applied sciences

Article

EssayGAN: Essay Data Augmentation Based on Generative Adversarial Networks for Automated Essay Scoring

Yo-Han Park [1], Yong-Seok Choi [1], Cheon-Young Park [2] and Kong-Joo Lee [1,*]

[1] Department of Radio and Information Communications Engineering, Chungnam National University, 99 Daehak-ro, Yuseong-gu, Daejeon 34134, Korea; happy005012@naver.com (Y.-H.P.); yongseok.choi.92@gmail.com (Y.-S.C.)
[2] Interaction AI Core, KT Institute of Convergence Technology, Seoul 06763, Korea; park.cheonyoung@kt.com
* Correspondence: kjoolee@cnu.ac.kr; Tel.: +82-42-821-5662

Abstract: In large-scale testing and e-learning environments, automated essay scoring (AES) can relieve the burden upon human raters by replacing human grading with machine grading. However, building AES systems based on deep learning requires a training dataset consisting of essays that are manually rated with scores. In this study, we introduce EssayGAN, an automatic essay generator based on generative adversarial networks (GANs). To generate essays rated with scores, EssayGAN has multiple generators for each score range and one discriminator. Each generator is dedicated to a specific score and can generate an essay rated with that score. Therefore, the generators can focus only on generating a realistic-looking essay that can fool the discriminator without considering the target score. Although ordinary-text GANs generate text on a word basis, EssayGAN generates essays on a sentence basis. Therefore, EssayGAN can compose not only a long essay by predicting a sentence instead of a word at each step, but can also compose a score-rated essay by adopting multiple generators dedicated to the target score. Since EssayGAN can generate score-rated essays, the generated essays can be used in the supervised learning process for AES systems. Experimental results show that data augmentation using augmented essays helps to improve the performance of AES systems. We conclude that EssayGAN can generate essays not only consisting of multiple sentences but also maintaining coherence between sentences in essays.

Keywords: automated essay scoring; data augmentation; EssayGAN; generative adversarial networks; score-rated essay generation; text generation

check for updates

Citation: Park, Y.-H.; Choi, Y.-S.; Park, C.-Y.; Lee, K.-J. EssayGAN: Essay Data Augmentation Based on Generative Adversarial Networks for Automated Essay Scoring. *Appl. Sci.* **2022**, *12*, 5803. https://doi.org/10.3390/app12125803

Academic Editors: Evgeny Nikulchev and Vladimir Borisovich Barakhnin

Received: 16 May 2022
Accepted: 3 June 2022
Published: 7 June 2022

1. Introduction

An essay-writing test is an important way to assess students' logical thinking, critical reasoning, and basic writing skills [1]. Students write an essay about a given prompt, and then human raters manually grade the essay according to a rubric. However, scoring by human raters requires an enormous amount of time and effort. Moreover, it is difficult to maintain consistency throughout the grading process. Therefore, automated essay scoring (AES) can relieve human raters of this burden and provide students with more opportunities to enhance their writing skills.

AES aims to automatically rate an essay by analyzing and identifying the main features required for rating it. Traditional AES systems utilize linguistic features crafted manually by human experts. However, selecting and implementing these features requires a great deal of time and effort from human experts. Recent advances in deep learning research have had a major impact on all natural language applications, including AES. The difficulties of feature selection can, in most cases, be mitigated via a deep learning approach. Thus, AES systems based on a deep learning architecture can achieve better performance than traditional systems without encountering the troubles associated with feature selection.

Human-rated essays are indispensable for training an AES system based on a deep learning architecture. However, the high cost of collecting human-rated essays can be a bottleneck in building a cutting-edge scoring system. Automatic data augmentation can be a solution to the chronic problem of a lack of training data. In this study, we propose an automatic essay generator based on generative adversarial networks (GANs) [2] to augment training data automatically for AES systems.

Recently, GANs have shown successful results in text generation. Conventional GANs consist of two sub-networks: a generator that produces fake data and a discriminator that differentiates real from fake data. The core idea of GANs is to play a min–max game between the discriminator and the generator, i.e., adversarial training. The goal of the generator is to generate data that the discriminator believes to be real.

In this study, we introduce an automatic essay generator based on GAN, which we call EssayGAN. To generate essays rated with different scores, EssayGAN has multiple generators and a discriminator. The number of generators in EssayGAN is determined depending on the score range used for grading the essays. As there are multiple generators in EssayGAN, each generator is dedicated to producing only essays with a specific score. Along with this, the discriminator is trained to distinguish between real and generated essays.

In general, text generation by GAN consists of predicting the next token from a set of pre-defined tokens to create the most authentic text. In the same vein, we consider essay generation as the sequence of prediction of the next sentence based on previously chosen sentences. An ordinary-text GAN model predicts the next word in every step, whereas EssayGAN predicts the next sentence.

There are two reasons why EssayGAN samples a sentence rather than a token. One is that GAN has difficulty in generating long text. Even a cutting-edge GAN model cannot generate a well-organized essay with a length of 150–650 words. When generating an essay on a sentence basis, EssayGAN can create a longer essay. The other reason is that we need to generate an essay with a given target score. EssayGAN can easily compose an essay corresponding to a specific target score by sampling sentences from the essays rated with the target score.

EssayGAN can produce as many essays as needed for a given score. Therefore, it can provide a training dataset large enough to train AES systems based on deep neural networks. Experimental results show that data augmentation by EssayGAN helps to improve the performance of AES systems.

The contributions of this study are as follows:

1. We propose EssayGAN, which can automatically augment essays rated with a given score.
2. We introduce a text-generation model that performs sentence-based prediction to generate long essays.
3. We show that essay data augmentation can improve AES performance.

The rest of the paper is organized as follows. We first explore related studies in Section 2, and we present EssayGAN in Section 3. We then present the experimental results in Section 4, followed by the conclusions in Section 5.

2. Related Studies

2.1. CS-GAN and SentiGAN

There are many variations of GANs. One line of research is based on integrating additional category information into GANs. There are two major approaches to handle category information in GANs. Auxiliary classifier GAN (ACGAN) [3] is one of the most popular architectures for categorical data generation. ACGAN deploys an additional classifier layer into a discriminator. A generator is trained to minimize losses calculated by the discriminator and the classifier in ACGAN. Another approach to categorical data generation is to adopt multiple generators, with each generator being dedicated to each category. SentiGAN [4] is a representative GAN with multiple generators.

