the intricate linguistic and cultural elements of the language [6]. Addressing these challenges requires adopting advanced NLP models capable of handling the nuances of Japanese text.

Moreover, the Japanese language often omits subjects, relies heavily on context, and contains particles that subtly shift meaning features not commonly found in English. As such, models trained on English corpora cannot be directly applied. To overcome these limitations, this research utilizes pretrained Japanese-specific models such as cl-tohoku/bert-base-japanese, which are trained on native corpora and better equipped to process the morphological and syntactic characteristics of Japanese.

This study introduces SIMPLE-O (Sistem Penilaian Esai Otomatis), an AES system specifically designed for Japanese-language essays. SIMPLE-O leverages cutting-edge deep learning models, including BERT, BiLSTM, and BiGRU, to evaluate the content and context of essays with high performance. The system incorporates advanced similarity evaluation metrics such as Manhattan Distance and Cosine Similarity to measure the alignment between student responses and reference answers. SIMPLE-O represents an evolution from earlier systems that relied on LSA, progressing through various iterations to include neural network architectures. The most recent implementation of SIMPLE-O integrates CNN-Bidirectional LSTM with Manhattan Distance, achieving significant performance improvements with a reported error rate of 29% [7].

Despite the existence of human-based scoring systems that have demonstrated acceptable reliability, they are not without drawbacks. Manual scoring is time-consuming, resource-intensive, and subject to variability between graders. In high-volume academic environments, maintaining consistency becomes increasingly difficult. Therefore, an automated scoring system such as SIMPLE-O is not intended

to replace human judgment but rather to augment and standardize the evaluation process, enabling scalable, objective, and reproducible assessments across institutions.

In practical application, SIMPLE-O can serve as a preliminary scoring tool in large classroom settings or standardized examinations, where immediate feedback is crucial. Its deployment could reduce grading turnaround time, assist educators in identifying learning trends, and provide students with more consistent evaluations thereby supporting improved educational outcomes.

The development of SIMPLE-O aligns with recent trends in AES research, which emphasize the integration of advanced NLP techniques and optimization strategies. Large Language Models (LLMs), such as GPT-4, have demonstrated potential in providing nuanced feedback and high-quality evaluations [8] [9]. Similarly, optimization methods like Artificial Bee Colony (ABC) algorithms have enhanced model performance, addressing issues such as catastrophic forgetting and improving prediction accuracy [10]. These advancements underscore the importance of adopting innovative approaches to optimize AES systems for complex languages like Japanese.

This research aims to achieve two primary objectives, first to develop and optimize SIMPLE-O as a robust AES system capable of accurately evaluating Japanese essays while reducing bias and increasing efficiency and second to contribute to the broader AES field by demonstrating the effectiveness of combining advanced NLP models, similarity metrics, and optimization techniques. Through its contributions, SIMPLE-O seeks to support educational standardization by providing an objective and scalable solution for assessing Japanese essays, ultimately enhancing the learning experience for students and the evaluation process for educators.

2. Literature Review

2.1 Long-Short Term Memory (LSTM)

LSTM is a specialized type of Recurrent Neural Network (RNN) developed to address the vanishing gradient problem commonly observed in traditional RNNs. Initially proposed by Hochreiter and Schmidhuber, LSTM introduces a gating mechanism to regulate the flow of information through the network. This architecture enables the model to selectively retain, forget, or utilize information based on the task requirements, making it particularly effective for processing long sequences [11][12].

The primary components of LSTM include the input gate, forget gate, and output gate, which collectively regulate the flow of information within the network. The input gate controls the inclusion of new information into the memory cell, determining how much of the incoming data should update the cell state. The forget gate plays a crucial role in discarding irrelevant or obsolete information, ensuring that the model does not accumulate unnecessary data over time. Lastly, the output gate governs the portion of the cell state to be outputted, influencing subsequent layers and future time steps. Together, these gates enable LSTM to effectively process and retain relevant information from sequential data [11] [12].

In this study, we employ BiLSTM as described by Graves [12]. Bidirectional Long Short-Term Memory (BiLSTM) extends the capabilities of LSTM by enabling sequential data processing in both forward (chronological) and backward (reverse) directions.

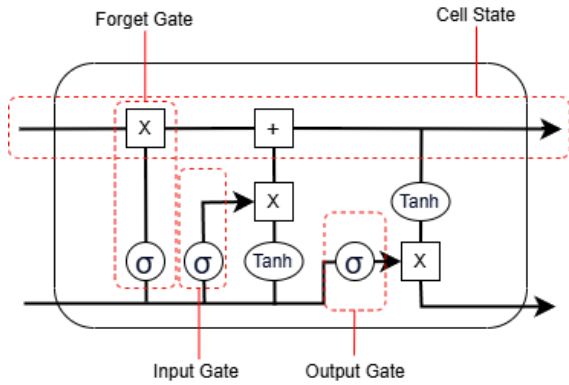

Figure 1. Architecture of LSTM Layer [13]

This is achieved by employing two parallel LSTM layers, one for the forward direction and the other for the backward direction. The outputs from these layers are combined using concatenation, summation, or other mechanisms, providing a richer and more comprehensive representation of the sequence [12] [13].

2.2 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a variant of the Long Short-Term Memory (LSTM) network designed to mitigate the vanishing gradient problem in Recurrent Neural Networks (RNNs). Introduced by Cho et al. in 2014 [1], GRU simplifies the complex structure of LSTM by utilizing only two types of gates: the update gate and the reset gate. This streamlined architecture enables GRU to achieve computational efficiency and faster training while effectively modeling sequential data, making it particularly valuable in Natural Language Processing (NLP) applications [14][15].