In text generation, category sentence GAN (CS-GAN) [5], which is an ACGAN, incorporates category information into the GAN to generate synthetic sentences with categories.

A generator of CS-GAN exploits long short-term memory (LSTM) networks, which are one of the most common methods used for sentence generation. Category information is added at each generation step. As with ACGAN, the training of the generator is led by the discriminator and the classifier in CS-GAN. A policy-gradient method of reinforcement learning is used to update the parameters of the generator in CS-GAN. In [5], the CS-GAN model generated sentences using various categories of information,with 2 to 13 classes for several datasets. Experimental results have shown that generated sentences with categories can perform well in supervised learning, especially in multi-category datasets.

SentiGAN [4] consists of multiple generators and one multi-class discriminator. The number of generators is determined by the number of categories. SentiGAN is designed to generate text with different sentiment labels. Owing to the multiple generators dedicated to a specific sentiment, SentiGAN can focus on accurately generating its own text with a specific sentiment label. SentiGAN's discriminator is a multi-class classifier with k classes and an additional fake class when there are k categories. This architecture allows the generators to focus on generating text with specific sentiment labels and avoid generating unacceptable sentiment text.

2.2. Automated Essay Scoring Based on Pre-Trained Models

The advent of pre-trained models such as BERT [6] has resulted in a huge leap forward in the development of natural language applications. The current trend in natural language processing (NLP) research is to employ a large-scale pre-trained language model as an initial model and then fine-tune it with target training data. A few studies of AES have also started using pre-trained models and have shown successful results.

Pre-trained models such as BERT and XLNet [7] were first adopted as the back-bone of AES systems in [8]. The output embedding of the special token '[CLS]' in the BERT model can be considered as the representation of a whole input sequence. An additional linear layer, acting as a scoring function, takes the embedding as an input, and produces a score as output. After the entire model, including the BERT model and the additional linear layer, is fine-tuned with AES tasks, it can grade an input essay. XLNet is a pre-trained model that diminishes the discrepancy between pre-training and fine-tuning by eliminating mask tokens from pre-training data. Instead, it generalizes the autoregressive pre-training method, which enables the learning of bidirectional contexts by permuting input sequences.

A new method called multi-loss was proposed in [9] to fine-tune BERT models with AES tasks. The AES system has the same architecture as the systems mentioned in [8], except that the final layer outputs two results: a regression score and a rank. Therefore, all weights in the AES system are updated with a combination of regression and ranking losses during a fine-tuning process. As training progresses, the significance of the regression loss increases, whereas that of the ranking loss decreases. The multi-loss objective has proven to be an effective approach to build AES systems based on the BERT model.

The AES system proposed in [10] shows that a simple ensemble of pre-trained models, even with a reduction in size, can achieve significant results in AES tasks. The pre-trained models adopted in [10] are the Albert [11], Reformer [12], Electra [13], and Mobile-BERT [14]. All of these models can be used in small-devices with power-efficient computing.

An adapter module for pre-trained models was introduced in [15]. In general, an entire model needs to be fine-tuned over a certain number of epochs with a target task, no matter how good a pre-trained model is. Therefore, the fine-tuning process requires at least several thousands parameter updates, even in the smallest pre-trained model. Instead of fine-tuning the entire model, ref. [15] employed an adapter module, freezing a part of the model and updating only a few thousand parameters to achieve excellent performance. The adapter module leverages the massive knowledge of pre-trained models to achieve high performance with relatively little fine-tuning.

3. EssayGAN

Typically, GANs that generate text sample a token from a set of pre-defined tokens to compose a new sentence. In the same way, EssayGAN samples a sentence from a sentence collection to compose a new essay.

Since EssayGAN composes a new essay by taking a sentence as it is, the newly generated essay consists of the same sentences as those in the essays of the training dataset. However, the newly generated essay differs from those of the training dataset in that it has multiple sources for its sentencesand a different sentence order. We assume that an essay composed through sampling sentences from several source essays can keep a minimal level of coherence, as long as the source essays share the same prompt.

The overall architecture of EssayGAN is shown in Figure 1. The code is available at https://github.com/YoHanPark512/EssayGAN, accessed on 7 June 2022.

Suppose that essays are rated with r grading rubrics, in which case we use r generators and one discriminator. The value of r can be determined as the number of the range of scores. Because EssayGAN has multiple generators based on the range of scores, each generator can be trained only to compose essays corresponding to the selected score. The discriminator is trained to distinguish whether an input essay is real or fake. SentiGAN [4], with multiple generators, shows that the framework with a mixture of generators can make each generator better to generate their own texts. As the multiple generators can help each other, the quality of texts generated by each single generator can be greatly improved.

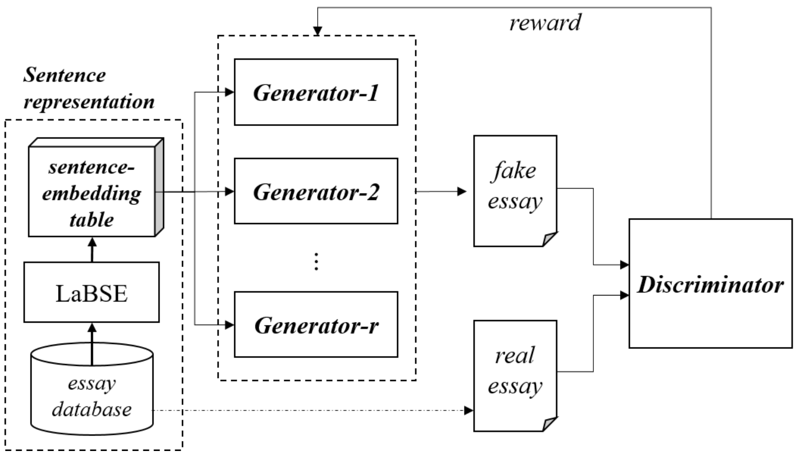

Figure 1. The overall architecture of EssayGAN.

The i-th generator G_i is parameterized with θ_i, denotated as G_{θ_i}, and the discriminator D_ϕ is ϕ-parameterized. The goal of the i-th generator G_{θ_i} is to generate essays that can be evaluated with the score c_i. Each generator G_{θ_i} generates a fake essay that can deceive the discriminator D_ϕ while it discriminates between a real and a fake essay.

Since the generators are trained only on a set of essays rated with a corresponding score, they can generate essays with the given score. Hence, the discriminator does not have to evaluate whether a generated essay is appropriate for the score, but can instead focus on identifying whether the essay is real or fake.