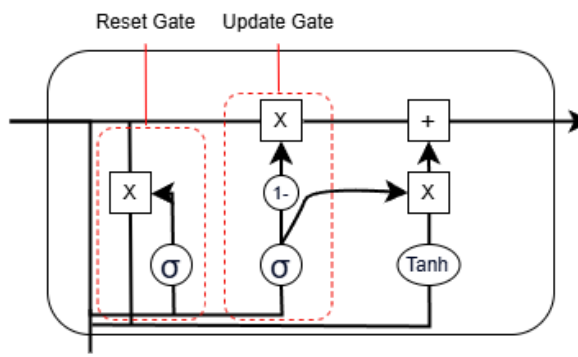

Figure 2. Architecture of GRU Layer [15]

The update gate determines the amount of information from previous time steps that should be retained and passed to the next state. By controlling the blending of past and current information, the update gate helps the model preserve relevant historical context while integrating new input. This balance ensures that the GRU can maintain long-term dependencies in the sequence without unnecessary redundancy. The reset gate, on the other hand, governs how much of the past information should be forgotten at a given time step. When the reset gate value is low, the model effectively "forgets" prior context, allowing it to focus solely on the new input. This functionality is particularly useful when previous information becomes irrelevant to the current computation, enabling the GRU to adapt dynamically to new patterns in the data [14] [15] [16].

The Bidirectional Gated Recurrent Unit (BiGRU) extends the GRU model by processing input sequences in both forward and backward directions. This bidirectional approach enables the model to capture dependencies from both past and future contexts, resulting in a richer understanding of the sequential data. BiGRU achieves this by employing two GRU layers: one that processes the sequence chronologically and another that processes it in reverse. The outputs of these layers are then combined, typically through concatenation, to form a comprehensive representation. This dual-directional processing is particularly advantageous for tasks like sentiment analysis, machine translation, and speech recognition, where understanding context from both directions is crucial. Furthermore, BiGRU retains the computational efficiency of GRU, making it a more resource-friendly alternative to Bidirectional LSTM (BiLSTM) in environments with limited hardware capabilities [15][16].

2.3 Bidirectional Encoder Representations from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer-based model designed to enhance performance in natural language understanding. By leveraging the self-attention mechanism, BERT can consider all words in a sentence simultaneously in a bidirectional context (both left-to-right and right-to-left). This results in richer and more accurate representations compared to previous models such as RNN or LSTM [17][18].

The training process of BERT is divided into two main stages: pre-training and fine-tuning. In the pre-training stage, the model is trained on a large unlabeled text corpus to understand the structure of language and relationships between words. The two main tasks during pre-training are the Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, some words in a sentence are randomly masked, and the model is trained to predict the missing words. Meanwhile, NSP aims to predict whether two given sentences form a logical pair or not. After pre-training, the model can be fine-tuned for specific tasks, such as question answering or sentiment analysis, using labeled data. This fine-tuning is relatively quick and efficient because the model already has a deep understanding of the language due to its pre-training on a massive text dataset [18].

One popular adaptation of BERT for the Japanese language is CL-Tohoku-BERT, specifically designed to handle Japanese text. This model focuses on fine-tuning with Japanese corpora and can handle unique characteristics of the Japanese language, such

as kanji, hiragana, and katakana. Additionally, CL-Tohoku-BERT considers nuances of the Japanese language that are not present in other languages, such as English. The architecture of CL-Tohoku-BERT follows the basic design of BERT developed by Google but incorporates adjustments to cater to the characteristics of Japanese text. This model is available in two variants: BERT-base and BERT-large. In this study, the BERT-base variant is used, consisting of 12 layers with a hidden state dimension of 768 and 12 attention heads [19].

The tokenization process in CL-Tohoku-BERT utilizes the WordPiece method, which splits words into smaller sub-words or tokens. This method is crucial for handling words that are not in the model's vocabulary. Tokenization allows the model to process rare or uncommon words efficiently by breaking them into smaller understandable units. Embedding in CL-Tohoku-BERT consists of three main components: token embeddings, segment embeddings, and position embeddings. Token embeddings provide numerical representations for each token (word or sub-word) in the input. Segment embeddings are used to differentiate between the first and second sentences in the input, which is relevant for tasks like next sentence prediction. Position embeddings provide information about the position of each token in the word sequence in a sentence. The combination of these three embedding types enables CL-Tohoku-BERT to learn rich and contextual representations of Japanese text [18][19][20].

2.4 Distance Metric

Distance metrics are crucial in machine learning for quantifying the closeness or disparity between data points, essential for clustering, recommendation systems, and similarity-based tasks. Among these, Manhattan Distance and Cosine Similarity are widely used due to their unique characteristics and computational efficiency. Manhattan Distance, also known as the L1-norm, calculates the distance between two points in an n-dimensional space by summing the absolute differences of their coordinates. Formally, for two points '$p = (p_1,p_2,...,p_n)$' and '$q = (q_1,q_2,...,q_n)$', the Manhattan Distance is defined as:

$$D(p, q) = \Sigma_{i=1}^{n} |p_i - q_i| \tag{1}$$

This metric models grid-like movement, such as navigating city blocks where only horizontal and vertical paths are possible. It is widely used in clustering and optimization tasks, especially for datasets with sparse or high-dimensional features. Its straightforward calculation and adaptability make it a reliable choice in various machine learning applications [21].

Cosine Similarity, in contrast, measures the similarity between two vectors based on the angle formed between them, rather than their magnitude. It is particularly prevalent in text analysis and natural language processing (NLP), where documents or sentences are represented as vectors in a multi-dimensional space. The formula for Cosine Similarity is as follows:

$$cos(S1, S2) = \frac{S1.S2}{|0S1|\,0\,|0S2|\,0} = \frac{\Sigma_{i=1}^{n} S1_i S2_i}{\sqrt{\Sigma_{i=1}^{n} (S1)^2}\sqrt{\Sigma_{i=1}^{n} (S2)^2}} \tag{2}$$

Unlike Manhattan Distance, this metric evaluates vector orientation rather than magnitude. It is particularly valuable in text analysis and NLP tasks, where semantic relationships between documents or sentences are represented as vectors. By focusing on directionality, Cosine Similarity effectively captures similarities in high-dimensional spaces, making it indispensable for tasks like document retrieval and clustering [22].

Manhattan Distance is well-suited for tasks involving physical distances, grid-based data, or clustering in structured datasets. Cosine Similarity, on the other hand, excels in text-based applications, such as evaluating sentence embeddings or comparing document vectors. Together, these metrics provide versatile tools for improving machine learning performance across diverse scenarios, ensuring accurate and meaningful insights [21] [22].

3. Methodology

3.1 System Design

The SIMPLE-O system is designed to provide automated scoring for Japanese language essays. In this study, the system compares student responses with three reference answers to determine scores based on the degree of similarity. Each question is processed separately, enabling the system to accommodate the unique characteristics of each question. The focus is on short-answer essay questions, emphasizing realibility in the responses. The system outputs predicted scores, which are then compared with actual scores provided by instructors to analyze the differences.

To achieve this functionality, SIMPLE-O leverages deep learning models trained on datasets generated through data augmentation processes. These datasets are designed to enhance data representation and handle a wide range of answer variations. The similarity assessment is conducted using two approaches: Manhattan Distance and Cosine Similarity. Each similarity measurement method operates with a dedicated model to identify the most effective deep learning configuration for each approach.

The system architecture employs a combination of BiLSTM and BiGRU models to capture the temporal and semantic nuances of text responses more comprehensively. Before evaluation begins, student and reference answers undergo a pre-processing stage that includes normalization to remove irrelevant characters and ensure text consistency. Text is tokenized using BERT, transforming each word into tokens and embedding them to produce vector representations through sentence embeddings. This step allows the model to precisely capture the contextual meaning of each response.

After embedding, the vectorized data is trained using models configured with optimized hyperparameters to enhance system performance. The training process incorporates LSTM or GRU layers with Attention Layers to improve the contextual understanding of the models. Separate model instances are created for each essay question, tailored to the specific characteristics of the questions. Various training scenarios are employed with adjusted parameters, such as batch size, the number of hidden state units, and learning rates, to identify the optimal parameter combination.

Model evaluation is conducted using two primary methods, Manhattan Distance which measures linear differences between vectors, and Cosine Similarity which evaluates angular similarity. During testing, student responses are used as evaluation data

to validate the system's generalization capabilities. The automated scores generated by SIMPLE-O are compared with scores from human raters to assess the model's performance. The differences are analyzed to evaluate overall system accuracy and provide recommendations for further iterations.

Through this approach, SIMPLE-O aims to deliver accurate, consistent, and reliable essay scoring, streamlining the academic evaluation process, particularly for Japanese language essays. By systematically comparing multiple models and distance metrics across individualized question sets, SIMPLE-O seeks not only to score accurately, but also to provide adaptable configurations that can be generalized across various test formats and educational institutions.

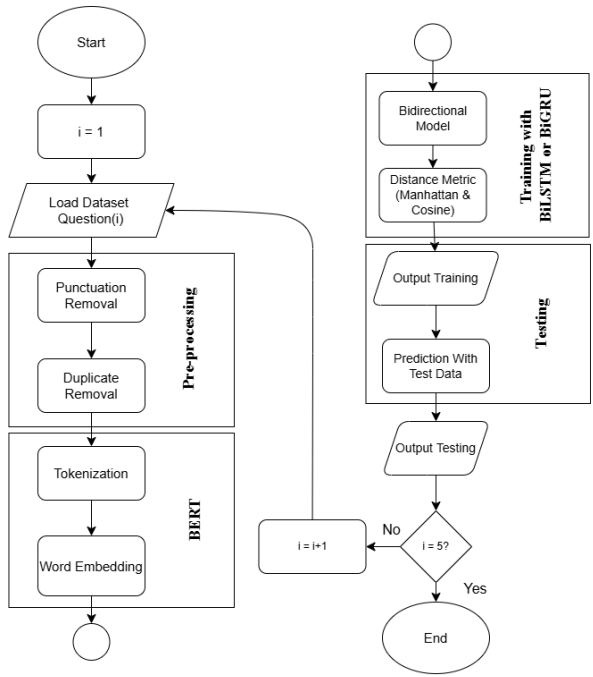

Figure 3. Flowchart of SIMPLE-O Design

3.2 Data Collection & Data Preparation

The development of SIMPLE-O in this study utilized data derived from the previous iterations of the system. This dataset was sourced from the Japanese Studies Program at the Faculty of Humanities, University of Indonesia [23]. The dataset consisted of 38 student essay responses, reference answers prepared by instructors, and augmented data. The student responses addressed five essay questions, while the augmented data consisted of modified versions of both student and instructor answers to create new variations for training purposes. Additionally, this study manually included data with a score of 0, such as irrelevant Japanese words, empty answers, and other similar examples.

The training data in this study encompassed augmented data derived from instructor reference answers and all student responses, irrespective of their scores. This approach aimed to evaluate the impact of data quality and diversity on the model's performance in providing automatic essay assessments. Table 1 outlines the scenarios used to generate augmented data in this study.