We adopted a reinforcement learning method to train the generators. The output scores of the discriminator are given to the generators as a reward value. The generators are trained alternatively with the discriminator until their parameters reach a stable state.

The following sections describe in detail the components in Figure 1, including sentence representation.

3.1. Sentence Representation

Since the generators of EssayGAN take sentences as their input, every sentence should be represented as a unique embedding vector.

To represent all sentences in the training essay data, we adopted language-agnostic bidirectional encoder representations from transformers (BERT) sentence embedding (LaBSE) [16], which produces language-agnostic cross-lingual sentence embeddings for 109 languages.

LaBSE is a pre-trained model built on a BERT-like architecture and uses the masked language model (MLM) and the translation language model (TLM). It is then fine-tuned using a translation ranking task. The resulting model can provide multilingual sentence embeddings in a single model.

In this study, we calculated the sentence embeddings of all sentences in the training data, using LaBSE in advance, and then saved those embeddings in a sentence-embedding table. After that, the embeddings were exploited by the discriminator and the generators of EssayGAN, which are described in the following sections.

3.2. Discriminator

The goal of the discriminator is to distinguish between human-written and generator-composed essays.

The discriminator is built based on bi-directional LSTM networks, as shown in Figure 2. The i-th sentence, s_i, in an input essay is converted into an embedding vector, E_{s_i}, by looking up the sentence-embedding table, which is described in Section 3.1.

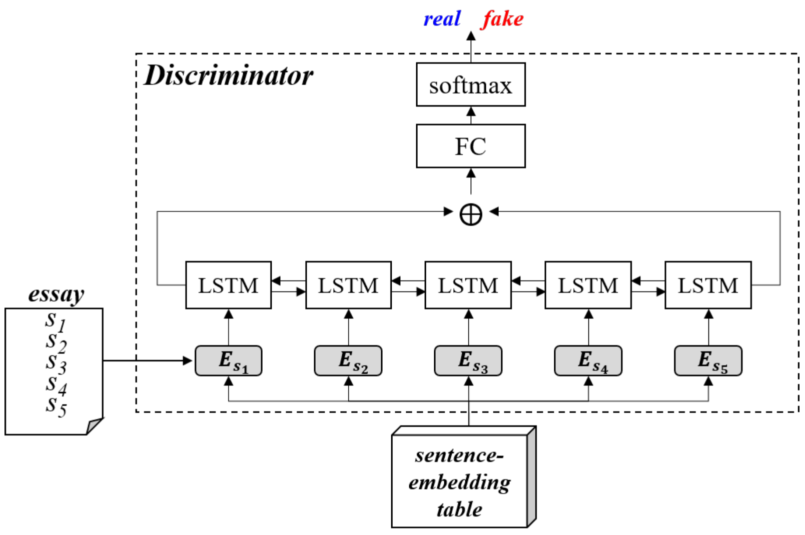

Figure 2. The architecture of the discriminator used in EssayGAN.

Sentence embeddings are fed into the LSTM hidden states, and the first and the last hidden states are concatenated into a representation of the essay. The final layer of the discriminator outputs a scalar value indicating the likelihood that an input essay is real. The discriminator is trained to output a value as close as possible to 1 for real essays and as small as possible for fake essays. The output value of the discriminator is provided to the generators as a reward value.

3.3. Generator and Reinforcement Learning

Figure 3 depicts the architecture of the i-th generator assigned to generate essays scored as c_i. We trained r generators to generate an essay with a specified score. The value of r is determined by the range of scores, which are specified in a scoring rubric.

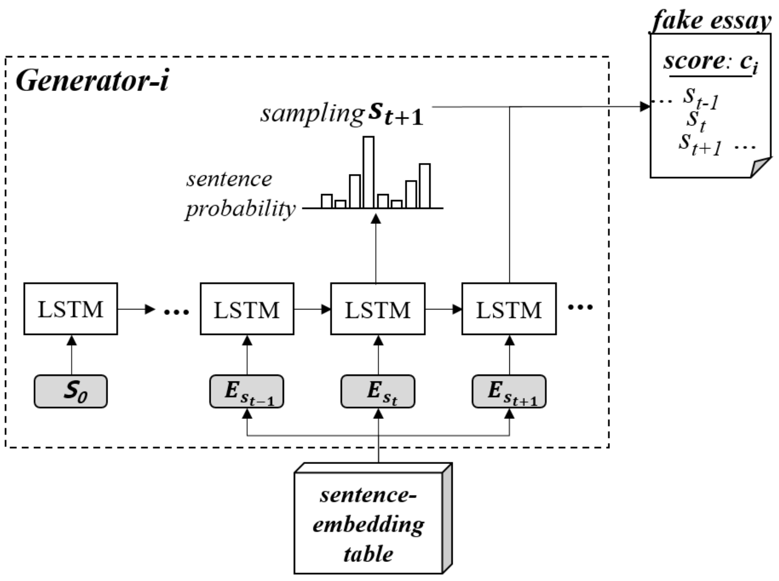

Figure 3. The architecture of the *i*-th generator in EssayGAN.

We utilized LSTM networks as a basic architecture of the generators. The LSTM networks were initially pre-trained with a sentence-level language model using a training dataset and using a conventional maximum likelihood estimation (MLE) method. Therefore, the pre-trained LSTM networks can predict the most likely next sentence based on previously selected sentences. After the pre-training phase, adversarial training was employed to train the generators and the discriminator in turn.

The output layer of each LSTM cell has the same dimensions as a sentence-level one-hot vector that can identify a specific sentence. Each LSTM cell, h_t, can be recursively defined according to Equation (1), and a predicted sentence of the LSTM cell can be defined as in Equation (2).

$$h_t = LSTM(h_{t-1}, E_{s_t}), \tag{1}$$

$$p(\hat{s}_{t+1} S_0, s_1, \ldots, s_t) = softmax(Vh_t + b), \tag{2}$$

where E_{s_t} is the embedding vector of the t-th sentence s_t and S_0 is a start state. V is a weight matrix and b is a bias. The next sentence is chosen through random sampling based on the expectation probability. A new essay is organized according to the sequence of sentences generated by LSTMs.

There is an obstacle involved in applying adversarial training to essay generation. The discriminator can only provide a reward value for a completed essay, whereas the generators require a reward value for incomplete essays at every sampling step. Therefore, to reward the generators at every sampling step, we applied a Monte Carlo search [17] to estimate the next unknown sentence to complete an essay.