The augmentation process was conducted using a Python-based program specifically designed for this research. The augmented data was structured to exhibit varying levels of error, ranging from 0% to 100%. By inserting characters or replacing words randomly, the program generated phrases closely resembling the original input to minimize significant deviations. The resulting augmented sentences often contained intentional structural inaccuracies, thereby enriching the dataset with diverse linguistic patterns. For example, the program created sentences with structural imperfections to simulate real-world variations in language use. A detailed pseudocode of this augmentation process, which illustrates its systematic approach, is presented in Figure 2.

Table 1. Data Augmentation Scenario

Data Type	Initial Score	Target Score Minimum
Instructor Answer Key Data	100	10
Student Answer (Score = 100)	100	10
Student Answer (Score = 95)	95	35
Student Answer (Score = 90)	90	40
Student Answer (Score = 85)	85	35
Student Answer (Score = 80)	80	40
Student Answer (Score = 75)	75	35
Student Answer (Score = 70)	70	40
Student Answer (Score = 65)	65	35
Student Answer (Score = 60)	60	40
Student Answer (Score = 55)	55	35
Student Answer (Score = 50)	50	40
Student Answer (Score = 25)	25	15
Dataset Score 0	-	-

Beyond expanding data volume, the augmentation strategy in this study was carefully designed to reflect the linguistic challenges specific to the Japanese language. Japanese exhibits a unique subject-object-verb (SOV) structure, strict particle usage, and multiple levels of politeness, which can lead to subtle but significant variations in meaning. Furthermore, its writing system consisting of kanji, hiragana, and katakana introduces layers of complexity, particularly for non-native speakers. By introducing structural and lexical errors during augmentation, this research aimed to simulate common and uncommon mistakes made by students, such as missing particles, inappropriate character choice, or incorrect sentence.

This augmentation strategy played a critical role in enhancing the model's ability to handle a wide array of student responses, ensuring better generalization during evaluation. By including a broad spectrum of data, the model's capacity to provide accurate and contextually relevant essay scoring was significantly improved.

```
START
IMPORT libraries (string, random, math, docx, etc.)

DEFINE:
  - Question arrays (Q1, Q2, Q3, Q4, Q5)
  - Alphabet and unique characters from a file (japanese_text)
  - Mistakes list (japanese_text, alphabet, empty)

INITIALIZE variables:
  - Document counter (a = 1)
  - Score iterations initial_score to target_score_minimum

CREATE a new Document object (rd_file)

FOR each score (initial_score to target_score_minimum):
  FOR each combination of answers from Q1 to Q5:
    - SELECT answers based on current combination
    - ADD scores to rd_file

    FOR each selected answer:
      - DETERMINE maximum mistakes based on score
      - RANDOMLY inject mistakes (character insertion, replacement, or deletion)

    SAVE modified answers to a new document with a unique filename
    INCREMENT document counter (a)

SAVE aggregated scores (rd_file) to the specified file path

  END
```

Figure 4. Pseudocode for Data Augmentation

3.3 Data Pre-processing

The pre-processing stage is crucial for preparing textual data in a structured and uniform format, ensuring that SIMPLE-O processes the input effectively to produce optimal outputs. This process begins with punctuation removal, where all irrelevant punctuation marks such as "。" 、", " 「 " , "」 ", "/', "(", ")", ",", ".", "?", and "!" are eliminated from the text. By removing these elements, the system reduces noise and focuses solely on the meaningful content of the text. Following punctuation removal, normalization is performed to identify and remove duplicate entries within the text. This step minimizes redundancy, ensuring that the dataset remains high in quality and free of repetitive information. Such normalization enhances the reliability of subsequent data analysis and processing.

By integrating these stages, the pre-processing pipeline ensures that SIMPLE-O can efficiently handle text data while maintaining both syntactic and semantic integrity. This robust preparation process lays the groundwork for accurate scoring and evaluation, making it a critical component of the system.

3.4 BERT Embedding

In this study, we utilize BERT for embedding, specifically using the pre-trained model 'cl-tohoku/BERT-base-japanese' for processing Japanese language text. The embedding process is preceded by tokenization, where each input is first tokenized using the BERTTokenizer. During tokenization, the input's length is adjusted according to a predefined maximum length. If the input exceeds this maximum, truncation is performed, effectively shortening the input. Conversely, if the input is shorter than the specified length, padding is applied, adding special tokens to ensure the input reaches the desired length.

Once tokenization is complete, BERT proceeds with the embedding phase. This step is essential for converting tokens into dense vector representations that capture the semantic meaning of the input text. BERT employs three types of embeddings

simultaneously: token embedding, segment embedding, and position embedding. These three embeddings are combined and passed through the BERT encoder block, which is composed of multiple transformer layers. The encoder leverages self-attention mechanisms to assess the relationships between tokens within the text, enabling BERT to capture contextual dependencies across the entire sequence. Each attention layer determines the influence of one token on another, which facilitates a more nuanced understanding of the text as a whole.

The transformer encoder not only considers individual tokens but also identifies intricate patterns in sentence structure that influence overall meaning. This enables BERT to generate a richer, context-aware representation of the input text, which is vital for various natural language processing tasks, including classification, information extraction, and translation.

3.5 Model Architecture

This study investigates four distinct model architectures, each designed to assess text similarity in Japanese. All models integrate BERT with different recurrent neural network (RNN) variants and distance metric, as illustrated in Figure 3, which represents the general architecture for all four models.

Model 1 and Model 2 explore the use of BERT with either BiLSTM or BiGRU for text representation and contextual understanding, with the key distinction being the choice of distance metric. Both models use Cosine Similarity as the distance metric, where BERT generates vector representations of the text, and BiLSTM or BiGRU captures contextual information. An attention layer is applied to highlight important parts of the input before calculating the similarity. Model 1 uses BiLSTM, while Model 2 replaces BiLSTM with the more computationally efficient BiGRU, but both models rely on Cosine Similarity to assess the angular distance between vectors.