The parameter θ_i of the i-th generator G_i was updated by means of the REINFORCE algorithm [18] using rewards provided by the discriminator D_ϕ. The objective of the generator G_i is to maximize the expected reward of Equation (3), in which R_n is the reward for a completed essay with the length n and $Q_{D_\phi}^{G_{\theta_i}}(s, a)$ is the action-value function of a sequence, i.e., the expected accumulative reward starting from state s, taking action a and following the policy G_{θ_i}.

$$J(\theta_i) = \mathbb{E}[R_n S_0 \cdot \theta_i] = \sum_{t=1}^{t=n} G_{\theta_i}(s_t S_{1:t-1}) \cdot Q_{D_\phi}^{G_{\theta_i}}(S_{1:t-1}, s_t), \tag{3}$$

where S_0 is the start state, $S_{1:t-1}$ is the previously generated sentence (current state), and s_t is the selected current sentence.

The action-value function $Q^{G_{\theta_i}}_{D_\phi}(s,a)$ is estimated as the output value of the discriminator, as defined in Equation (4). $D_\phi(S_{1:n})$ is the probability that an essay consisting of n sentences $(s_1, s_2, \ldots, s_{n-1}, s_n)$ is real.

$$Q^{G_{\theta_i}}_{D_\phi}(S_{1:n-1}, s_n) = D_\phi(S_{1:n}), \tag{4}$$

where $S_{1:n-1}$ is a state which specifies a generated sentence $(s_1, s_2, \ldots, s_{n-1})$, whereas s_n is an action which selects the n-th sentence.

To evaluate the action-value for an intermediate state, we applied a Monte Carlo search with a roll-out policy G_β to sample the last unknown $n-t$ sentences. In this work, G_{β_i} has the same parameters as those of the generator G_{θ_i}. Hence, the action-value function can be calculated using Equation (5).

$$Q^{G_{\theta_i}}_{D_\phi}(S_{1:t-1}, s_t) = \begin{cases} \frac{1}{K}\sum_{k=1}^{K} Q^{G_{\theta_i}}_{D_\phi}(S^K_{1:n-1}, s^k_n), S^k_{1:n} \in MC^{G_{\beta_i}}(S_{1:t}; K) & \text{for } t < n \\ Q^{G_{\theta_i}}_{D_\phi}(S_{1:n-1}, s_n), & \text{for } t = n, \end{cases} \tag{5}$$

where $MC^{G_{\beta_i}}(S_{1:t}; K)$ is a set $\{S^1_{1:n}, \ldots, S^K_{1:n}\}$, a K-time Monte Carlo search result.

4. Experimental Results

4.1. *Training EssayGAN and Dataset*

Algorithm 1 provides a full description of how to perform the adversarial training of EssayGAN. To avoid high discrimination results due to poor generation, the training iteration ratio between the generators (g-steps in Algorithm 1) and the discriminator (d-steps in Algorithm 1) was kept at 1:5.

In addition, the generators are prone to forget what they learn from a sentence-basis language model, which guides basic coherence in writing. Therefore, the MLE-based teacher of the generators intervenes in the training phase of EssayGAN, as shown in lines 7–11 (t-steps) of Algorithm 1. In this study, we kept the training iteration ratio between t-steps, g-steps, and d-steps at 1:2:10.

The hyperparameters for training EssayGAN are described in Table 1.

Table 1. The hyperparameters used for training EssayGAN.

	Generator	**Discriminator**
embedding dimension	768 (input) 1536 (hidden)	768 (input) 1536 (hidden)
pre-training epoch(steps)	150	3 (5)
batch size	256	256
learning rate	0.01 (pretrain) 1×10^{-4} (adversarial)	1×10^{-4}
Monte Carlo search rollout K	16	NA
dropout	NA	0.2

In this study, we used the Automated Student Assessment Prize (ASAP) dataset, which is the de facto standard dataset in the AES domain for training and evaluation. The ASAP dataset contains 12,978 essays on eight different prompts (essay topics) that were written by students from grades 7–10. A detailed description of the dataset is given in Table 2.

Algorithm 1 Training EssayGAN for essay data augmentation

```
Input: Generator $\{G_{\theta_i}\}_{i=1}^{i=r}$, Policy models $\{G_{\beta_i}\}_{i=1}^{i=r}$, Discriminator $D_\phi$, Essay dataset $T = \{T_1, T_2, \ldots T_r\}$ assorted by $r$ score ranges
Output: well-trained Generators $\{G_{\theta_i}\}_{i=1}^{i=r}$
1: Initialize the parameters of $\{G_{\theta_i}\}_{i=1}^{i=r}$, $D_\phi$ with random values
2: Pre-train $\{G_{\theta_i}\}_{i=1}^{i=r}$ on $T_i$ using MLE
3: Generate fake essays $F = \{F_1, F_2, \ldots, F_r\}$ using $\{G_{\theta_i}\}_{i=1}^{i=r}$ to train $D_\phi$
4: pre-train $D_\phi$ on $\{T \cup F\}$ by minimizing the cross-entropy loss
5: $\beta_i \leftarrow \theta_i$ for each $i$ in 1:r
6: repeat
7:     for t-steps do
8:         for $i$ in $1 : r$ do
9:             Train $G_{\beta_i}$ on $T_i$ using MLE
10:        end for
11:    end for
12:    for g-steps do
13:        for $i$ in $1 : r$ do
14:            Generate fake essays $\hat{S}_{1:n} = (\hat{s}_1, \ldots, \hat{s}_n) \sim G_{\theta_i}$
15:            for $t$ in $1 : n$ do
16:                Compute $Q(s = \hat{S}_{1:t-1}; a = s_t)$ by (5)
17:            end for
18:            Update the parameters $\theta_i$ via policy gradient
19:        end for
20:    end for
21:    for d-steps do
22:        Generate fake essays $F = \{F_1, F_2, \ldots, F_r\}$ using $\{G_{\theta_i}\}_{i=1}^{i=r}$
23:        Train $D_\phi$ on $\{T \cup F\}$ using the cross-entropy loss
24:    end for
25:    $\beta_i \leftarrow \theta_i$ for each $i$ in 1:r
26: until EssayGAN convergence
```

Table 2. Characteristics of the ASAP dataset. In the column "Type of essay", ARG denotes argumentative essays, RES denotes source-dependent response essays, and NAR denotes narrative essays.