Model 3 and Model 4 follow a similar structure to the previous models, but they employ Manhattan Distance as the distance metric instead of Cosine Similarity. In these models, BERT is combined with either BiLSTM (Model 3) or BiGRU (Model 4), and an attention layer is applied before calculating Manhattan Distance. Manhattan Distance measures the sum of absolute differences between vector components, providing a different approach to assessing text similarity. The main difference between these models is the choice of neural architecture, with Model 3 using BiLSTM and Model 4 using BiGRU, but both models evaluate similarity based on Manhattan Distance.

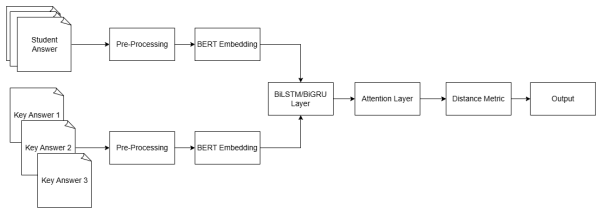

Figure 5. Architecture Diagram of the Proposed Model

The goal of these models is to explore various combinations of neural architectures

and distance metrics, including Cosine Similarity and Manhattan Distance, to determine the most effective model for assessing text similarity in the context of Japanese language processing.

3.6 Experiment Scenario

The evaluation of SIMPLE-O was conducted through various training scenario variations to identify the optimal configuration for assessing student essays. In this process, hyperparameter settings were varied to determine the configuration that minimizes performance discrepancies. Hyperparameter tuning was aimed at optimizing model performance during training, focusing on scenarios with low, medium, and high computational capacity. The goal was to assess how different computational conditions impact the model's ability to generate accurate predictions. Details of the hyperparameters used in the experiments are provided in Table 2. These configurations were selected to evaluate the model's robustness across varying hardware capabilities and to fine-tune its ability to achieve consistent performance across diverse computational environments.

Table 2. Scenario Experiment

Scenario	Batch Size	Epoch	Learning Rate	Hidden State Unit
1	32	15	0.01	128
2	16	30	0.001	256
3	16	60	0.00001	128
4	8	100	0.00001	256

3.7 Metric Evaluation

Metric evaluation is a method used to assess the performance of a machine learning model. In this study, the model's performance was evaluated using residual error, which is the absolute difference between the actual value (y) and the predicted value ($\hat{y}$) generated by the system. The formula for calculating residual error is as follows:

$$ResidualError = |y - \hat{y}| \tag{3}$$

Additionally, another metric used to evaluate the model's performance is Agreement with Human Rater. This metric measures the agreement between the system's predicted value and the human rater's value. The formula for calculating Agreement with Human Rater is:

$$Agrement\ with\ Human\ Rater = 100 - Residual\ Error \tag{4}$$

To further compare with previous studies, the total score (comprising questions 1 through 5) was also evaluated using two additional metrics: Absolute Percentage Error (APE) and Mean Absolute Percentage Error (MAPE). Absolute Percentage Error (APE) measures the percentage error by comparing the absolute difference between the system's predicted value and the human rater's value, normalizing it by the human rater's value, and multiplying by 100 to obtain the percentage error. The formula for APE is:

$$\%Error = \frac{|\hat{y} - y|}{y} x100 \quad (5)$$

Mean Absolute Percentage Error (MAPE) is the average of the APE over all instances in the dataset, providing an overall measure of the model's prediction accuracy. The formula for MAPE is:

$$\%MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|\hat{y} - y|}{y} \quad (6)$$

These metrics were employed to evaluate and compare the model's performance, offering insights into the accuracy of the model and the effectiveness of different evaluation scenarios.

4. Result and Discussion

After the preprocessing stage, the number of data points in the dataset for each question used in the embedding and training phases is as follows:

- Question 1: 9,561 data points
- Question 2: 9,660 data points
- Question 3: 9,246 data points
- Question 4: 9,522 data points
- Question 5: 9,591 data points

The dataset was split with a 70:30 ratio for training and validation. Randomization was applied to the input data to ensure that the model does not only learn specific patterns but can also generalize better to new data during the testing phase.

This study employed four scenarios for each model, with each scenario involving four experiments. Each experiment used 38 data points from students' answers as test data. In every scenario, combinations of several hyperparameters were applied with the primary goal of achieving the best prediction accuracy. Each experiment involved five questions, resulting in five instances of deep learning models for each experiment.

The system's results were compared with test data to predict scores for each question, which were then aggregated to compute final results. The system's performance was evaluated using the residual error metric, which calculates the absolute difference between the system's predicted scores and the scores assigned by human raters. Additionally, the total score evaluation was conducted using absolute percentage error (APE) and mean absolute percentage error (MAPE) metrics.

Tables 3 through 6 present the performance comparisons of various scenarios for the four model architectures proposed in this study. Table 3 summarizes the best results for each scenario using the BERT-BiLSTM architecture with Cosine Similarity, while Table 4 highlights the results for the BERT-BiGRU architecture using the same similarity metric. Similarly, Table 5 details the best results for each scenario with the BERT-BiLSTM architecture utilizing Manhattan Distance, and Table 6 focuses on the BERT-BiGRU architecture with Manhattan Distance.