Prompt	Number of Essays	Average Length	Average Sentences per Essay	Score Range (Normalized)	Type of Essay	Number of Unique Sentences
1	1783	350	22.76	2–12 (0–4)	ARG	40,383
2	1800	350	20.33	1–6 (0–3)	ARG	36,359
3	1726	150	6.27	0–3 (0–3)	RES	10,551
4	1772	150	4.63	0–3 (0–3)	RES	8134
5	1805	150	6.60	0–4 (0–4)	RES	11,614
6	1800	150	7.78	0–4 (0–4)	RES	13,457
7	1569	250	11.64	0–30 (0–4)	NAR	17,927
8	723	650	34.76	0–60 (0–4)	NAR	24,943

The number of generators in EssayGAN depends on the score range. However, for prompts 1, 2, 7, and 8, which have a broader score range, EssayGAN requires too many generators, which results in a shortage of data. Therefore, we limited the number of generators for EssayGAN to five. For instance, the scores from 0–60 of prompt 8 were discretized into five partitions using a simple partitioning model. We can obtain a normalized score range by using the partitioning model described in detail in Appendix A.

The normalized score ranges for each prompt are presented in the parentheses of the 'Score range' column of Table 2. In summary, EssayGAN adopted four generators for prompts 2, 3, and 4 and five generators for prompts 1, 5, 6, 7, and 8.

4.2. Characteristics of Augmented Essays

We set two baseline models of data-augmentation for comparisons with EssayGAN. Random is a data-augmentation model that generates a new essay by composing randomly selected sentences from essays with the same target score.

All sentences in each essay in the training dataset were assumed to be sequentially numbered. Random$_{\text{Order}}$ generates a new essay by collecting sentences in non-decreasing order from essays with the same target score to ensure minimal coherence between sentences.

First of all, we observed the characteristics and statistics of the augmented essays, which are summarized in Tables 3 and 4. We introduced the 'number of source essays' and the 'number of sentences in reversed order' to examine the consistency and coherence of the augmented essays according to the augmentation techniques used. We augmented the same numbers of essays as those in the training data for each prompt.

Figure 4 depicts an example of the number of source essays and the number of sentences in reversed order to help readers fully understand these concepts. The number of source essays refers to the number of essays from which sentences of a newly generated essay are extracted. We can reason that the consistency of contents in a newly generated essay scarcely remains when the number of source essays is too large. The number of sentences in reversed order refers to the frequency of the occurrence of a decreasing order of sentences in a newly generated essay. This measurement indirectly indicates how well a newly generated essay is organized to maintain coherence.

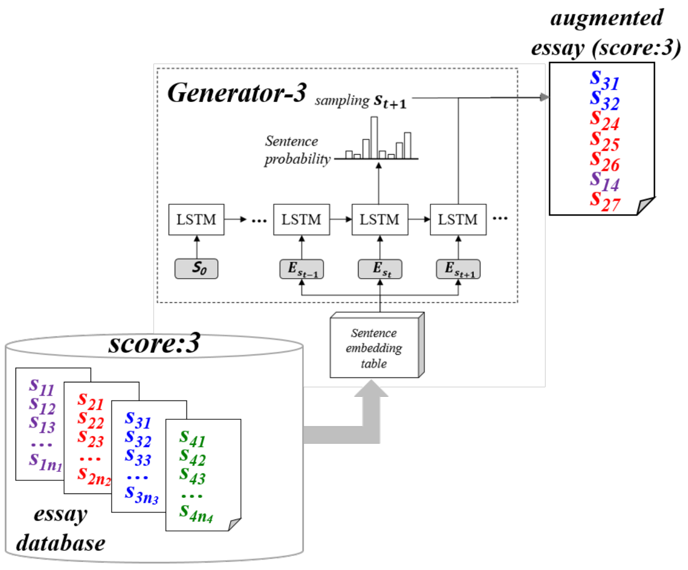
augmented essay (score:3)
Generator-3
sampling s_{t+1}
Sentence probability
LSTM
s_0
$E_{s_{t-1}}$
E_{s_t}
$E_{s_{t+1}}$
s_{31} s_{32} s_{24} s_{25} s_{26} s_{14} s_{27}
Sentence embedding table
score:3
s_{11} s_{12} s_{13} … s_{1n_1}
s_{21} s_{22} s_{23} … s_{2n_2}
s_{31} s_{32} s_{33} … s_{3n_3}
s_{41} s_{42} s_{43} … s_{4n_4}
essay database

Figure 4. An example of an augmented essay, showing the number of source essays and the number of sentences in reversed order.

In the example shown in Figure 4, Generator-3 was trained to compose new essays with a score of 3. In the essay database, each sentence s_{ij} in a source essay indicates a *j*-th sentence in an *i*-th essay. The augmented essay has three different source essays (1, 2, and 3) and one sentence in reversed order between s_{26} and s_{14}.

Table 3 represents the number of source essays according to the augmentation techniques. The Random and Random$_{\text{Order}}$ categories had almost the same numbers of source essays as the average number of sentences per essay because they collected sentences randomly from the source essays. EssayGAN had less than three source essays for prompts

3–7, of which the average number of sentences was 4.63–11.64. EssayGAN trained without the *t*-step (in Algorithm 1) had a higher number of source essays than EssayGAN for all prompts.

Table 3. Comparison of the number of source essays according to the augmentation techniques used.

Prompt	**Number of Source Essays**				
	Random	**Random$_{\text{Order}}$**	**EssayGAN w/o *t*-Step**	**EssayGAN**	**Average Sentences per Essay**
1	18.05	17.85	7.18	4.2	22.76
2	17.96	17.72	8.24	4.8	20.33
3	5.38	5.33	2.45	2.26	6.27
4	4.76	4.72	2.37	2.24	4.63
5	5.64	5.58	2.41	2.23	6.60
6	6.17	6.10	2.78	2.78	7.78
7	10.05	9.88	3.22	2.99	11.64
8	22.42	21.96	10.37	6.82	34.76
AVG	11.30	11.14	4.88	3.54	14.35

A comparison of the number of sentences in reversed order is shown in Table 4. Essays generated from Random$_{\text{Order}}$ did not have a reversed sentence order because they were composed under the strategy of monotonically increasing the sentence order. Random had an average of 5.73 sentences in reversed order in an augmented essay, whereas EssayGAN trained with and without the *t*-step had 0.51 and 1.01 on average, respectively. We can thus reason that EssayGAN can compose a new essay more coherently than the other augmentation techniques that were considered in this study.

Table 4. Comparison of the number of sentences in reversed order according to the augmentation technique.

Prompt	**Number of Sentences in Reversed Order**			
	Random	**Random$_{\text{Order}}$**	**EssayGAN w/o *t*-Step**	**EssayGAN**
1	9.02	0.00	1.41	0.49
2	9.64	0.00	2.01	0.75
3	2.08	0.00	0.07	0.05
4	1.77	0.00	0.04	0.04
5	2.26	0.00	0.04	0.03
6	2.47	0.00	0.08	0.08
7	4.55	0.00	0.30	0.16
8	14.02	0.00	4.14	2.51
AVG	5.73	0.00	1.01	0.51

Next, we used a more explicit metric to examine the coherence of the augmented essays. Coherence is a property of well-organized texts that makes them easy to read and understand [19]. Semantic coherence, proposed in [19], quantifies the degree of semantic relatedness within a text.In this work, we adopted this metric for coherence measurements. The local coherence, as expressed in Equation (6), is a measure of the similarity between consecutive sentences in text T with n sentences, whereas the global coherence, as expressed in Equation (7), is a measure of the similarity between every pair of sentences.

$$coherence_L(T) = \frac{\sum_{i=1}^{n-1} sim(s_i, s_{i+1})}{(n-1)} \tag{6}$$

$$coherence_G(T) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sim(s_i, s_j)}{\sum_{i=1}^{n-1}(i)} \tag{7}$$

In this study, we evaluated the semantic similarity $sim(s_i, s_j)$ between sentences s_i and s_j by means of the cosine similarity between the embedding vectors of the sentences encoded by LaBSE [16].

Table 5 shows comparisons of the local and global coherence between sentences according to the augmentation techniques. 'Training Data + n swapped' is a real essay in which n sentences were replaced with random n sentences from different essays. The coherence of essays augmented by means of the Random and Random$_{\text{Order}}$ techniques was too low to be understood. In terms of coherence, EssayGAN without the t-step performed slightly better than EssayGAN. To sum up, the local coherence of essays augmented by means of EssayGAN was almost close to that of the training data with one swapped sentence, and the global coherence was close to that of the training data with two swapped sentences. We conclude that EssayGAN can generate essays of which the coherence is approximately the same as that of human-written essays in which one or two sentences are incorrectly inserted.

Table 5. Comparison of local and global coherence between sentences in essays according to the augmentation techniques. 'Training Data + 1 swapped' refers to a real essay in which one sentence was replaced with a random sentence. 'Training Data + 2 swapped' refers to one with two swapped sentences. The numbers inside parentheses are values of global coherence. Boldface indicates the highest values of coherence, excluding the 'Training Data'.

	Baseline	**Data Augmentation (×1)**				**Comparison**	
Prompt	**Training Data**	**Random**	**Random$_{\text{Order}}$**	**EssayGAN w/o *t*-Step**	**EssayGAN**	**Training Data + 1 Swapped**	**Training Data + 2 Swapped**
1	0.312 (0.278)	0.221 (0.220)	0.224 (0.221)	0.292 (0.254)	0.293 (0.255)	**0.300** (0.270)	0.289 (0.262)
2	0.320 (0.287)	0.215 (0.214)	0.220 (0.219)	0.294 (0.255)	0.297 (0.259)	**0.303** (0.273)	0.290 (0.264)
3	0.252 (0.244)	0.190 (0.190)	0.189 (0.185)	**0.233** (0.214)	0.232 (0.217)	0.227 (0.221)	0.211 (0.207)
4	0.300 (0.297)	0.226 (0.224)	0.220 (0.221)	**0.266** (0.256)	0.258 (0.250)	0.254 (0.254)	0.235 (0.236)
5	0.334 (0.321)	0.250 (0.249)	0.254 (0.246)	**0.307** (0.286)	**0.307** (0.285)	0.302 (0.292)	0.281 (0.275)
6	0.292 (0.288)	0.256 (0.256)	0.258 (0.255)	**0.282** (0.271)	0.278 (0.268)	0.278 (0.276)	0.269 (0.268)
7	0.331 (0.303)	0.208 (0.209)	0.211 (0.210)	**0.298** (0.260)	0.296 (0.258)	0.291 (0.270)	0.268 (0.250)
8	0.332 (0.284)	0.214 (0.214)	0.219 (0.215)	0.293 (0.246)	0.293 (0.246)	**0.321** (0.276)	0.310 (0.269)
AVG	0.309 (0.288)	0.222 (0.222)	0.224 (0.221)	0.283 (0.255)	0.282 (0.255)	**0.284** (0.266)	0.269 (0.254)

4.3. Experimental Results

The main objective of the following experiments was to show whether generated essays were useful as training data for AES systems. Since building an AES system was not the main concern of this study, we employed a simple deep learning architecture for AES systems. The BERT model with an additional layer for producing essay scores was the AES system used in these experiments. Since EssayGAN can automatically generate essays, our AES system was fine-tuned using both real and generated essays.

In the experiments, we adopted the quadratic weighted kappa (QWK) value [20], which is widely used as a de facto standard metric. It measures the agreement of scoring between human raters and AES systems and varies from 0 (random agreement) to 1 (compete agreement). It is defined in Equation (8).

$$k = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}}, \tag{8}$$

where $O_{i,j}$ of matrix O is the number of essays that receive a rating i from a human rater and a rating j from an AES system. The matrix E is the outer product of the vectors of human ratings and system ratings. The weight entries of matrix W are the differences between rater scores, calculated by $W_{i,j} = \frac{(i-j)^2}{(N-1)^2}$, where N is the number of possible ratings and i and j are the human rating and the system rating, respectively.

To evaluate our approach, we carried out a fivefold cross validation of the ASAP dataset in the same way as in [21,22].

Table 6 summarizes the comparison of QWK correlation according to the data augmentation techniques. The baseline is the QWK correlation of the model trained exclusively on the training data. The simple data-augmentation methods, Random and Random$_{\text{Order}}$, were not useful for the AES system. The performances of those methods were lower than that of the baseline.

From the perspective of data augmentation, EssayGAN performed better than Random and Random$_{\text{Order}}$. In addition, EssayGAN outperformed EssayGAN without the t-step. This indicates that EssayGAN can mitigate the catastrophic forgetting problem that is pervasive during the fine-tuning process when adopting t-step of Algorithm 1.