Table 3. Results of All Scenarios in the BERT-BiLSTM-Cosine Similarity Architecture

BERT-BiLSTM-Cosine Similiarity							
Scenario	**Question 1**	**Question 2**	Question 3	Question 4	Question 5	Avg Residual Error	Agreement With Human Rater
1	9.684	16.184	9.868	17.184	12.421	13.068	86.932
2	6.289	6.500	6.184	8.026	12.395	7.879	92.121
3	8.316	8.737	9.000	9.184	11.711	9.390	90.61
4	4.842	5.500	5.868	5.342	8.368	5.984	94.016

Table 4. Results of All Scenarios in the BERT- BiGRU -Cosine Similarity Architecture

BERT-BiGRU-Cosine Similiarity							
Scenario	**Question 1**	**Question 2**	Question 3	Question 4	Question 5	Avg Residual Error	Agreement With Human Rater
1	7.079	14.500	8.816	14.632	11.342	11.274	88.726
2	5.868	7.737	5.868	10.737	11.763	8.395	91.605
3	8.684	8.684	9.184	10.474	12.711	9.947	90.053
4	6.395	6.289	6.500	6.342	9.316	6.968	93.032

Table 5. Results of All Scenarios in the BERT-BiLSTM-Manhattan Distance Architecture

BERT-BiLSTM-Manhattan Distance							
Scenario	**Question 1**	**Question 2**	Question 3	Question 4	Question 5	Avg Residual Error	Agreement With Human Rater
1	15.921	19.737	20.026	24.053	16.053	19.158	80.842
2	8.421	7.842	9.184	9.000	11.316	9.153	90.847
3	11.184	9.711	14.263	11.553	14.947	12.332	87.668
4	6.789	7.000	10.237	7.000	13.026	8.810	91.19

Table 6. Results of All Scenarios in the BERT-BiGRU-Manhattan Distance Architecture

BERT-BiGRU-Manhattan Distance							
Scenario	**Question 1**	**Question 2**	Question 3	Question 4	Question 5	Avg Residual Error	Agreement With Human Rater
1	15.526	16.553	17.526	22.211	19.658	18.295	81.705
2	7.658	8.921	9.184	10.263	11.447	9.495	90.505
3	13.105	11.237	14.289	17.079	16.053	14.353	85.647
4	7.105	6.763	10.526	9.474	14.263	9.626	90.374

For the BERT-BiLSTM with Cosine Similarity architecture, the Agreement with Human Rater scores, as shown in Table 3, are 86.932 for Scenario 1, 92.121 for Scenario 2, 90.610 for Scenario 3, and 94.016 for Scenario 4. Meanwhile, Table 4 presents the results for the BERT-BiGRU with Cosine Similarity architecture, with scores of 88.726 for Scenario 1, 91.605 for Scenario 2, 90.053 for Scenario 3, and 93.032 for Scenario 4. In comparison, the BERT-BiLSTM with Manhattan Distance architecture, as summarized in Table 5, yields Agreement with Human Rater scores of 80.842, 90.847, 87.668, and 91.190 for Scenarios 1 through 4, respectively. Lastly, Table 6 shows the results for BERT-BiGRU with Manhattan Distance, with corresponding scores of 81.705, 90.505, 85.647, and 90.374.

Overall, the highest Agreement with Human Rater score across all architectures and scenarios is achieved in Scenario 4 of the BERT-BiLSTM with Cosine Similarity

model, with a score of 94.016 and an average residual error of 5.984. This result marks it as the best-performing scenario in this study.

The experiments demonstrated that high-computation hyperparameter configurations, such as those in scenario 4 for each model architecture, consistently yielded better performance compared to the other three scenarios. For example, in the BERT-BiLSTM-Cosine Similarity architecture, the average residual error for scenario 4 was 5.984, which is lower than the averages for scenarios 1, 2, and 3, which were 13.068, 7.879, and 9.390, respectively. This trend was observed across other architectures as well. These results indicate that the augmented dataset effectively mirrors the testing data.

The experimental results also showed that scenarios with a hidden state unit configuration of 256 performed better than other configurations. Scenarios 2 and 4, which utilized 256 hidden state units, consistently recorded lower average residual error values compared to configurations with smaller hidden state units. This advantage can be attributed to the optimal representational capacity of the model, enabling it to capture patterns and relationships in the data without becoming overly complex.

Regarding architecture, BiLSTM models demonstrated superior performance compared to BiGRU models. This can be observed from the lower average residual error values in scenarios involving BiLSTM compared to BiGRU. This advantage can be explained by the more complex structure of BiLSTM, which features two LSTM layers that read data in both forward and backward directions, enabling the model to capture global context in Japanese essays. In contrast, BiGRU, while more computationally efficient, tends to lose critical information due to its simpler mechanism.

For distance metrics, Cosine Similarity consistently outperformed Manhattan Distance in all experiments. Cosine Similarity measures the similarity between vectors based on their angles, while Manhattan Distance only measures absolute distances. This makes Cosine Similarity more suitable for tasks involving textual data with complex vector distributions.

Table 7 provides detailed results of the predicted scores for 38 students in the experiment, showcasing the largest agreement with human raters in Scenario 4. The 'MO' column represents the model's output scores, indicating the values calculated by the machine, while the 'HR' column shows the scores given by human raters. From the evaluation of the total student scores, the average residual error in this scenario was 5.689. The largest residual error was recorded for student number 11, with an error of 17, and the smallest residual error was for student number 8, with an error of 0.8. In terms of percentage error evaluation, the average percentage error was 7.230%. The highest percentage error occurred for student number 11, with an error of 17.53%, while the lowest percentage error was found for student number 15, at 3.45%.

The success of this study can be attributed to the combination of transformer-based architectures such as BERT and the Cosine Similarity metric. BERT's ability to capture contextual information in text data, along with Cosine Similarity's capability to measure vector similarity proportionally, enabled the model to achieve more accurate predictions.

Table 7. Result of Scenario 4

-2*No	Question 1		Question 1		Question 1		Question 1		Question 1		Total Score			
	MO	HR	MO	HR	MO	HR	MO	HR	MO	HR	MO	HR	Residual Error	APE
1	69	70	18	25	92	100	73	80	86	100	67.6	75	7.4	9.87 %
2	88	90	88	90	87	90	85	90	73	90	84.2	90	5.8	6.44 %
3	82	90	87	90	89	95	18	25	36	25	62.4	65	2.6	4.00 %
4	94	100	69	70	88	95	68	75	76	90	79	86	7	8.14 %
5	82	85	0	0	82	85	90	95	65	75	63.8	68	4.2	6.18 %
6	91	95	93	100	93	100	89	95	91	95	91.4	97	5.6	5.77 %
7	**23**	**25**	**79**	**80**	**83**	**90**	**21**	**25**	**66**	**65**	54.4	57	2.6	4.56 %
8	24	25	30	25	0	0	1	0	24	25	15.8	15	0.8	5.33 %
9	95	100	82	85	83	90	23	25	81	85	72.8	77	4.2	5.45 %
10	87	90	90	95	81	90	79	85	82	95	83.8	91	7.2	7.91 %
11	91	95	27	90	99	100	94	100	89	100	80	97	17	17.53 %
12	85	90	86	90	89	100	92	100	84	85	87.2	93	5.8	6.24 %
13	100	100	90	95	87	90	93	100	92	95	92.4	96	3.6	3.75 %
14	93	100	80	85	94	100	24	25	77	90	73.6	80	6.4	8.00 %
15	88	90	80	85	84	85	93	100	75	75	84	87	3	3.45 %
16	86	95	77	80	94	100	74	80	71	85	80.4	88	7.6	8.64 %
17	87	95	88	95	92	100	92	100	93	100	90.4	98	7.6	7.76 %
18	89	90	76	80	91	100	100	100	84	100	88	94	6	6.38 %
19	87	90	94	100	92	100	85	90	77	100	87	96	9	9.38 %
20	86	90	95	100	92	100	93	100	94	100	92	98	6	6.12 %
21	92	100	80	85	94	100	24	25	71	85	72.2	79	6.8	8.61 %
22	91	95	49	50	93	100	88	90	59	60	76	79	3	3.80 %
23	77	80	32	35	91	100	89	95	79	95	73.6	81	7.4	9.14 %
24	87	95	51	50	86	90	51	55	54	60	65.8	70	4.2	6.00 %
25	86	90	62	65	82	85	78	85	72	75	76	80	4	5.00 %
26	60	60	57	50	18	25	32	25	36	25	40.6	37	3.6	9.73 %
27	70	75	68	70	80	85	94	100	67	75	75.8	81	5.2	6.42 %
28	19	25	24	25	20	25	78	85	24	25	33	37	4	10.81 %
29	92	100	92	100	84	90	1	0	90	95	71.8	77	5.2	6.75 %
30	93	100	78	80	86	90	93	100	58	80	81.6	90	8.4	9.33 %
31	67	75	19	25	91	95	79	85	73	85	65.8	73	7.2	9.86 %
32	65	65	20	25	65	70	20	25	27	25	39.4	42	2.6	6.19 %
33	90	100	94	100	93	100	77	85	91	95	89	96	7	7.29 %
34	93	100	51	55	94	100	92	100	58	80	77.6	87	9.4	10.80 %
35	89	95	85	90	92	100	86	90	92	95	88.8	94	5.2	5.53 %
36	94	100	83	85	93	100	88	95	59	60	83.4	88	4.6	5.23 %
37	93	100	21	25	93	100	19	25	68	65	58.8	63	4.2	6.67 %
38	86	95	87	90	85	90	19	25	59	60	67.2	72	4.8	6.67 %
Avg	80.816	85.658	65.316	70.132	82.421	88.289	65.658	70.526	69.816	76.711	72.805	78.263	5.689	7.230 %

Additionally, focused data augmentation tailored to the characteristics of students' answers improved the model's robustness in handling variations in writing styles and errors in Japanese essay data. The average residual error for 38 students was 5.689, indicating the difference between the predicted scores by the system and the scores given by human raters. Moreover, the Mean Absolute Percentage Error (MAPE) for Scenario 4 was 7.230%. These results represent a significant advancement compared to previous studies, which recorded an average MAPE of 29% [7].

5. Conclusion

This study demonstrates the effectiveness of combining transformer-based architectures like BERT with deep learning models such as BiLSTM and BiGRU, alongside advanced similarity metrics like Cosine Similarity, in improving the accuracy of automatic essay scoring systems. The results show that the optimal configuration, particularly in Scenario 4, yielded significantly better performance compared to other configurations, with the BERT-BiLSTM-Cosine Similarity architecture achieving the lowest residual error of 5.689. In the evaluation of individual questions, the best residual error for each question under Scenario 4 was as follows: question 1 achieved a residual error of 4.842, question 2 reached 5.50, question 3 was 5.868, question 4 scored 5.342, and question 5 had a residual error of 8.368. These results not only illustrate the power of deep learning for essay scoring but also highlight the importance of hyperparameter tuning, such as selecting the right number of hidden state units, in maximizing model performance.

The study also demonstrated the effectiveness of data augmentation tailored to student responses, which enhanced the model's ability to generalize across different writing styles and levels of proficiency. In comparison with previous studies, which reported an average MAPE of 29% [7], the proposed system's MAPE of 7.230% marks a significant advancement, providing a strong foundation for future research in this field.

The experimental results also suggest that BiLSTM outperforms BiGRU in terms of predictive accuracy, likely due to its more complex architecture that can capture richer contextual information. Additionally, Cosine Similarity consistently outperformed Manhattan Distance, confirming its superior capability in handling the nuanced nature of textual data.