CS-GAN [5] is designed to generate new sentences tagged with category information. It has three sub-networks: a generator, a discriminator, and a classifier. To compare EssayGAN with other GAN models, we exploited CS-GAN, in which the classifier is replaced by a scorer to generate new essays rated with score information. The performance of the AES system trained on data augmented by CS-GAN was worse than that of the baseline model. Data augmentation by CS-GAN applied to only source-dependent prompts 3–6, with a small scale of scores in this work. (The data augmentation by CS-GAN for prompts 1, 2, 7, and 8 did not work properly, so we did not include the results for these processes. These prompts required CS-GAN to generate argumentative or narrative essays of 250–650 words. This task seemed to be too difficult for CS-GAN with only one generator).

Table 7 shows a performance comparison with the state-of-the-art AES systems. Our AES system has the same architecture as 'BERT-CLS' from [8]; however, it can be trained with more training data thanks to EssayGAN. The R^{2}BERT model achieved the best performance among the comparative models by adopting a multi-loss learning strategy. Our AES system achieved the second best performance on average. Based on the experimental results shown in Table 7, we found that the more training data were augmented elaborately by EssayGAN, the better the performance of the AES system, and this performance could be improved up to the level of the state-of-the-art AES models.

In Table A1 of Appendix B, we present examples of the automatically generated essays for prompt 3.

Table 6. QWK correlation comparison between different augmentation techniques.

	Baseline	**Data Augmentation (×1)**				
Prompt	**Training Data**	**Random**	**Random$_{\text{Order}}$**	**EssayGAN w/o *t*-Step**	**EssayGAN**	**CS-GAN**
1	0.7894	0.7601	0.7663	0.7656	**0.818**	-
2	0.6741	0.6587	0.6507	0.6918	**0.696**	-
3	0.6796	0.6525	0.6462	0.6776	**0.686**	0.665
4	0.8062	0.7695	0.7773	0.8128	**0.828**	0.788
5	0.8038	0.7614	0.7723	0.8054	**0.824**	0.790
6	0.8090	0.7767	0.7794	0.8166	**0.829**	0.799
7	0.8271	0.8056	0.8102	0.8286	**0.865**	-
8	0.6454	0.6989	0.7110	0.7195	**0.761**	-
AVG	0.7543	0.7354	0.7392	0.7647	**0.788**	-

Table 7. Performance comparison with the state-of-the-art AES systems from the literature.

	AES Systems					
Prompt	**BERT–CLS** [8]	**XLNet–CLS** [8]	**R^{2}BERT** [9]	**BERT –Ensemble** [10]	**BERT –Adapter** [15]	**BERT-CLS + Augmented Data of EssayGAN (Ours)**
1	0.792	0.776	0.817	**0.831**	0.743	0.818
2	0.679	0.680	**0.719**	0.679	0.674	0.696
3	0.715	0.692	0.698	0.690	**0.718**	0.686
4	0.800	0.806	0.845	0.825	**0.884**	0.828
5	0.805	0.783	**0.841**	0.817	0.834	0.824
6	0.805	0.793	**0.847**	0.822	0.842	0.829
7	0.785	0.786	0.839	0.841	0.819	**0.865**
8	0.595	0.628	0.744	0.748	0.744	**0.761**
AVG	0.748	0.743	**0.794**	0.782	0.785	0.788

5. Conclusions

In this study, we introduced EssayGAN, which can automatically augment essays rated with specific scores. It consists of multiple generators and a discriminator. EssayGAN can generate as many essays as necessary by sampling sentences from the training set of essays rated with target scores. To the best of our knowledge, EssayGAN is the first attempt to automatically augment text data on a sentence basis. In addition, EssayGAN can maintain coherence between sentences in an augmented essay.

We performed several experiments on the AES task to verify the usefulness of EssayGAN. The experimental results proved that EssayGAN is a reliable data augmentation tool for supervised learning. Therefore, EssayGAN can alleviate the problem of a lack of training data, especially when complex AES systems based on deep learning networks are required. Furthermore, EssayGAN, using multiple generators, can augment essays with a higher quality than that of a conventional GAN using a single generator. A simple AES system, even compared with the state-of-the-art AES models, can yield promising results if it can be trained on more augmented data.

Our future work will include increasing the coherence levels between sentences in an augmented essay and applying EssayGAN to various applications that require augmented data consisting of multiple sentences.

Author Contributions: Conceptualization, C.-Y.P. and K.-J.L.; methodology, Y.-H.P. and Y.-S.C.; software, Y.-H.P.; validation, Y.-H.P.; writing—original draft preparation, K.-J.L. and Y.-H.P.; writing—review and editing, K.-J.L. and Y.-H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by the research fund of Chungnam National University.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Automated Essay Scoring
ASAP	Automated Student Assessment Prize
GANs	Generative Adversarial Networks
LSTM	Long Short-Term Memory
MLE	Maximum Likelihood Estimation
LaBSE	Language-agnostic BERT Sentence Embedding
MC	Monte Carlo
QWK	Quadratic Weighted Kappa

Appendix A. Partition of Score Range

In order to perform the partitioning of the score ranges, we adopted an entropy-based discretization [23] method:

$$H_s(v) = -\sum_{i=1}^{n} p(v_i) \cdot \log p(v_i), \tag{A1}$$

where n is the number of partitions and $p(v_i)$ is the probability of the i-th partition. When considering entropy only, we can determine the number of partitionssuch that a few partitions become too wide. Therefore, we penalized partitions that were too wide to avoid having them take up most of the partitions. Equation (A2) is the entropy of the partition S in terms of the width of the partition.

$$H_s(w) = -\sum_{i=1}^{n} p(w_i) \cdot \log p(w_i), \tag{A2}$$

where $p(w_i)$ is the proportion of width of the i-th partition.

The final partitioning model is described in Equation (A3), which is the weighted average of Equations (A1) and (A2). In this experiment, we set α to 0.35. The number of generators for the prompts 1, 2, 7, and 8 can be decided according to the partition S, maximizing Equation (A3).

$$Partition_s = \max_s(\alpha \cdot H_s(v) + (1+\alpha) \cdot H_s(w)). \tag{A3}$$

Appendix B. Examples of Automatically Generated Essays

Table A1. Examples of augmented essays corresponding to Prompt 3. (The different colors of the generated essays indicate the different sources of sentences).