Despite the promising results, this study is not without limitations. The dataset, while augmented, remains relatively small and may not fully represent the diversity of real-world student responses. Furthermore, the use of pre-trained BERT models could introduce bias, particularly toward essays that follow more formal or structurally rigid patterns, potentially disadvantaging creative or unconventional writing styles.

Looking ahead, future work should focus on exploring larger datasets, incorporating more granular linguistic features, and examining real-time essay scoring applications. Using other augmentation techniques, such as back-translation or paraphrasing, could enrich the dataset and improve model robustness. Further exploration of other transformer models, such as GPT-based models, T5, or RoBERTa, could offer additional insights into improving essay scoring accuracy. The promising results of this study pave the way for further advancements in automatic essay scoring systems, particularly for languages like Japanese, where capturing intricate textual nuances is essential.

References

[1] R. H. Chassab, L. Q. Zakaria, and S. Tiun. "Automatic Essay Scoring: A Review on the Feature Analysis Techniques". In: *International Journal of Advanced Computer Science and Applications* 12.10 (2021). DOI: 10.14569/IJACSA.2021.0121028.

[2] M. Uto. "A review of deep-neural automated essay scoring models". In: *Behaviormetrika* 48.2 (July 2021), pp. 459–484. DOI: 10.1007/s41237-021-00142-y.

[3] Y. Liu. "GEEF: A neural network model for automatic essay feedback generation by integrating writing skills assessment". In: *Expert Systems with Applications* 245 (July 2024), p. 123043. DOI: 10.1016/j.eswa.2023.123043.

[4] A. Sharma. "Modeling essay grading with pre-trained BERT features". In: *Applied Intelligence* 54.6 (Mar. 2024), pp. 4979–4993. DOI: 10.1007/s10489-024-05410-4.

[5] *Language: The Japanese Language*. http://afe.easia.columbia.edu/japan/japanworkbook/Language/lsp.htm. [Accessed: 7-Dec-2024]. 2021.

[6] E. B. Page. "Statistical and linguistic strategies in the computer grading of essays". In: *Proceedings of the 1967 Conference on Computational Linguistics*. 1967. DOI: 10.3115/991566.991598.

[7] A. A. Putri Ratna et al. "Hybrid Deep Learning CNN-Bidirectional LSTM and Manhattan Distance for Japanese Automated Short Answer Grading: Use case in Japanese Language Studies". In: *Proceedings of the 8th International Conference on Communication and Information Processing (ICCIP '22)*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 22–27. DOI: 10.1145/3571662.3571666.

[8] B. Okgetheng and K. Takeuchi. "Estimating Japanese Essay Grading Scores with Large Language Models". In: *Proceedings of the 30th Annual Meeting of the Society of Language Processing (NLP2024)*. Kobe, Japan, 2024.

[9] W. Li and H. Liu. "Applying large Language models for automated essay scoring for non-native Japanese". In: *Humanities and Social Sciences Communications* 11 (2024), p. 723. DOI: 10.1057/s41599-024-03209-9.

[10] R. H. Chassab, L. Q. Zakaria, and S. Tiun. "An optimized BERT fine-tuned model using an artificial bee colony algorithm for automatic essay score prediction". In: *PeerJ Computer Science* 10 (2024), e2191. DOI: 10.7717/peerj-cs.2191.

[11] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[12] A. Graves and J. Schmidhuber. "Framewise phoneme classification with bidirectional LSTM Networks". In: *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*. Vol. 4. Montreal, QC, Canada, 2005, pp. 2047–2052. DOI: 10.1109/IJCNN.2005.1556215.

[13] C. Qin. "Long short-term memory with activation on gradient". In: *Neural Networks* 164 (July 2023), pp. 135–145. DOI: 10.1016/j.neunet.2023.04.026.

[14] K. Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *IEEE Transactions on Neural Networks* (2014). DOI: 10.48550/arXiv.1406.1078.

[15] W. Shang, H. Zhen, and W. Zhang. "Sentiment Analysis of Hybrid Network Model". In: *Proceedings of the 2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)*. Taiyuan, China, 2023, pp. 109–113. DOI: 10.1109/SNPD-Winter57765.2023.10224037.

[16] J. Chung et al. *Empirical evaluation of gated recurrent neural Networks on sequence modeling*. Retrieved from ProQuest. 2014.

[17] A. Vaswani et al. *Attention is all you need*. https://www.proquest.com/working-papers/attention-is-all-you-need/docview/2076493815/se-2. Accessed via ProQuest. 2023.

[18] J. Devlin et al. *BERT: Pre-training of deep bidirectional transformers for Language understanding*. Retrieved from ProQuest. 2019.

[19] CL-Tohoku. *BERT Japanese*. https://github.com/cl-tohoku/BERT-japanese. [Accessed: 16-Dec-2024]. 2020.

[20] C. Sun et al. *How to Fine-Tune BERT for Text Classification?* https://www.proquest.com/working-papers/how-fine-tune-bert-text-classification/docview/2225512747/se-2. Accessed via ProQuest. 2020.

[21] M. S. Ergon Cugler de. *Asymptotic behavior of the Manhattan Distance in n-dimensions: Estimating multidimensional scenarios in empirical experiments*. Retrieved from ProQuest. 2024.

[22] A. W. Qurashi, V. Holmes, and A. P. Johnson. "Document Processing: Methods for Semantic Text Similarity Analysis". In: *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. Novi Sad, Serbia, 2020, pp. 1–6.

[23] S. F. Shalihah et al. "Development of the Oral Examination Assessment System (SIPENILAI) in Japanese Using Latent Semantic Analysis (LSA) Algorithm". In: *Proceedings of the 6th International Conference on Communication and Information Processing (ICCIP '20)*. 2021, pp. 12–19. DOI: 10.1145/3442555.3442558.