Prompt 3 (Normalized Score = 3)	
(1) number of source essays = 2	Being the early summer in souther California, the temperatures are going to be hot. He had gotten directions from a group of old men who hadent gone the places they told him about in a long time. Where they said there would be a town it was more like a ghost town. The day that the cyclist was riding was very hot and he was riding through the desert, @CAPS made it worse. Because of the heat he was drinking his water regularly to stay hydrated. This meant he was running out of it fast since the old men had given him directions through places that were no longer open he couldnt get more water. The cyclist also had to go over roads that were in bad condition @CAPS slowed him down even more. Those features at the setting affected the cyclist a lot on his trip.
(2) number of source essays = 3	Author Joe Kurmaskie learns the dangers of a lifeless wilderness during his experience described in do not exceed posted speed limit. As a cyclist riding in one of the most undeveloped areas in California, the lack of water found in this setting proves crippling to his survival. The trails he was directed to had no towns or sources of fresh, drinkable water for days. Another quote that contributed to the cyclists journey was, that smoky blues tune Summer time rattled around in the dry honeycombs of my deteriorating brain. This expressed how dehydrated the summer heat way making him. This meant he was running out of it fast since the old men had given him directions through places that were no longer open he couldnt get more water. The cyclist also had to go over roads that were in bad condition @CAPS slowed him down even more. Those features at the setting affected the cyclist a lot on his trip.
(3) number of source essays = 3	The features of the setting greatly effected the cyclist. He was riding along on a route he had little confidence would end up anywhere. That being the first time ever being on that rode and having only not of date knowledge about it made the cyclist rework. The temperature was very hot, where was little shade, the sun was beating down on him. Next, he came to a shed desperate for water only to find rusty pumps with tainted, unsafe water. Wide rings of dried sweat circled my shirt, and the growing realization that I could drop from heat stroke This states that he @CAPS the danger of not having enough water in his system. Features such as water and heat affected him/her throughout the story. Not having enough water could make him/her lose more sweat and the heat is making him lose even more sweat which can cause extreme heatstroke.
(4) number of source essays = 6	The features of the setting affect the authors dispotion as well as his ability to complete the journey, thus creating an obstacle the author must overcome. At the end of the paragraph five the author writes that I was traveling through the high deserts of California. This setting is important because it adds a sense of urgency to his trip when he starts to run low on water. The terrain changing into short rolling hills didnt help the cyclist. For example, when the cyclist headed from the intense heat, he needed to find a town. Its June in California and the cyclist fears he @MONTH soon suffer from heat stroke if he doesnt find water yet. He said, I was going to die and the birds would pick me clean. Normally, this would not cause too much struggle, but since he was dehydrated and overheated, each hill seemed crippling, My thoughts are that if the setting had been a bit cooler and perhaps on the time period the old men had lived in, the cyclist would have had a much more enjoyable experience.

References

1. Hussein, M.A.; Hassan, H.; Nassef, M. Automated language essay scoring systems: A literature review. *PeerJ Comput. Sci.* **2019**, *5*, e208. [CrossRef] [PubMed]
2. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems 27, Montréal, QC, Canada, 8–13 December 2014; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; pp. 2672–2680.

3. Odena, A.; Olah, C.; Shlens, J. Conditional Image Synthesis with Auxiliary Classifier GANs. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; Proceedings of Machine Learning Research; PMLR: International Convention Centre: Sydney, Australia, 2017; Volume 70, pp. 2642–2651.
4. Wang, K.; Wan, X. SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; pp. 4446–4452. [CrossRef]
5. Li, Y.; Pan, Q.; Wang, S.; Yang, T.; Cambria, E. A generative model for category text generation. *Inf. Sci.* **2018**, *450*, 301–315. [CrossRef]
6. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MI, USA, 2–7 June 2019; Volume 1 (Long and Short Papers); Association for Computational Linguistics: Minneapolis, MI, USA, 2019; pp. 4171–4186. [CrossRef]
7. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019, Volume 32.
8. Rodriguez, P.U.; Jafari, A.; Ormerod, C.M. Language models and Automated Essay Scoring. *arXiv* **2019**, arXiv:1909.09482. .
9. Yang, R.; Cao, J.; Wen, Z.; Wu, Y.; He, X. Enhancing Automated Essay Scoring Performance via Fine-tuning Pre-trained Language Models with Combination of Regression and Ranking. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 1560–1569. [CrossRef]
10. Ormerod, C.M.; Malhotra, A.; Jafari, A. Automated essay scoring using efficient transformer-based language models. *arXiv* **2021**, arXiv:2102.13136.
11. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the 8th International Conference on Learning Representations (ICLR), Online, 27–30 April 2020.
12. Kitaev, N.; Kaiser, L.; Levskaya, A. Reformer: The Efficient Transformer. *arXiv* **2020**, arXiv:2001.04451.
13. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv* **2020**, arXiv:2003.10555.
14. Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; Zhou, D. MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics, Online, 5–10 July 2020; pp. 2158–2170. [CrossRef]
15. Sethi, A.; Singh, K. Natural Language Processing based Automated Essay Scoring with Parameter-Efficient Transformer Approach. In Proceedings of the 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 29–31 March 2022; pp. 749–756. [CrossRef]
16. Zügner, D.; Kirschstein, T.; Catasta, M.; Leskovec, J.; Günnemann, S. Language-Agnostic Representation Learning of Source Code from Structure and Context. *arXiv* **2021**, arXiv:2103.11318.
17. Chaslot, G.; Bakkes, S.; Szita, I.; Spronck, P. Monte-Carlo tree search: A new framework for game AI. In Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE), Stanford, CA, USA, 22–24 Octorber 2008; pp. 216–217.
18. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
19. Lapata, M.; Barzilay, R. Automatic Evaluation of Text Coherence: Models and Representations. In Proceedings of the IJCAI-05, Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, UK, 30 July–5 August 2005; pp. 1085–1090.
20. Chen, H.; He, B. Automated Essay Scoring by Maximizing Human-Machine Agreement. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 Octorber 2013; Association for Computational Linguistics: Seattle, WA, USA, 2013; pp. 1741–1752.
21. Dong, F.; Zhang, Y.; Yang, J. Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL), Vancouver, BC, Canada, 3–4 August 2017; pp. 153–162.
22. Tay, Y.; Phan, M.C.; Tuan, L.A.; Hui, S.C. SkipFlow: Incorporating Neural Coherence Features for End-to-End Automatic Text Scoring. In Proceedings of the Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 5948–5955.
23. Grzymala-Busse, J.W.; Hippe, Z.S.; Mroczek, T. Reduced Data Sets and Entropy-Based Discretization. *Entropy* **2019**, *21*, 51. [CrossRef]